

Procesamiento de datos con Hadoop

© EDICIONES ROBLE, S.L.

< S.L.

Indice

Procesamiento de datos con Hadoop	4
I. Introducción	4
II. Objetivos	4
III. Hadoop. Conceptos básicos	4
IV. ¿Qué es Apache Hadoop?	5
V. Ecosistema Hadoop	8
5.1. Monitorización e ingestá	8
5.2. Procesamiento	11
5.3. Bases de datos	15
VI. ¿Quién usa Hadoop?	16
VII. Módulos de Hadoop	16
7.1. Hadoop Common	16
7.2. Hadoop Distributed File System (HDFS™)	16
7.3. Hadoop YARN	18
7.4. Hadoop MapReduce	18
7.4.1. Programación de trabajos	20
VIII. Instalación de Hadoop	21
8.1. Prerrequisitos	22
8.2. Descargar e instalar Hadoop	23
8.3. Actualizar la máquina virtual	24
8.4. Confirmar versión de Java	25
8.5. Crear y configurar los certificados SSH	26
8.6. Modificar ficheros de configuración	28
8.6.1. <code>~/.bashrc</code>	29
8.6.2. <code>core-site.xml</code>	30
8.6.3. <code>hadoop-env.sh</code>	31
8.6.4. <code>hdfs-site.xml</code>	32
8.6.5. <code>mapred-site.xml</code>	34
8.6.6. <code>yarn-site.xml</code>	36
8.7. Formatear el sistema de archivos de Hadoop	37
8.8. Demonios	38
8.8.1. Arrancar demonios HDFS	39
8.8.2. Arrancar demonios YARN	40
8.8.3. Parar demonios HDFS	41
8.8.4. Parar demonios YARN	41
8.8.5. Arrancar demonios individualmente	41
8.8.6. Interfaces web de Hadoop	44
IX. Ejemplo de comprobación de la instalación	46
X. HDFS	51
10.1. Arquitectura HDFS	52
10.2. Replicación de datos	53
10.3. El nodo secundario	54
10.4. Arrancar Hadoop	55
10.4.1. Comandos básicos de HDFS	56
10.5. Ejemplo de uso de comandos básicos HDFS	57
10.6. API de programación de HDFS	62
10.7. Ejemplos para ejercicio de evaluación	62
XI. MAP REDUCE	68

11.1. Procesamiento paralelo	68
11.2. Fundamentos	70
11.2.1. Dimensión hardware	71
11.2.2. Dimensión software	71
11.3. Aplicación	73
11.3.1. CPU pequeña y pocos datos	74
11.3.2. CPU grande y pocos datos	74
11.3.3. CPU pequeña y muchos datos	74
11.3.4. CPU grande y muchos datos	74
11.4. Ejemplo sencillo de MapReduce	74
11.5. Explicación del modelo MapReduce	76
11.6. Ejemplo	77
11.6.1. Datos de entrada	78
11.6.2. Resultado	80
11.7. Interfaz web	83
11.8. Ejemplo II	83
11.9. Ejemplo de MapReduce con Python: Mapper	88
XII. Resumen	93
Ejercicios	94
Ejercicio 1	94
Se pide	99
Solución	99
Ejercicio 2	105
Solución	106
Recursos	110
Enlaces de Interés	110
Bibliografía	110
Glosario.	111

Procesamiento de datos con Hadoop

I. Introducción

En esta unidad, estudiaremos uno de los máximos exponentes en cuanto a herramientas vinculadas al mundo del Big Data. Hadoop ha sido un punto de inflexión en cuanto a sistemas de almacenamiento y procesamiento paralelo. Estos sistemas no son nuevos, pero Hadoop brinda una serie de características que lo han hecho muy popular.

Conoceremos algo de su historia, sus componentes principales y su funcionamiento. Se harán también ejercicios prácticos para ilustrar cómo trabaja esta herramienta con sus componentes estrella: HDFS como sistema de archivos distribuidos y YARN como gestor de recursos del clúster.

II. Objetivos



Los objetivos que se deben alcanzar tras el estudio de esta unidad son:

- Aprender conceptos básicos de Hadoop y su ecosistema.
- Conocer los distintos demonios que componen Hadoop.
- Aprender a instalar Hadoop en un servidor.
- Conocer las distintas interfaces web de Hadoop.

III. Hadoop. Conceptos básicos

Antes de estudiar Hadoop en profundidad, es necesario refrescar algunos conceptos, en particular:

Ley de Moore

Término informático con origen en la década de los sesenta y en la figura de Gordon Moore (cofundador de Intel), que establece que la velocidad del procesador o la capacidad de procesamiento total de las computadoras se duplica cada doce meses.

Nodo

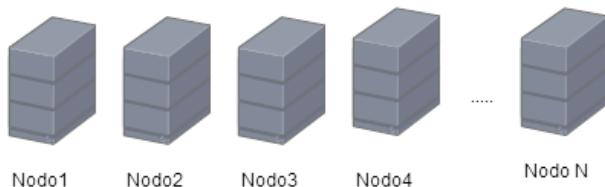
Equipo físico que posee sus propios componentes *hardware* y *software*.



Nodo1

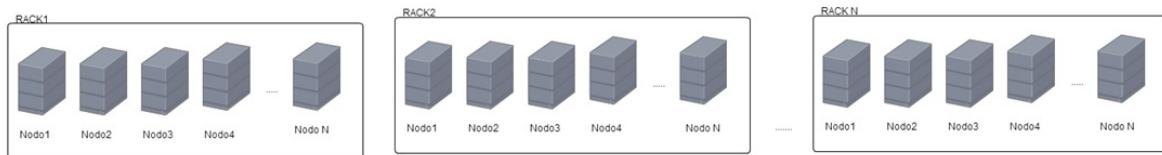
Rack

Conjunto de 1 a n nodos, que se suelen agrupar en estructuras metálicas con este nombre que presentan unas dimensiones establecidas por la industria y localizados en una misma red.



Clúster

Conjunto de 1 a n racks.



Sistemas distribuidos

Conjunto de ordenadores o nodos separados físicamente y conectados entre sí por una red de comunicaciones; cada equipo posee sus componentes de *hardware* y *software* que el programador percibe como un solo sistema. Frente a lo establecido por la ley de Moore, este tipo de sistemas son una alternativa al escalado de servidores.

IV. ¿Qué es Apache Hadoop?



Figura 1. Logo de Apache Hadoop. Fuente: <http://hadoop.apache.org/>

Según la definición del propio proyecto Hadoop, “*The Apache™ Hadoop® project develops open-source software for reliable, scalable, distributed computing*”.¹

¹Apache Hadoop. “What Is Apache Hadoop?” [En línea] URL disponible en: <http://hadoop.apache.org/>

Apache Hadoop es por lo tanto un **framework** de *software* libre que permite escribir y ejecutar aplicaciones en **sistemas distribuidos** para procesar **grandes cantidades de datos**. Está diseñado para escalar de servidores individuales a miles de máquinas, cada una con procesamiento y almacenamiento local. En lugar de confiar en el *hardware* para ofrecer alta disponibilidad, el **framework** en sí está diseñado para detectar y controlar los errores en la capa de aplicación, facilitando un servicio de alta disponibilidad en la capa superior y haciendo transparente para el usuario la posible caída de un nodo.

El **framework** de Hadoop está escrito en **Java** y es una evolución del **subproyecto Nutch**, que a su vez fue un **subproyecto de Lucene**, todos ellos desarrollados por Doug Cutting, que “bautizó” el proyecto con ese nombre en honor al elefante de juguete de su hijo.

Lucene

Lucene: Es un proyecto Java orientado a la **búsqueda e indexación de texto**. Aunque se ha utilizado para la implementación de motores de búsqueda, ya que es capaz de procesar millones de documentos, es útil para cualquier aplicación que requiera indexación y búsqueda de textos completos.

Nutch

Nutch: Es una extensión de Lucene que permite construir un**motor de búsquedas web** usando Lucene como su núcleo. Es capaz de procesar miles de millones de páginas web sin llegar a tener un coste exorbitante. Otra diferencia con Lucene es que Nutch debe correr sobre un clúster distribuido. Para ello, se tuvo que crear una capa que permitiera gestionar el procesamiento distribuido, la redundancia, el balanceo de carga y la recuperación frente a fallos.

Alrededor de 2004, Google publicó dos informes que describían Google File System (GFS)² y el **framework** de MapReduce. Doug Cutting implementó una evolución del **framework** de Nutch basado en estas tecnologías y rápidamente superó las capacidades de Nutch, pues pudo procesar varios cientos de miles de páginas web en clústers de docenas de nodos.

²Sanjay Ghemawat, Howard Gobioff y Shun-Tak Leung. *The Google File System*. 19th ACM Symposium on Operating Systems Principles, Lake George, NY, October 2003. [En línea] URL

disponible en: <https://research.google.com/archive/gfs.html>

Aunque el concepto de programación distribuida está muy extendido, los puntos clave de Hadoop son, por lo tanto:

Accesible

Hadoop se ejecuta en grandes grupos de máquinas en clústers o en nubes tales como [Amazon's Elastic Compute Cloud \(EC2\)](#).

Robusto

Debido a que está pensado para funcionar con *hardware* de equipos básicos, Hadoop está diseñado con la propensión de averías de *hardware* frecuentes. Gracias a ello, es capaz de manejar la mayoría de estos fallos.

Escalable

Hadoop permite el escalado horizontal para gestionar volúmenes de datos más grandes mediante la replicación, añadiendo más nodos al clúster.

Simple

Hadoop permite a los usuarios escribir código eficiente en paralelo empleando conceptos complejos como MapReduce o el sistema distribuido de archivos HDFS diseñado según el GSF de Google.

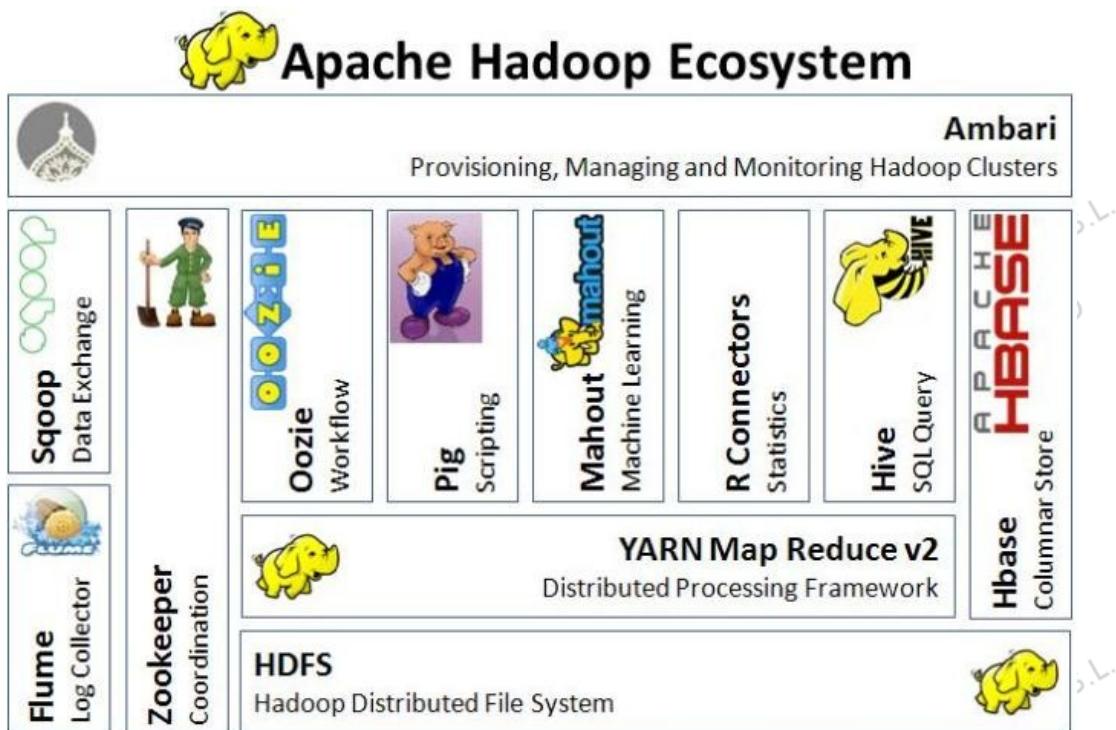


Figura 2. Ecosistema Hadoop.
Fuente: Fundación Apache.

V. Ecosistema Hadoop

El ecosistema de Hadoop crece día a día. Algunos de sus proyectos más conocidos, organizados por su uso, son los que se enuncian a continuación.

5.1. Monitorización e ingestá



Figura 3. Logos de Apache Ambari, Chukwa, Flume y ZooKeeper.



Chukwa

Chukwa: Es un sistema de captura de datos y framework de análisis que trabaja con Hadoop para procesar y analizar grandes volúmenes de *logs*. Incluye herramientas para mostrar y monitorizar los datos capturados.

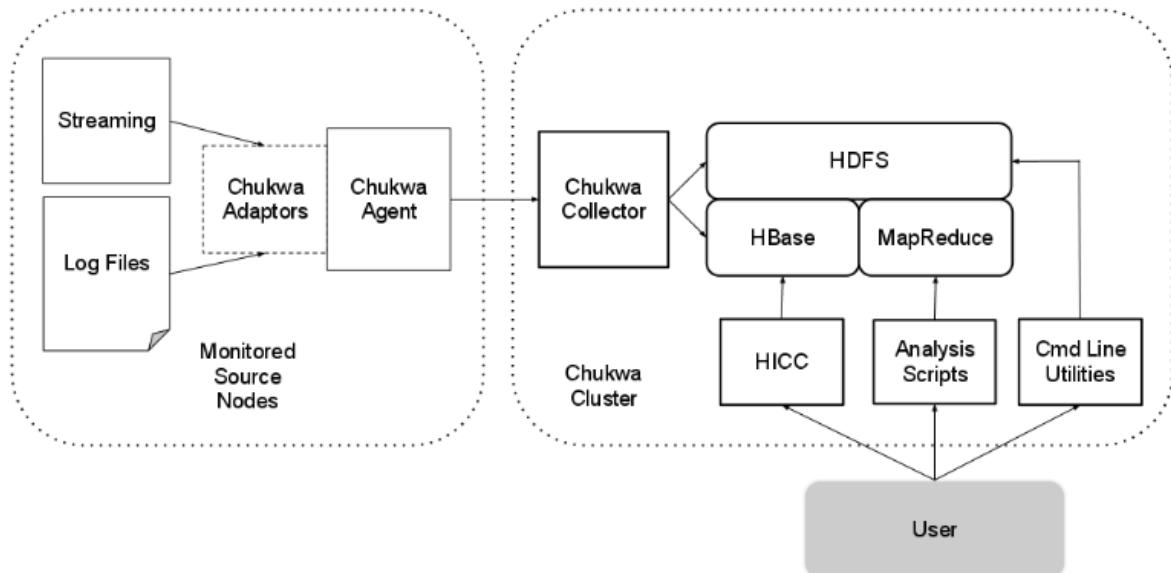


Figura 5. Chukwa. Fuente: Fundación Apache.

Flume

Flume: Es un sistema distribuido para capturar de forma eficiente, agregar y mover grandes cantidades de archivos de *logs* de diferentes orígenes (servidores) a un repositorio central, simplificando el proceso de recolectar estos datos para almacenarlos en Hadoop y poder analizarlos. La principal diferencia con Chukwa es que Chukwa está pensado para ser usado en Batch.

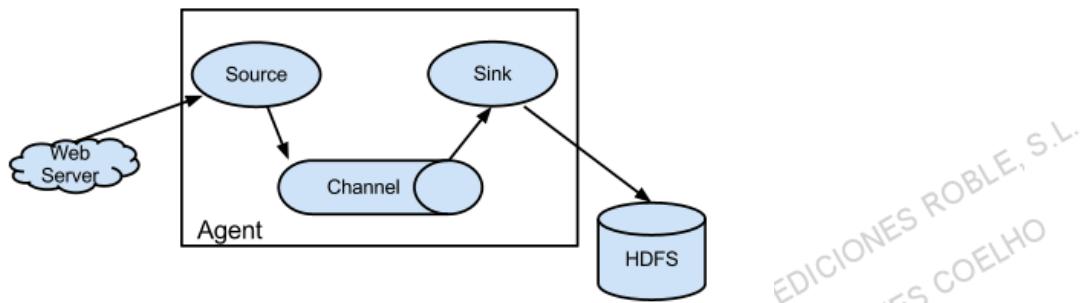


Figura 6. Flume. Fuente: Fundación Apache.

ZooKeeper

ZooKeeper: Servicio que proporciona una infraestructura para gestionar sistemas distribuidos. Facilita un conjunto de primitivas simples que pueden usar las aplicaciones distribuidas para implementar servicios con mayor grado de sincronización y configuración. Es fácil de programar y usa un modelo de datos del estilo de árbol de directorios de los sistemas de archivos. La motivación detrás del proyecto ZooKeeper es liberar a las aplicaciones distribuidas de la responsabilidad de crear servicios de coordinación desde cero.

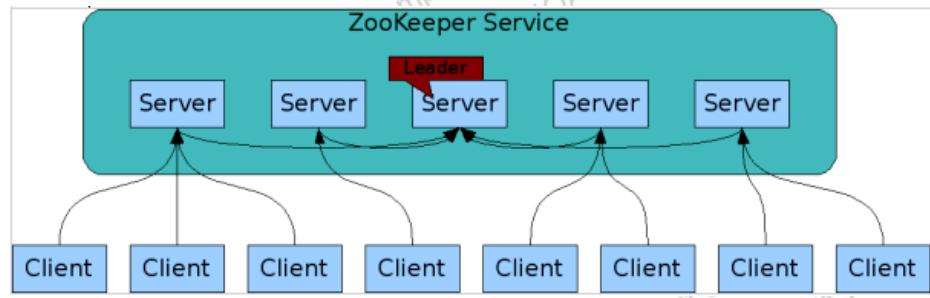


Figura 7. Zookeeper. Fuente: Fundación Apache.

5.2. Procesamiento



Figura 8. Logos de Avro, Drill, Impala, Hive, HCatalog, Lucene, Mahout, Oozie, Spark, Pig, Sqoop, Tez.

Avro

Avro. La serialización es un proceso que convierte un objeto a un formato que se pueda procesar y almacenar. En los proyectos en Hadoop suele haber grandes cantidades de datos, por lo que la serialización se usa para procesarlos y almacenarlos, de forma que el rendimiento en tiempo sea eficiente. Esta serialización puede ser en texto plano, JSON o en formato binario. Con Avro podemos almacenar y leer los datos fácilmente desde diferentes lenguajes de programación. Está optimizado para minimizar el espacio en disco necesario para almacenar datos.

Drill

Drill. Schema-free SQL Query Engine for Hadoop, NoSQL and Cloud Storage. Permite consultar casi cualquier almacén de datos no relacional a través de sintaxis SQL. Drill soporta una gran variedad de bases de datos NoSQL y sistemas de archivos:

<ul style="list-style-type: none"> → HBase. → MongoDB. → MapR-DB. → HDFS. → MapR-FS. → Amazon S3. → Blob Azure. → Google Cloud Storage. → Swift. → NAS. → Archivos locales. 	<pre>SELECT * FROM <u>dfs.root.`/web/logs`</u>; SELECT country, count(*) FROM <u>mongodb.web.users</u> GROUP BY country; SELECT timestamp FROM <u>s3.root.`clicks.json`</u> WHERE user_id = 'jdoe';</pre>
--	---

En una única consulta se pueden unir datos de varios sistemas de almacenamiento. Por ejemplo, puede unirse a una colección de perfiles de usuarios en MongoDB con un directorio de registros de eventos en Hadoop.

Impala

Impala. Apache Impala es la base de datos analítica nativa de código abierto para Apache Hadoop. Impala es soportado por distribuciones como Cloudera, MapR, Oracle y Amazon.

- Impala aumenta el rendimiento de consultas SQL sobre Apache Hadoop, manteniendo una experiencia de usuario familiar.
- Permite consultar datos, ya sea almacenados en HDFS o Apache HBase, incluyendo SELECT, JOIN y funciones de agregación en tiempo real.
- Utiliza los mismos metadatos, la sintaxis SQL (Hive SQL), el controlador ODBC y la interfaz de usuario que Apache Hive.
- Proporciona una plataforma familiar y unificada para las consultas por lotes o en tiempo real.

Hive

Hive. Es una infraestructura de almacén de datos que proporciona métodos de agregación, queries *ad hoc* y análisis de grandes *datasets* almacenados en Hadoop.

HCatalog

HCatalog. Se trata de un servicio de gestión de tablas y almacenamiento que provee un esquema común que permite abstraerse de las diferentes estructuras de tablas, de cómo se almacenan y de los datos que contienen. Así, se facilitan las operaciones de lectura y escritura.

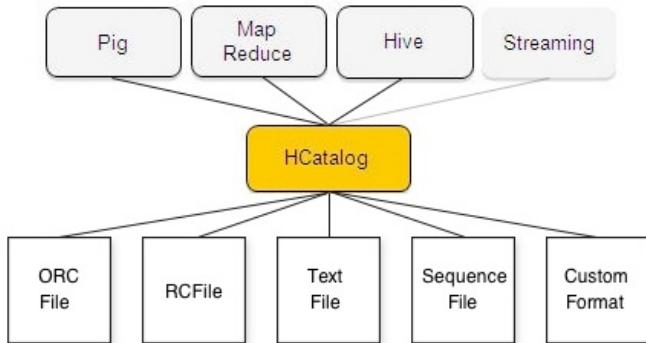


Figura 9. HCatalog. *Fuente:* Fundación Apache.

Lucene

Lucene. Es una librería de búsqueda de texto escrita en Java.

Mahout

Mahout. Es un proyecto de aprendizaje automático y minería de datos empleado para ayudar en la búsqueda de patrones en grandes *datasets*. Posee algoritmos de recomendación, clustering y clasificación.

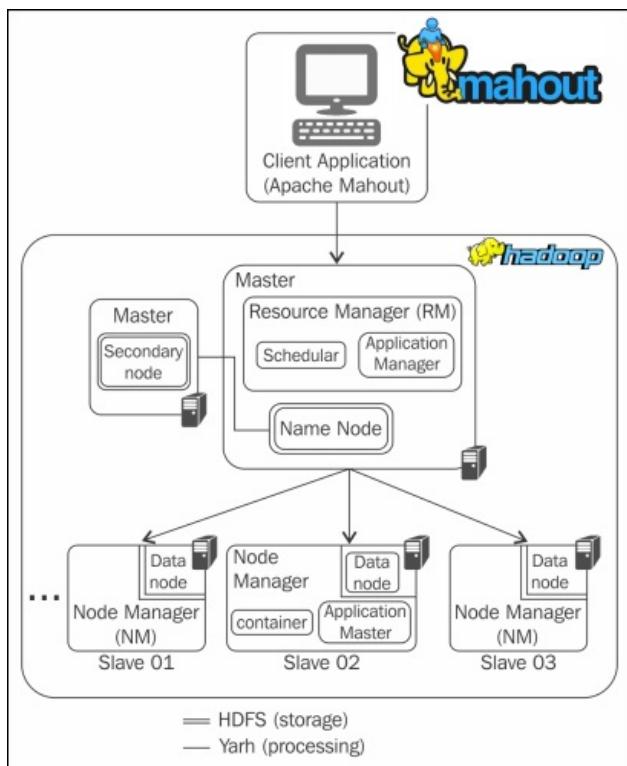


Figura 10. Mahout. Fuente: Fundación Apache.

Oozie

Oozie. Es un sistema orientado a la programación de flujos de trabajo y la coordinación entre cada uno de los procesos. Permite que el usuario pueda definir acciones y las dependencias entre dichas acciones. Un flujo de trabajo en Oozie se define mediante un grafo acíclico llamado Directed Acyclic Graph (DAG)³. Es acíclico puesto que no permite ciclos en el grafo; es decir, solo hay un punto de entrada y de salida y todas las tareas y dependencias parten del punto inicial al punto final sin puntos de retorno.

³Weisstein, Eric W. "Acyclic Digraph" From MathWorld. A Wolfram Web Resource. [En línea] URL disponible en: <http://mathworld.wolfram.com/AcyclicDigraph.html>

Pig

Pig. Lenguaje procedimental de alto nivel empleado para el análisis de datos y ejecución de procesos en paralelo.

Spark

Spark. Motor de computación de datos. Spark proporciona un modelo de programación simple y expresivo que da soporte a múltiples aplicaciones incluyendo ETL (*extract, transform, load*), aprendizaje máquina, procesamiento de cadenas y computación gráfica.

Sqoop

Sqoop. Apache Sqoop (“Sql-to-Hadoop”) es una herramienta diseñada para transferir de forma eficiente información entre Hadoop y sistemas de almacenamiento con datos estructurados, como bases de datos relacionales. Algunas de sus características son:

- Permite importar tablas individuales o bases de datos enteras a HDFS.
- Genera clases Java que permiten interactuar con los datos importados.
- Además, permite importar de las bases de datos SQL a Hive.

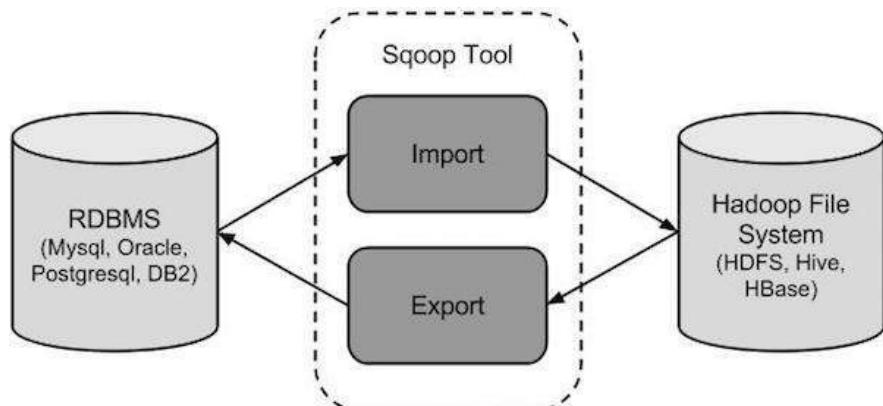


Figura 11. Sqoop. Fuente: Fundación Apache.

Tez

Tez. Framework de programación construido sobre Hadoop YARN, que proporciona un motor flexible y potente para ejecutar mediante flujos de gráficos tareas de procesamiento de datos. Tez ha sido adoptado por Hive y Pig y otros frameworks del ecosistema de Hadoop para reemplazar el motor de ejecución de MapReduce.

5.3. Bases de datos



Figura 11. Logos de Cassandra y HBase.

- **Cassandra.** Base de datos NoSQL distribuida basada en un modelo de almacenamiento clave-valor.

- **HBase.** Base de datos orientada a columnas, escalable y distribuida que permite el almacenamiento de grandes volúmenes de datos.

VI. ¿Quién usa Hadoop?

En la Wiki de Apache Hadoop se ha publicado una lista⁴ de las empresas que actualmente utilizan Hadoop; asimismo, se expone para qué lo usan y algunos otros apuntes. En la figura 12 se recogen algunas de estas empresas:



Figura 12. Logos de Alibaba.com, Ebay, Google, Infochimp, LinkedIn, Rackspace, Twitter, Yahoo e IBM, algunas de las empresas que utilizan Hadoop.

⁴Hadoop Wiki: “Powered by Apache Hadoop” [En línea] URL disponible en: <https://wiki.apache.org/hadoop/PoweredBy>

VII. Módulos de Hadoop

El proyecto Apache Hadoop incluye los siguientes módulos:

- **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 los datos.
- **Hadoop YARN.** Framework para planificación de tareas y gestión de recursos del clúster.
- **Hadoop MapReduce.** Un sistema basado en YARN para el procesamiento en paralelo de grandes conjuntos de datos.

7.1. Hadoop Common

Conjunto de utilidades que dan soporte a otros módulos de Hadoop. Contiene los archivos .jar y scripts necesarios para ejecutar Hadoop. Además, el paquete Hadoop Common proporciona el código fuente, documentación y una sección de contribución que incluye proyectos de la comunidad Hadoop.

7.2. Hadoop Distributed File System (HDFS™)

HDFS es un sistema de archivos distribuido que proporciona alto rendimiento en el acceso a los datos. Fue creado a partir del Google File System (GFS)⁵. HDFS tiene optimizada la lectura y escritura de grandes masas de datos y ficheros grandes. Su diseño reduce la entrada-salida en la red, es escalable y altamente disponible gracias a las técnicas de replicación y tolerancia ante fallos que implementa.

⁵Sanjay Ghemawat, Howard Gobioff y Shun-Tak Leung. *The Google File System*. 19th ACM Symposium on Operating Systems Principles. Lake George, NY, October, 2003. [En línea] URL disponible en: <https://research.google.com/archive/gfs.html>

En un clúster completamente configurado, “hacer correr Hadoop” significa ejecutar una serie de “demonios” o programas residentes en diferentes servidores de la red. Estos demonios tienen roles específicos, algunos existen solo en un servidor y otros en múltiples servidores. Estos demonios incluyen:

NameNode

Solo hay uno en el clúster y actúa por lo tanto como **nodo maestro**. Regula el acceso de entrada/salida a los ficheros por parte de los clientes. Su función principal es mantener en memoria la estructura de cómo se dividen los ficheros en bloques y qué DataNode almacena cada uno de esos bloques. A diferencia del resto de los demonios, donde si hay algún fallo de *hardware* o *software* el clúster Hadoop continúa funcionando o se puede reiniciar, en caso de fallo del NameNode la recuperación no es sencilla.

DataNode

Son los responsables de leer y escribir las peticiones de los clientes, los llamados **nodos esclavos**. Cuando un cliente solicita una lectura o escritura de datos, el fichero se divide en bloques y el NameNode es el encargado de decir dónde se encuentra o almacena cada uno de estos bloques. Además, los DataNodes se comunican con otros nodos para replicar los datos, aumentar la redundancia y favorecer el control frente a errores.

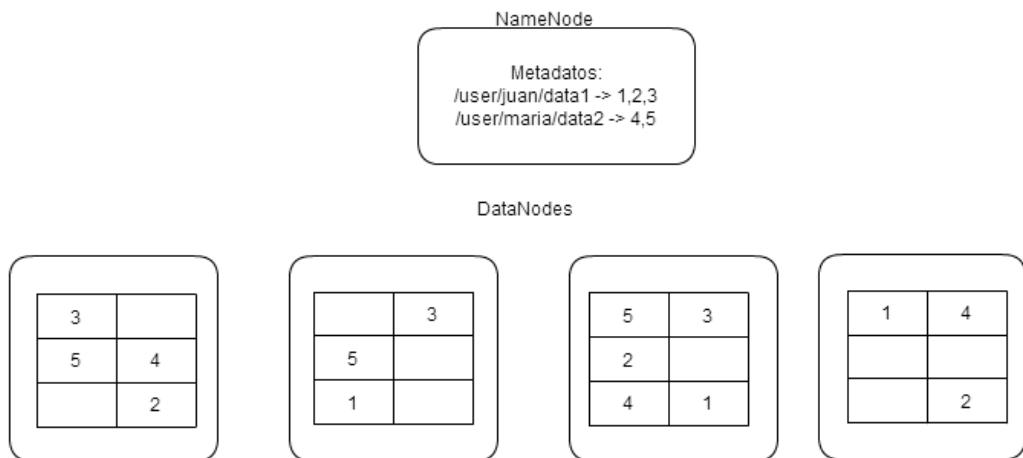


Figura 13. Replicación. Fuente: Fundación Apache.

Secondary NameNode

Su papel principal es mezclar la imagen del **NameNode** con el *log* de transacciones ejecutadas para evitar que el *log* crezca demasiado. Normalmente este demonio corre en una máquina física separada, ya que este proceso requiere mucha CPU y mucha memoria. **Mantiene una copia de la imagen del NameSpace** para que pueda usarse en caso de que el **NameNode** falle.

JobTracker y TaskTracker

Demonios que se explicarán en el apartado de MapReduce.

Además, el sistema de archivos HDFS no se restringe solo al uso de MapReduce. Puede usarse para otras aplicaciones como por ejemplo almacenamiento de datos de HBase, aprendizaje automático de Mahout y operaciones de matriz.

**Actividad de lectura**

Se recomienda la lectura del siguiente *paper* de Google sobre arquitectura de Google File System (GFS):

- Sanjay Ghemawat, Howard Gobioff y Shun-Tak Leung. *The Google File System*. 19th ACM Symposium on Operating Systems Principles. Lake George, NY, October, 2003. [En línea] URL disponible en: <https://research.google.com/archive/gfs.html>

7.3. Hadoop YARN

Framework para planificación de tareas y gestión de recursos del clúster. Es un administrador de recursos que se creó en la primera versión de Hadoop mediante la separación de los motores de procesamiento y manejo de recursos de MapReduce. A menudo se llama YARN al sistema operativo de Hadoop, ya que es el responsable de la gestión y el seguimiento de las cargas de trabajo, el mantenimiento de un entorno multiusuario, la implementación de los controles de seguridad y la gestión de las características de alta disponibilidad de Hadoop.

En Hadoop v.1 los usuarios tenían la opción de escribir programas de MapReduce en Java, Python, Ruby y otros lenguajes de *scripting* usando *streaming* o PIG.

En 2012, YARN se convierte en un subproyecto del ecosistema Hadoop y pasa a llamarse en algunas ocasiones MapReduce 2.0 ya que permite múltiples modelos de procesamiento además de MapReduce, que en algunos casos era algo limitado. Por ejemplo, los clústers de Hadoop pueden correr queries interactivas y aplicaciones de entrada de datos simultáneamente con procesos *batch* de MapReduce.

7.4. Hadoop MapReduce

Es un sistema basado en YARN para el procesamiento en paralelo de grandes conjuntos de datos. El modelo de MapReduce simplifica el procesamiento en paralelo, abstrayendo la complejidad que hay en los sistemas distribuidos. Básicamente, las funciones *map* transforman un conjunto de datos inicial en un conjunto de pares *key/value*. Cada uno de estos elementos se encontrará ordenado por su clave. Asimismo, las funciones *reduce* se usan para combinar los valores (con la misma clave) en un mismo resultado.

Un programa en MapReduce se suele conocer como *job*, la ejecución de un *job* empieza cuando el cliente manda la configuración del *job* al JobTracker; esta configuración especifica las funciones *map*, *combine* (*shuttle*) y *reduce*, además de la entrada y salida de los datos.

A parte de gestionar el sistema de archivos en rack, MapReduce está formado por dos demonios:

JobTracker

Como su propio nombre indica (rastreador de trabajos), se encarga de intentar mantener cada uno de los trabajos que se envían al motor MapReduce lo más cerca posible de los datos. Al tener un sistema de archivos en rack, el JobTracker debe saber qué nodo contiene la información y qué otras máquinas están cerca. Si el trabajo no puede almacenarse en el nodo donde residen los datos, da prioridad a nodos del mismo rack. De este modo se reduce el tráfico de red.

TaskTracker

O (rastreador de tareas). Se encarga de monitorizar los trabajos para relanzarlos en caso de caída. En cada nodo se genera un demonio diferente para evitar que el propio TaskTracker falle si el *job* que tiene que gestionar falla. Cada pocos minutos, el TaskTracker envía información del estado del trabajo al JobTracker. En las versiones anteriores a Hadoop 0.20, si el JobTracker fallaba, todo el trabajo en curso se perdía. Desde la versión 0.21 se han añadido mejoras de autoguardado, de modo que el TaskTracker va haciendo guardados temporales para que cuando se reinicie el JobTracker por un fallo pueda recomenzar el trabajo desde donde se dejó.

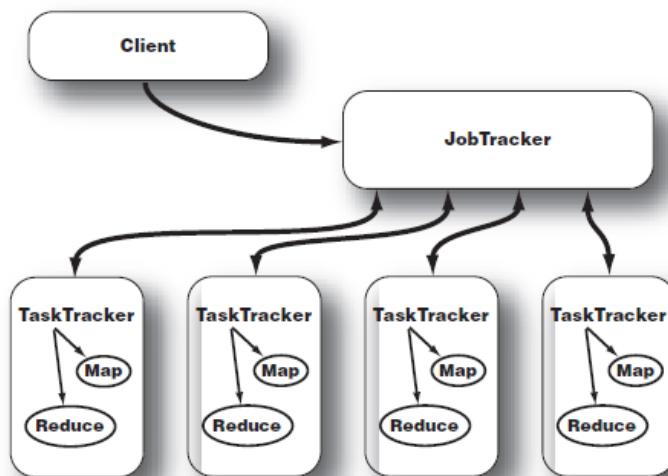


Figura 14. Demonios YARN. *Fuente:* Fundación Apache.



Anotación: Funcionamiento de estos demonios

- La asignación de trabajo de los JobTracker es muy sencilla. Cada TaskTracker tiene un número de ranuras o *slots* disponibles (por ejemplo, "4 slots"). Cada *map* activo o cada *reduce* toma (ocupa) una posición. El JobTracker asigna trabajo para el TaskTracker más cercano a los datos con una ranura disponible. No hay ninguna consideración de la carga activa actual de la máquina asignada ni, por tanto, de su disponibilidad real.
- Si una tarea de seguimiento es muy lenta, se puede retrasar toda la operación MapReduce, especialmente hacia el final de un trabajo, donde todo puede quedar a la espera de una única tarea.

7.4.1. Programación de trabajos

Programación de tiempo

Por defecto Hadoop usa el método FIFO⁶ para programar la ejecución de trabajos desde una cola. En la versión 0.19 el *job scheduler* (programador de trabajos) fue refactorizado fuera de Hadoop, lo cual añadió la posibilidad de usar un programador alternativo.

⁶Wikipedia: "First in, first out". [En línea] URL disponible en: https://es.wikipedia.org/wiki/First_in,_first_out

Programador justo

El *fair scheduler* (programador justo o limpio) fue desarrollado por Facebook⁷. El objetivo del programador es proporcionar una respuesta rápida para trabajos pequeños y calidad de servicio⁸ (QoS, por sus siglas en inglés) para trabajos de producción. El *fair scheduler* se basa en tres conceptos básicos:

- Los trabajos se agrupan en *pools*⁹.
- Cada *pool* tiene asignada una porción mínima garantizada de recursos.
- El exceso de capacidad se distribuye entre trabajos.

Los trabajos que están sin categorizar van a un *pool* por defecto. Las *pools* tienen que especificar el número mínimo de *slots* de *map*, de *reduce*, y un límite en el número de trabajos ejecutándose.

⁷Wikipedia: "Facebook". [En línea] URL disponible en: <https://es.wikipedia.org/wiki/Facebook>

⁸Wikipedia: "Calidad de servicio". [En línea] URL disponible en: https://es.wikipedia.org/wiki/Calidad_de_servicio

⁹Wikipedia: "Pool (informática)". [En línea] URL disponible en: https://es.wikipedia.org/wiki/Pool_%28inform%C3%A1tica%29

Programador de capacidad

El programador de capacidad fue desarrollado por Yahoo¹⁰. Soporta varias funciones similares a las del *fair scheduler*:

- Los trabajos se presentan en las colas.
- A las colas se les asigna una fracción de la capacidad total de recursos.
- Los recursos libres se asignan a las colas más allá de su capacidad total.
- Dentro de una cola, un trabajo con un alto grado de prioridad tendrá acceso a los recursos de la cola.

¹⁰Wikipedia: "Yahoo". [En línea] URL disponible en: <https://es.wikipedia.org/wiki/Yahoo>

VIII. Instalación de Hadoop

En este epígrafe se detalla el proceso de instalación de Hadoop. Debido a que se publican versiones de Hadoop cada poco tiempo, es difícil prever cuál será la última versión en el momento del curso. A continuación se detallan los pasos necesarios para identificar esta versión.

En primer lugar, se debe acceder a la página de publicaciones de Apache Hadoop:

- <http://hadoop.apache.org/releases.html> y seleccionar el enlace de descarga. En este caso hadoop-2.8.0.¹¹

Apache Hadoop Releases

Download

Hadoop is released as source code tarballs with corresponding binary tarballs for convenience. The downloads are distributed via mirror sites and should be checked for tampering using GPG or SHA1.

Version	Release Date	Tarball	GPG	SHA1
3.0.0-alpha3	26 May, 2017	source binary	signature signature	checksum file checksum file
2.8.0	22 March, 2017	source binary	signature signature	77B6A9AS F1324A00.. 3CDC6053 651970C3..
2.7.3	25 August, 2016	source binary	signature signature	227785DC 6E3E6F8.. D489DF3B 08244B90..
2.6.5	08 October, 2016	source binary	signature signature	3AB43F18 73D9951A.. 001AD18D 4B6D0FES..
2.5.2	19 Nov, 2014	source binary	signature signature	139EF872 09C5637E.. 0BDB4850 A3825208..

Este nos lleva a la página:

- <http://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-2.8.0/hadoop-2.8.0.tar.gz>



Google Custom
The Apache Way
Contribute
ASF Sponsors

We suggest the following mirror site for your download:

<http://apache.rediris.es/hadoop/common/hadoop-2.8.0/hadoop-2.8.0.tar.gz>

Other mirror sites are suggested below. Please use the backup mirrors only to download PGP and MD5 signatures to verify your downloads or if no other mirrors are working.

HTTP

<http://apache.rediris.es/hadoop/common/hadoop-2.8.0/hadoop-2.8.0.tar.gz>

<http://apache.uvigo.es/hadoop/common/hadoop-2.8.0/hadoop-2.8.0.tar.gz>

<http://ftp.cixug.es/apache/hadoop/common/hadoop-2.8.0/hadoop-2.8.0.tar.gz>

¹¹ También se puede consultar la siguiente página web con el histórico de versiones de Hadoop, Apache "Hadoop Releases". [En línea] URL disponible en: <https://archive.apache.org/dist/hadoop/common/> Asimismo, es preciso señalar que la velocidad a la que salen las distintas versiones hace difícil saber cuáles serán las versiones disponibles en el momento de realizar la formación. En cuanto a Hadoop, la versión con la que se va a trabajar en este módulo es la 2.8.0.; ya que el funcionamiento de versiones posteriores es muy similar, y Hadoop 2.8.0 es una versión estable y ampliamente implantada en el mundo profesional.

8.1. Prerrequisitos

En este módulo se empleará la **máquina virtual HADOOP** que contiene descargado el fichero con el código compilado de la versión 2.8.0 de Hadoop.

En los siguientes puntos se explicarán los pasos para configurar correctamente esta máquina con anterioridad a la modificación e instalación de Hadoop. Se recomienda ir siguiendo los pasos indicados a medida que se va leyendo la documentación.

El **usuario** de acceso a esta máquina virtual es '**bigdata**' y la **contraseña "bigdata"**.



A continuación, se facilitan los enlaces de descarga de la máquina virtual. La versión de Ubuntu de estas máquinas es Ubuntu 17.10. Las máquinas virtuales tienen descargados los paquetes de *software* de las distintas herramientas que se van a estudiar en este módulo. Se recomienda no cambiar estas versiones, dado que existen limitaciones de compatibilidad en algún ejemplo. Por lo demás, el funcionamiento de versiones posteriores es muy similar. Como ya se ha comentado, es preciso señalar que la velocidad a la que salen las distintas versiones hace difícil saber cuáles serán las versiones disponibles en el momento de realizar la formación.

Opción A

Opción A, máquina con los paquetes descargados sin configurar: [Ubuntu 18.04 + Hadoop 2.8 + paquetes \[DEF\].ova](#). Se puede descargar la máquina virtual en el enlace que aparece a continuación: [máquina sin configurar](#).

Opción B

Opción B, máquina con HDFS y YARN instalados: [Ubuntu 18.04 + Hadoop 2.8 Configurado \(HDFS + Yarn\) + paquetes \[DEF\].ova](#). Se puede descargar la máquina virtual en el enlace que aparece a continuación: [máquina configurada](#).

8.2. Descargar e instalar Hadoop

Una vez vemos cuál es la última versión estable publicada, es momento de proceder a instalarla. En este caso se instalará la versión hadoop-2.8.0.tar.gz; si se selecciona otra versión, basta con sustituir el archivo comprimido por el de la versión elegida.

En primer lugar se obtiene la versión de Hadoop de uno de sus *mirrors*¹¹ con el siguiente comando:

```
wget
```

```
https://archive.apache.org/dist/hadoop/core/hadoop-2.8.0/hadoop-2.8.0.tar.gz
```

```
bigdata@bigdata:~$ sudo wget http://apache.rediris.es/hadoop/common/hadoop-2.8.0/hadoop-2.8.0.tar.gz
--2017-06-19 19:17:27-- http://apache.rediris.es/hadoop/common/hadoop-2.8.0/hadoop-2.8.0.tar.gz
Resolving apache.rediris.es (apache.rediris.es)... 130.206.1.5
Connecting to apache.rediris.es (apache.rediris.es)|130.206.1.5|:80... connected.
Peticion HTTP enviada, esperando respuesta... 200 OK
Longitud: 429928937 (410M) [application/x-gzip]
Guardando como: "hadoop-2.8.0.tar.gz"

hadoop-2.8.0.tar.gz          49%[=====] 201,06M 1,89MB/s eta 2m 3s
```



IMPORTANTE: como Hadoop va introduciendo nuevas versiones periódicamente y puede que el paquete ya no esté disponible cuando el alumno curse este módulo, se ha dejado el fichero descomprimido en la ruta: /home/bigdata/

```
bigdata@bigdata:~$ ls
Descargas  Escritorio      hadoop-2.8.0.tar.gz  Música    Público
Documentos Documentos  examples.desktop  Imágenes  Plantillas  Vídeos
```

Tras descargar el fichero, se descomprime mediante el siguiente comando:

```
tar xfz hadoop-2.8.0.tar.gz
```

```
bigdata@bigdata:~$ tar xfz hadoop-2.8.0.tar.gz
bigdata@bigdata:~$ ls
Descargas  Documentos  Escritorio  examples.desktop  hadoop-2.8.0  hadoop-2.8.0.tar.gz  Imágenes  Música  Plantillas  Público  Vídeos
```

A continuación se mueve el directorio extraído a otra ruta usando el siguiente comando:

```
mv hadoop-2.8.0 /home/bigdata/hadoop
```

Por último, se borra el fichero comprimido, que ya no es necesario:

```
rm hadoop-2.8.0.tar.gz
```

```
bigdata@bigdata:~$ rm hadoop-2.8.0.tar.gz
rm: ¿borrar el fichero regular 'hadoop-2.8.0.tar.gz' protegido contra escritura? (s/n) s
bigdata@bigdata:~$ ls
Descargas  Documentos  Escritorio  examples.desktop  hadoop  Imágenes  Música  Plantillas  Público  Vídeos
```

¹¹Apache Hadoop: “Apache Hadoop Releases” [En línea] URL disponible en: <http://hadoop.apache.org/releases.html>

8.3. Actualizar la máquina virtual

Para actualizar la máquina virtual, se debe ejecutar el siguiente comando:

```
sudo apt-get update
```

```
bigdata@bigdata:~$ sudo apt-get update
Obj:1 http://es.archive.ubuntu.com/ubuntu zesty InRelease
Des:2 http://es.archive.ubuntu.com/ubuntu zesty-updates InRelease [89,2 kB]
Des:3 http://es.archive.ubuntu.com/ubuntu zesty-backports InRelease [89,2 kB]
Des:4 http://security.ubuntu.com/ubuntu zesty-security InRelease [89,2 kB]
Ign:5 http://repo.mongodb.org/apt/ubuntu xenial/mongodb-org/3.4 InRelease
Obj:6 http://repo.mongodb.org/apt/ubuntu xenial/mongodb-org/3.4 Release
Descargados 268 kB en 6s (41,5 kB/s)
Leyendo lista de paquetes... Hecho
bigdata@bigdata:~$
```

8.4. Confirmar versión de Java

Hadoop requiere que la versión de Java 1.6 (o superior) esté instalada; no obstante, algunas herramientas requieren la versión 1.8, por lo que se comprobará esta última.

java -version

```
bigdata@bigdata:~$ java -version
openjdk version "1.8.0_131"
OpenJDK Runtime Environment (build 1.8.0_131-8u131-b11-0ubuntu1.17.04.1-b11)
OpenJDK 64-Bit Server VM (build 25.131-b11, mixed mode)
```

Una vez se confirme que la versión Java 1.8 se encuentra correctamente instalada, se busca la ruta en la que se encuentra instalado Java y se comprueba la variable de entorno JAVA_HOME que se requerirá en pasos posteriores.

update-alternatives --config java

```
bigdata@bigdata:~$ update-alternatives --config java
Sólo hay una alternativa en el grupo de enlaces java (provee /usr/bin/java): /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java
Nada que configurar.
```

Por lo tanto, el *path* completo es: */usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java*.

El valor de la variable JAVA_HOME es todo lo que se encuentra antes de */jre/bin/java*, que es por lo tanto */usr/lib/jvm/java-8-openjdk-amd64*.

Proceso: Opcional (en caso de no tenerlo instalado)

Se ejecutan los siguientes comandos:

```
sudo apt-get install openjdk-8-jre
```

```
bigdata@bigdata:~$ sudo apt-get install openjdk-8-jre
Leyendo lista de paquetes... Hecho
Creado árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  ca-certificates-java fonts-dejavu-extra java-common libatk-wrapper-java libatk-wrapper-java-jni libgif7 openjdk-8-jre-headless
Paquetes sugeridos:
  default-jre icedtea-8-plugin fonts-ipafont-gothic fonts-ipafont-mincho ttf-wqy-microhei | ttf-wqy-zenhei fonts-indic
Se instalarán los siguientes paquetes NUEVOS:
  ca-certificates-java fonts-dejavu-extra java-common libatk-wrapper-java libatk-wrapper-java-jni libgif7 openjdk-8-jre openjdk-8-jre-headless
0 actualizados, 8 nuevos se instalarán, 0 para eliminar y 112 no actualizados.
Se necesita descargar 29,8 MB de archivos.
Se utilizarán 110 MB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] S
```

Si se ha tenido que instalar Java, se debe ejecutar nuevamente el siguiente comando para verificar que está instalado correctamente y que la versión es, al menos, la 1.8.

```
java -version
```

Obtendremos una información similar a la siguiente:

```
bigdata@bigdata:~$ java -version
openjdk version "1.8.0_131"
OpenJDK Runtime Environment (build 1.8.0_131-8u131-b11-0ubuntu1.17.04.1-b11)
OpenJDK 64-Bit Server VM (build 25.131-b11, mixed mode)
```

Como no está correctamente instalada la versión requerida, se debe ejecutar el siguiente comando para seleccionar la versión correcta:

```
sudo update-alternatives --config java
```

```
bigdata@bigdata:~$ update-alternatives --config java
Sólo hay una alternativa en el grupo de enlaces java (provee /usr/bin/java): /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java
Nada que configurar.
```

8.5. Crear y configurar los certificados SSH

Hadoop usa SSH (para acceder a sus nodos y gestionar los demonios remotos), lo que requerirá un usuario y una contraseña. Sin embargo, se puede eliminar este requisito creando unos certificados SSH.

1

Antes es necesario comprobar la instalación de SSH y rsync con los siguientes comandos:

sudo apt-get install ssh

```
bigdata@bigdata:~$ sudo apt-get install ssh
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  ncurses-term openssh-server openssh-sftp-server ssh-import-id
Paquetes sugeridos:
  molly-guard monkeysphere rssh ssh-askpass
Se instalarán los siguientes paquetes NUEVOS:
  ncurses-term openssh-server openssh-sftp-server ssh ssh-import-id
0 actualizados, 5 nuevos se instalarán, 0 para eliminar y 112 no actualizados.
Se necesita descargar 633 kB de archivos.
Se utilizarán 5.245 kB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] S
Des:1 http://es.archive.ubuntu.com/ubuntu zesty/main amd64 openssh-sftp-server amd64 1:7.4p1-10 [39,9 kB]
Des:2 http://es.archive.ubuntu.com/ubuntu zesty/main amd64 openssh-server amd64 1:7.4p1-10 [334 kB]
```

sudo apt-get install rsync

```
bigdata@bigdata:~$ sudo apt-get install rsync
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
rsync ya está en su versión más reciente (3.1.2-1).
fijado rsync como instalado manualmente.
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 112 no actualizados.
```

2

Después se ejecutará el siguiente comando para crear el certificado:

```
ssh-keygen -t rsa -P "
```

En este punto se solicitará el nombre del fichero, pero se puede dejar en blanco:

```
bigdata@bigdata:~$ ssh-keygen -t rsa -P ''
Generating public/private rsa key pair.
Enter file in which to save the key (/home/bigdata/.ssh/id_rsa):
Created directory '/home/bigdata/.ssh'.
Your identification has been saved in /home/bigdata/.ssh/id_rsa.
Your public key has been saved in /home/bigdata/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:UtiDtzbALDFAzabOnlR4pqld8k6r6htLfNMWoEPy+oY bigdata@bigdata
The key's randomart image is:
+---[RSA 2048]---+
|oo=+o
| .=.. +
| ..+o = =
| o+.+ = o
| ooB .. S
| .Bo o ...
| =*.*.o
| E+*.+
| o*+oo
+---[SHA256]---+
bigdata@bigdata:~$
```

Se añade la nueva clave a la lista de claves autorizadas, de modo que Hadoop podrá usar SSH sin solicitar contraseña:

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

```
bigdata@bigdata:~$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

8.6. Modificar ficheros de configuración

Para terminar de configurar Hadoop, se van a modificar algunos de los ficheros de configuración de Hadoop para cambiar sus propiedades por defecto.

Estos ficheros se encuentran en la ruta: /home/bigdata/hadoop/etc/hadoop

```
bigdata@bigdata:~$ ls /home/bigdata/hadoop/etc/hadoop
capacity-scheduler.xml      httpfs-env.sh          mapred-env.sh
configuration.xsl           httpfs-log4j.properties  mapred-queues.xml.template
container-executor.cfg       httpfs-signature.secret mapred-site.xml.template
core-site.xml                httpfs-site.xml        slaves
hadoop-env.cmd              kms-acls.xml          ssl-client.xml.example
hadoop-env.sh                kms-env.sh            ssl-server.xml.example
hadoop-metrics2.properties   kms-log4j.properties  yarn-env.cmd
hadoop-metrics.properties    kms-site.xml          yarn-env.sh
hadoop-policy.xml            log4j.properties     yarn-site.xml
hdfs-site.xml
```

Además de modificar estos ficheros, se modificará el *bashrc* para incluir algunas rutas por defecto y otras variables de entorno:

- *~/.bashrc*: contiene las variables de entorno de la máquina Ubuntu.
- */home/bigdata/hadoop/etc/hadoop/core-site.xml*: contiene las propiedades de arranque de Hadoop.
- */home/bigdata/hadoop/etc/hadoop/hadoop-env.sh*: contiene algunas de las variables de entorno usadas por Hadoop.
- */home/bigdata/hadoop/etc/hadoop/hdfs-site.xml*: se usa para especificar los directorios que se usarán como NameNode y DataNode.
- */home/bigdata/hadoop/etc/hadoop/mapred-site.xml.template*: este archivo se usa para especificar el framework que se va a usar para MapReduce.
- */home/bigdata/hadoop/etc/hadoop/yarn-site.xml*: contiene las propiedades necesarias para el proceso de MapReduce.

8.6.1. *~/.bashrc*

Mediante el programa de edición que se prefiera, en este caso nano, se procederá a la edición del fichero *~/.bashrc*.

nano ~/.bashrc

```
bigdata@bigdata:~$ nano ~/.bashrc
```

Se añaden las siguientes líneas al final del fichero:

Para ver el código que necesitas, pulsa [aquí](#).

```

fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
    if [ -f /usr/share/bash-completion/bash_completion ]; then
        . /usr/share/bash-completion/bash_completion
    elif [ -f /etc/bash_completion ]; then
        . /etc/bash_completion
    fi
fi

#HADOOP VARIABLES START
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_HOME=/home/bigdata/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
#HADOOP VARIABLES END

```

Después de guardar (CTRL+O) y cerrar el fichero (CTRL+X), se ejecutará el siguiente comando para que el sistema reconozca las nuevas variables de entorno creadas, de modo que las variables estarán visibles cada vez que se reinicie la máquina virtual:

```
source ~/.bashrc
```

```
bigdata@bigdata:~$ source ~/.bashrc
```

8.6.2. core-site.xml

Este fichero contiene las propiedades que usa Hadoop en su arranque. Es necesario sobreescribir sus parámetros por defecto.

```
nano /home/bigdata/hadoop/etc/hadoop/core-site.xml
```

```
bigdata@bigdata:~$ sudo nano /home/bigdata/hadoop/etc/hadoop/core-site.xml
```

Para ello se añade un nuevo bloque de configuración que reemplace a <configuration></configuration>

```
GNU nano 2.7.4          Archivo: /home/bigdata/hadoop/etc/hadoop/core-site.xml

<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
</configuration>
```

Para ver el código que necesitas, pulsa [aquí](#).

```
GNU nano 2.7.4          Archivo: /home/bigdata/hadoop/etc/hadoop/core-site.xml

<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
  <name>fs.default.name</name>
  <value>hdfs://localhost:9000</value>
</property>
</configuration>
```

8.6.3. hadoop-env.sh

Contiene algunas de las variables de entorno usadas por Hadoop.

Se abre el fichero con el siguiente comando:

```
nano /home/bigdata/hadoop/etc/hadoop/hadoop-env.sh
```

```
bigdata@bigdata:~$ sudo nano /home/bigdata/hadoop/etc/hadoop/hadoop-env.sh
```

Y se reemplaza la ruta del *export* del JAVA_HOME por la siguiente:

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

```
GNU nano 2.7.4          Archivo: /home/bigdata/hadoop/etc/hadoop/hadoop-env.sh

# Licensed to the Apache Software Foundation (ASF) under one
# or more contributor license agreements. See the NOTICE file
# distributed with this work for additional information
# regarding copyright ownership. The ASF licenses this file
# to you under the Apache License, Version 2.0 (the
# "License"); you may not use this file except in compliance
# with the License. You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

# Set Hadoop-specific environment variables here.

# The only required environment variable is JAVA_HOME. All others are
# optional. When running a distributed configuration it is best to
# set JAVA_HOME in this file, so that it is correctly defined on
# remote nodes.

# The java implementation to use.
export JAVA_HOME=${JAVA_HOME}
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

8.6.4. hdfs-site.xml

Este fichero debe editarse en cada máquina del clúster. Se usa para especificar los directorios que se usarán como NameNode y DataNode.

Antes de editar este fichero, se deben crear dos directorios que contengan el NameNode y el DataNode.

```
mkdir -p /home/bigdata/hadoop_store/hdfs/namenode
```

```
mkdir -p /home/bigdata/hadoop_store/hdfs/datanode
```

```
bigdata@bigdata:~$ mkdir -p /home/bigdata/hadoop_store/hdfs/namenode
bigdata@bigdata:~$ mkdir -p /home/bigdata/hadoop_store/hdfs/datanode
```

Nota: es posible crear los directorios en otras rutas; en ese caso se debe modificar el fichero *hdfs-site.xml* de acuerdo con esas rutas.

Una vez creados los directorios, se edita el fichero:

```
nano /home/bigdata/hadoop/etc/hadoop/hdfs-site.xml
```

```
bigdata@bigdata:~$ sudo nano /home/bigdata/hadoop/etc/hadoop/hdfs-site.xml
```

Y se introducen las propiedades dentro de su configuración <configuration></configuration>

```
GNU nano 2.7.4          Archivo: /home/bigdata/hadoop/etc/hadoop/hdfs-site.xml

<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
 Licensed under the Apache License, Version 2.0 (the "License");
 you may not use this file except in compliance with the License.
 You may obtain a copy of the License at

 http://www.apache.org/licenses/LICENSE-2.0

 Unless required by applicable law or agreed to in writing, software
 distributed under the License is distributed on an "AS IS" BASIS,
 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 See the License for the specific language governing permissions and
 limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
</configuration>
```

Para ver el código que necesitas, pulsa [aquí](#).

```
GNU nano 2.7.4          Archivo: /home/bigdata/hadoop/etc/hadoop/hdfs-site.xml

<?xml version="1.0" encoding="UTF-8"?>
<x?ml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
  <name>dfs.replication</name>
  <value>1</value>
</property>
<property>
  <name>dfs.namenode.name.dir</name>
  <value>file:/home/bigdata/hadoop_store/hdfs/namenode</value>
</property>
<property>
  <name>dfs.datanode.data.dir</name>
  <value>file:/home/bigdata/hadoop_store/hdfs/datanode</value>
</property>
</configuration>
```

8.6.5. mapred-site.xml

Este archivo se usa para especificar el *framework* que se va a usar para MapReduce.

Por defecto, la carpeta `/home/bigdata/hadoop/etc/hadoop/` contiene la plantilla `/home/bigdata/hadoop/etc/hadoop/mapred-site.xml.template` que debe copiarse o renombrarse a `mapred-site.xml`.

Esta modificación se puede hacer con el siguiente comando:

```
cp /home/bigdata/hadoop/etc/hadoop/mapred-site.xml.template
/home/bigdata/hadoop/etc/hadoop/mapred-site.xml
```

```
bigdata@bigdata:~$ sudo cp /home/bigdata/hadoop/etc/hadoop/mapred-site.xml.template /home/bigdata/hadoop/etc/hadoop/mapred-site.xml
bigdata@bigdata:~$
```

Una vez ejecutado, hay que abrir el nuevo fichero:

```
nano /home/bigdata/hadoop/etc/hadoop/mapred-site.xml
```

```
bigdata@bigdata:~$ sudo nano /home/bigdata/hadoop/etc/hadoop/mapred-site.xml
```

Y añadir al bloque de configuración <configuration></configuration> lo siguiente:

```
GNU nano 2.7.4          Archivo: /home/bigdata/hadoop/etc/hadoop/mapred-site.xml

?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
</configuration>
```

Para ver el código que necesitas, pulsa [aquí](#).

```
GNU nano 2.7.4          Archivo: /home/bigdata/hadoop/etc/hadoop/mapred-site.xml

<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
</property>
</configuration>
```

8.6.6. yarn-site.xml

Este fichero contiene las propiedades necesarias para el proceso MapReduce.

nano /home/bigdata/hadoop/etc/hadoop/yarn-site.xml

bigdata@bigdata:~\$ sudo nano /home/bigdata/hadoop/etc/hadoop/yarn-site.xml

Se añade el nuevo bloque de propiedades de configuración que reemplace a *<configuration></configuration>*

```
GNU nano 2.7.4          Archivo: /home/bigdata/hadoop/etc/hadoop/yarn-site.xml

<?xml version="1.0"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->
<configuration>

<!-- Site specific YARN configuration properties -->

</configuration>
```

Para ver el código que necesitas, pulsa [aquí](#).

```
-->
<configuration>
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>
<property>
  <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
  <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
</configuration>
```

8.7. Formatear el sistema de archivos de Hadoop

Tras haber modificado los ficheros de configuración anteriores, hay que formatear el sistema de archivos de Hadoop antes de poder usarlo. Se puede hacer ejecutando el siguiente comando:

```
hdfs namenode -format
```

```
bigdata@bigdata:~/hadoop$ bin/hdfs namenode -format
17/06/16 12:07:57 INFO namenode.NameNode: STARTUP_MSG:
*****STARTUP_MSG: Starting NameNode
STARTUP_MSG:   user = bigdata
STARTUP_MSG:   host = bigdata/127.0.0.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.8.0
STARTUP_MSG:   classpath = /home/bigdata/hadoop/etc/hadoop:/home/bigdata/hadoop/share/hadoop/common/lib/junit-4
11.jar:/home/bigdata/hadoop/share/hadoop/common/lib/jaxb-api-2.2.2.jar:/home/bigdata/hadoop/share/hadoop/common
lib/htrace-core4-4.0.1-incubating.jar:/home/bigdata/hadoop/share/hadoop/common/lib/jsp-api-2.1.jar:/home/bigdat
/hadoop/share/hadoop/common/lib/jcip-annotations-1.0.jar:/home/bigdata/hadoop/share/hadoop/common/lib/jackson-j
xrs-1.9.13.jar:/home/bigdata/hadoop/share/hadoop/common/lib/stax-api-1.0-2.jar:/home/bigdata/hadoop/share/hadoo
/common/lib/hadoop-auth-2.8.0.jar:/home/bigdata/hadoop/share/hadoop/common/lib/java-xmlbuilder-0.4.jar:/home/bi
data/hadoop/share/hadoop/common/lib/netty-3.6.2.Final.jar:/home/bigdata/hadoop/share/hadoop/common/lib/jettison
1.1.jar:/home/bigdata/hadoop/share/hadoop/common/lib/jersey-server-1.9.jar:/home/bigdata/hadoop/share/hadoop/co
mon/lib/commons-io-2.4.jar:/home/bigdata/hadoop/share/hadoop/common/lib/snappy-java-1.0.4.1.jar:/home/bigdata/h
adoop/share/hadoop/common/lib/slf4j-api-1.7.10.jar:/home/bigdata/hadoop/share/hadoop/common/lib/jets3t-0.9.0.jar
/home/bigdata/hadoop/share/hadoop/common/lib/apacheds-i18n-2.0.0-M15.jar:/home/bigdata/hadoop/share/hadoop/com
mon/lib/commons-beanutils-1.7.0.jar:/home/bigdata/hadoop/share/hadoop/common/lib/curator-client-2.7.1.jar:/home/b
igdata/hadoop/share/hadoop/common/lib/paranamer-2.3.jar:/home/bigdata/hadoop/share/hadoop/common/lib/curator-rec
pes-2.7.1.jar:/home/bigdata/hadoop/share/hadoop/common/lib/apacheds-kerberos-codec-2.0.0-M15.jar:/home/bigdata/h
adoop/share/hadoop/common/lib/commons-configuration-1.6.jar:/home/bigdata/hadoop/share/hadoop/common/lib/hamcre
```



IMPORTANTE: esto se debe hacer antes de reiniciar Hadoop. Si este comando se ejecuta tras haber usado Hadoop, **destruirá todo el sistema de archivos y la información almacenada en ellos.**

8.8. Demonios

Los distintos módulos de Hadoop se controlan mediante una serie de “demonios”. Según el tamaño del clúster con el que se esté trabajando, estos demonios podrán estar distribuidos en diferentes máquinas para mejorar el rendimiento del sistema. En este caso vamos a trabajar con un único servidor donde se encontrarán todos los demonios, entre los que se reparten los recursos del sistema.

A continuación se describen los distintos demonios y procesos que arrancan, así como su funcionamiento. Lo explicaremos mediante un ejemplo de un clúster para entender mejor el funcionamiento de cada uno de los ficheros.

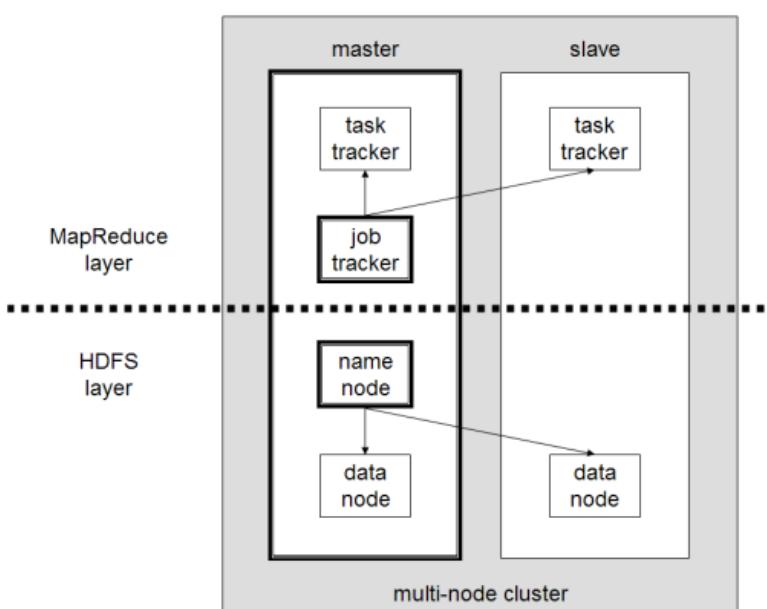


Figura 15. Demonios del clúster.
Fuente: Fundación Apache.

Capa HDFS
Contiene los nodos empleados para el almacenamiento de la información.
<ul style="list-style-type: none"> → NameNode: nodo maestro. → DataNode: nodo secundario. Habrá uno por cada servidor del clúster. → Secondary DataNode: nodo de recuperación empleado para las posibles caídas del sistema.
Capa MapReduce 2 o YARN
Contiene los nodos encargados de la distribución de tareas.
<ul style="list-style-type: none"> → JobTracker: maestro. → TaskTracker: secundario.

Básicamente, los demonios del nodo maestro son los responsables de la coordinación y la gestión de los nodos esclavos mientras que estos últimos almacenan los datos y procesan el trabajo.

8.8.1. Arrancar demonios HDFS

Para poder arrancar los demonios de la capa HDFS se ejecuta el comando:

```
./sbin/start-dfs.sh
```

```
bigdata@bigdata:~/hadoop$ ./sbin/start-dfs.sh
Starting namenodes on [localhost]
localhost: starting namenode, logging to /home/bigdata/hadoop/logs/hadoop-bigdata-namenode-bigdata.out
localhost: starting datanode, logging to /home/bigdata/hadoop/logs/hadoop-bigdata-datanode-bigdata.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/bigdata/hadoop/logs/hadoop-bigdata-secondarynamenode-bigdata.out
```

Como resultado se arrancarán los distintos nodos (NameNode, DataNode y SecondaryNameNode), como se puede comprobar ejecutando el comando:

```
jps
```

```
bigdata@bigdata:~/hadoop$ jps
4289 Jps
3841 NameNode
4178 SecondaryNameNode
3993 DataNode
bigdata@bigdata:~/hadoop$
```



Nota 1: Si al probar el comando jps este no funcionara, es preciso realizar lo siguiente:

```
sudo apt install openjdk-8-jdk-headless
```

```
update-alternatives --config java
```

Nota 2: Si tras arrancar los demonios de hdfs (./sbin/start-dfs.sh) vemos que el datanode no aparece, es preciso realizar lo siguiente:

Parar todos los demonios, hay que borrar los directorios namenode y datanode en: /home/bigdata/hadoop_store/hdfs/namenode, y crearlos de nuevo con:

```
mkdir -p /home/bigdata/hadoop_store/hdfs/namenode
```

```
mkdir -p /home/bigdata/hadoop_store/hdfs/datanode
```

Posteriormente, se debe formatear el sistema de archivos hdfs con:

```
hdfs namenode -format
```

Finalmente, es necesario arrancar de nuevo los demonios de hdfs y yarn.

8.8.2. Arrancar demonios YARN

Este conjunto de demonios es el encargado de controlar la distribución de tareas en MapReduce. Para arrancarlos, se ejecuta el siguiente comando:

```
./sbin/start-yarn.sh
```

```
bigdata@bigdata:~/hadoop$ ./sbin/start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /home/bigdata/hadoop/logs/yarn-bigdata-resourcemanager-bigdata.out
localhost: starting nodemanager, logging to /home/bigdata/hadoop/logs/yarn-bigdata-nodemanager-bigdata.out
bigdata@bigdata:~/hadoop$
```

Como resultado, se habrán arrancado los siguientes demonios:

```
jps
```

Al ejecutar el comando anterior, se mostrará algo similar a lo siguiente con los ids de cada uno de los procesos que se están ejecutando:

```
bigdata@bigdata:~/hadoop$ jps
10001 ResourceManager
9506 NameNode
9848 SecondaryNameNode
10121 NodeManager
10459 Jps
9660 DataNode
bigdata@bigdata:~/hadoop$
```

8.8.3. Parar demonios HDFS

Para parar los procesos, podemos ejecutar el comando:

```
./sbin/stop-dfs.sh
```

Como resultado se pararán los demonios asociados a HDFS configurados en el nodo del clúster en el que nos encontramos.

```
bigdata@bigdata:~/hadoop$ ./sbin/stop-dfs.sh
Stopping namenodes on [localhost]
localhost: stopping namenode
localhost: stopping datanode
Stopping secondary namenodes [0.0.0.0]
0.0.0.0: stopping secondarynamenode
bigdata@bigdata:~/hadoop$
```

8.8.4. Parar demonios YARN

Para parar los procesos, podemos ejecutar el comando:

```
./sbin/stop-yarn.sh
```

Como resultado se pararán los demonios asociados a MapReduce configurados en el nodo del clúster en el que nos encontramos.

```
bigdata@bigdata:~/hadoop$ ./sbin/stop-yarn.sh
stopping yarn daemons
stopping resourcemanager
localhost: stopping nodemanager
localhost: nodemanager did not stop gracefully after 5 seconds: killing with kill -9
no proxyserver to stop
bigdata@bigdata:~/hadoop$
```

8.8.5. Arrancar demonios individualmente

En `/home/bigdata/hadoop/sbin` se localizan todos los *scripts* de arranque de cualquier demonio de Hadoop.

```
bigdata@bigdata:~/hadoop/sbin$ ls /home/bigdata/hadoop/sbin/
distribute-exclude.sh  kms.sh          start-balancer.sh   stop-all.cmd    stop-yarn.cmd
hadoop-daemon.sh      mr-jobhistory-daemon.sh  start-dfs.cmd    stop-all.sh    stop-yarn.sh
hadoop-daemons.sh     refresh-namenodes.sh  start-dfs.sh     stop-balancer.sh  yarn-daemon.sh
hdfs-config.cmd       slaves.sh        start-secure-dns.sh stop-dfs.cmd    yarn-daemons.sh
hdfs-config.sh        start-all.cmd   start-yarn.cmd   stop-dfs.sh    stop-secure-dns.sh
httpfs.sh             start-all.sh   start-yarn.sh    stop-yarn.sh
bigdata@bigdata:~/hadoop/sbin$
```

A continuación, se muestran algunos ejemplos de arranque de esos demonios, algunos de los cuales se pueden arrancar con los *scripts* vistos anteriormente o de forma individual. Es muy importante tenerlo en cuenta para el arranque individual de dichos demonios en un clúster con múltiples máquinas dedicadas.

→ **NameNode**

`sbin/hadoop-daemon.sh start namenode`

```
bigdata@bigdata:~/hadoop$ sbin/hadoop-daemon.sh start namenode
starting namenode, logging to /home/bigdata/hadoop/logs/hadoop-bigdata-namenode-bigdata.out
```

`jps`

```
bigdata@bigdata:~/hadoop$ jps
11140 Jps
11065 NameNode
bigdata@bigdata:~/hadoop$
```

→ **DataNode**

`sbin/hadoop-daemon.sh start datanode`

```
bigdata@bigdata:~/hadoop$ sbin/hadoop-daemon.sh start datanode
starting datanode, logging to /home/bigdata/hadoop/logs/hadoop-bigdata-datanode-bigdata.out
```

`jps`

```
bigdata@bigdata:~/hadoop$ jps
11251 Jps
11172 DataNode
11065 NameNode
bigdata@bigdata:~/hadoop$
```

Para arrancar los demonios de MapReduce se ejecuta lo siguiente, en función del demonio que se quiera lanzar:

→ **Resource Manager**

`sbin/yarn-daemon.sh start resourcemanager`

```
bigdata@bigdata:~/hadoop$ sbin/yarn-daemon.sh start resourcemanager
starting resourcemanager, logging to /home/bigdata/hadoop/logs/yarn-bigdata-resourcemanager-bigdata.out
```

jps

```
bigdata@bigdata:~/hadoop$ jps
11172 DataNode
11286 ResourceManager
11512 Jps
11065 NameNode
```

→ **Node Manager**

sbin/yarn-daemon.sh start nodemanager

```
bigdata@bigdata:~/hadoop$ sbin/yarn-daemon.sh start nodemanager
starting nodemanager, logging to /home/bigdata/hadoop/logs/yarn-bigdata-nodemanager-bigdata.out
```

jps

```
bigdata@bigdata:~/hadoop$ jps
11680 Jps
11172 DataNode
11286 ResourceManager
11065 NameNode
11547 NodeManager
```

→ **Job History Server**

sbin/mr-jobhistory-daemon.sh start historyserver

```
bigdata@bigdata:~/hadoop$ sbin/mr-jobhistory-daemon.sh start historyserver
starting historyserver, logging to /home/bigdata/hadoop/logs/mapred-bigdata-historyserver-bigdata.out
```

jps

```
bigdata@bigdata:~/hadoop$ jps
11172 DataNode
11717 JobHistoryServer
11286 ResourceManager
11753 Jps
11065 NameNode
11547 NodeManager
```

8.8.6. Interfaces web de Hadoop

Hadoop facilita varias interfaces web para consultar el estado de sus demonios:

- <http://localhost:50070/> – NameNode y DataNode
- <http://localhost:8088/> - ResourceManager
- <http://localhost:19888/> - MapReduce JobHistory

NameNode Web Interface (HDFS layer)

El interfaz de usuario del NameNode muestra un resumen del clúster 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. Por último, permite el acceso a los *logs* de Hadoop.

Por defecto, está disponible en <http://localhost:50070/>

The screenshot shows a browser window titled 'Namenode information'. The address bar shows the URL 'localhost:50070/dfshealth.html#tab-overview'. The page has a green header bar with tabs: 'Hadoop' (selected), 'Overview', 'Datanodes', 'Datanode Volume Failures', 'Snapshot', 'Startup Progress', and 'Utilities'. Below the header is a section titled 'Overview 'localhost:9000' (active)'. Underneath this, there is a table with the following data:

Started:	Fri Jun 16 13:22:03 +0200 2017
Version:	2.8.0, r91f2b7a13d1e97be65db92ddabc627cc29ac0009
Compiled:	Fri Mar 17 05:12:00 +0100 2017 by jdu from branch-2.8.0
Cluster ID:	CID-9bb18c9b-06e9-4d28-b54f-dc7e66751efa
Block Pool ID:	BP-1285781601-127.0.1.1-1497611666923

Overview 'localhost:9000' (active)

Started:	Fri Jun 16 13:22:03 +0200 2017
Version:	2.8.0, r91f2b7a13d1e97be65db92ddabc627cc29ac0009
Compiled:	Fri Mar 17 05:12:00 +0100 2017 by jdu from branch-2.8.0
Cluster ID:	CID-9bb18c9b-06e9-4d28-b54f-dc7e66751efa
Block Pool ID:	BP-1285781601-127.0.1.1-1497611666923

Summary

Security is off.

Safemode is off.

7 files and directories, 0 blocks = 7 total filesystem object(s).

Heap Memory used 36.32 MB of 60.06 MB Heap Memory. Max Heap Memory is 966.69 MB.

Non Heap Memory used 38.77 MB of 39.78 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	17.59 GB
DFS Used:	24 KB (0%)
Non DFS Used:	8.62 GB
DFS Remaining:	8.05 GB (45.78%)
Block Pool Used:	24 KB (0%)

The screenshot shows a browser window titled "Browsing HDFS" with the URL "localhost:50070/explorer.html#/. The page has a green header bar with links for Hadoop, Overview, Datanodes, Datanode Volume Failures, Snapshot, Startup Progress, and Utilities. Below the header is a table showing a single entry: "/tmp" owned by "bigdata" and group "supergroup" with size 0 B, last modified on Jun 16 13:24, replication factor 0, block size 0 B, and name tmp. There are buttons for Go!, Search, and a folder icon.

Browse Directory

The screenshot shows a browser window titled "Browse Directory" with the URL "localhost:50070/explorer.html#/. The page has a green header bar with links for Hadoop, Overview, Datanodes, Datanode Volume Failures, Snapshot, Startup Progress, and Utilities. Below the header is a table showing a single entry: "/tmp" owned by "bigdata" and group "supergroup" with size 0 B, last modified on Jun 16 13:24, replication factor 0, block size 0 B, and name tmp. There are buttons for Go!, Search, and a folder icon.

Hadoop, 2017.

Resource Manager Web Interface (YARN layer)

Permite consultar las estadísticas y métricas de los recursos usados en el clúster, los *jobs* que se están ejecutando y los que hay programados.

Por defecto, está disponible en <http://localhost:8088/>

The screenshot shows a browser window titled "All Applications" with the URL "localhost:8088/cluster". The page has a header with the Hadoop logo and the title "All Applications". On the left is a sidebar with "Cluster Metrics" (Apps Submitted: 0, Apps Pending: 0, Apps Running: 0, Apps Completed: 0, Containers Running: 0, Memory Used: 0 B, Memory Total: 8 GB, Memory Reserved: 0 B, Vcores Used: 0, Vcores Total: 0, Vcores Reserved: 0), "Cluster Nodes Metrics" (Active Nodes: 1, Decommissioning Nodes: 0, Decommissioned Nodes: 0, Lost Nodes: 0, Unhealthy Nodes: 0, Rebooted Nodes: 0, Shutdown Nodes: 0), and "Scheduler Metrics" (Scheduler Type: Capacity Scheduler, Scheduling Resource Type: [MEMORY], Minimum Allocation: <memory:1024, vCores:1>, Maximum Allocation: <memory:8192, vCores:4>, Maximum Cluster Application Priority: 0). The main area shows a table for "All Applications" with columns: ID, User, Name, Application Type, Queue, Application Priority, StartTime, EndTime, State, FinalStatus, Running Containers, Allocated CPU Vcores, Allocated Memory MB, % of Queue, % of Cluster, Progress, Tracking UI, and Blacklisted Nodes. The table is currently empty. There are buttons for Show 20 entries, Search, and First, Previous, Next, Last.

MapReduce JobHistory

Muestra información y detalles de trabajos MapReduce ejecutados.

Por defecto, está disponible en <http://localhost:19888/>

The screenshot shows a web browser window titled "JobHistory" with the URL "localhost:19888/jobhistory". The page features a "hadoop" logo at the top left. A sidebar on the left has "Application" and "About Jobs" sections, and a "Tools" section which is currently selected. The main content area is titled "Retired Jobs" and displays a table header 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. Below the header, there is a search bar and a message: "No data available in table". At the bottom of the table area, it says "Showing 0 to 0 of 0 entries".

IX. Ejemplo de comprobación de la instalación

En este epígrafe, se propone un ejercicio que comprobará la correcta instalación del sistema. El ejercicio consistirá en:

- Creación de un directorio.
- Creación de un archivo de ejemplo.
- Inclusión del directorio creado en el sistema de archivos HDFS.
- Ejecución de un trabajo que cuente el número de repeticiones de cada palabra existente en el fichero.

A continuación, se detalla el ejemplo de *wordcount* para verificar la instalación.

1

Primero, se creará el árbol de directorios donde se ubicará el archivo:

```
sudo mkdir /home/bigdata/hadoop/ejercicios
```

```
sudo mkdir /home/bigdata/hadoop/ejercicios/ejercicio1
```

```
sudo mkdir /home/bigdata/hadoop/ejercicios/ejercicio1/in
```

```
sudo chmod -R 777 /home/bigdata/hadoop/ejercicios
```

```
bigdata@bigdata:~/hadoop$ sudo mkdir /home/bigdata/hadoop/ejercicios
bigdata@bigdata:~/hadoop$ sudo mkdir /home/bigdata/hadoop/ejercicios/ejercicio1
bigdata@bigdata:~/hadoop$ sudo mkdir /home/bigdata/hadoop/ejercicios/ejercicio1/in
bigdata@bigdata:~/hadoop$ sudo chmod -R 777 /home/bigdata/hadoop/ejercicios
```

2

Se crea un archivo llamado "ejemplo.txt" y se escriben dos líneas de ejemplo.

```
cat >/home/bigdata/hadoop/ejercicios/ejercicio1/in/ejemplo.txt
```

Esta es una línea

Esta es otra línea de ejemplo

CTRL+C finalizará la escritura.

```
bigdata@bigdata:~/hadoop$ cat > /home/bigdata/hadoop/ejercicios/ejercicio1/in/ejemplo.txt
Esta es una linea
Esta es otra linea de ejemplo
^C
bigdata@bigdata:~/hadoop$
```

3

Como resultado se habrá creado un fichero con esa información.

```
cd ejercicios/ejercicio1/
```

```
ls
```

```
cd in/
```

```
ls
```

```
bigdata@bigdata:~/hadoop$ cd ejercicios/ejercicio1/
bigdata@bigdata:~/hadoop/ejercicios/ejercicio1$ ls
in
bigdata@bigdata:~/hadoop/ejercicios/ejercicio1$ cd in/
bigdata@bigdata:~/hadoop/ejercicios/ejercicio1/in$ ls
ejemplo.txt
bigdata@bigdata:~/hadoop/ejercicios/ejercicio1/in$
```

4

Una vez creado el fichero, se añade a HDFS:

```
bin/hadoop dfs -copyFromLocal /home/bigdata/hadoop/ejercicios/ejercicio1/in /in
```

```
bigdata@bigdata:~/hadoop$ bin/hadoop dfs -copyFromLocal /home/bigdata/hadoop/ejercicios/ejercicio1/in /in
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
```

En la nueva versión de Hadoop, el comando que se debe ejecutar es:

```
hdfs dfs -copyFromLocal /home/bigdata/hadoop/ejercicios/ejercicio1/in /in
```

```
bigdata@bigdata:~/hadoop$ bin/hdfs dfs -copyFromLocal /home/bigdata/hadoop/ejercicios/ejercicio1/in /in
bigdata@bigdata:~/hadoop$
```

5

Con el siguiente comando se comprobará si se ha añadido el directorio a HDFS de manera correcta:

```
hdfs dfs -ls /
```

```
bigdata@bigdata:~/hadoop$ hdfs dfs -ls /
Found 2 items
drwxr-xr-x  - bigdata supergroup          0 2017-08-11 21:11 /in
drwxr-xr-x  - bigdata supergroup          0 2017-08-11 21:12 /tmp
bigdata@bigdata:~/hadoop$
```

Para eliminar un directorio se emplea la instrucción:

```
hdfs dfs -rm -r hdfs:/in
```

```
bigdata@bigdata:~/hadoop$ hdfs dfs -rm -r hdfs:/in
Deleted hdfs:///in
bigdata@bigdata:~/hadoop$
```

6

Se vuelven a copiar los ficheros, pero esta vez en una ruta del directorio ejercicio1:

```
cd /home/bigdata/hadoop/ejercicios/ejercicio1
```

```
hdfs dfs -copyFromLocal in hdfs:/ejercicio1
```

```
hdfs dfs -ls /
```

```
bigdata@bigdata:~/hadoop$ cd /home/bigdata/hadoop/ejercicios/ejercicio1
bigdata@bigdata:~/hadoop/ejercicios/ejercicio1$ hdfs dfs -copyFromLocal in hdfs:/ejercicio1
bigdata@bigdata:~/hadoop/ejercicios/ejercicio1$ hdfs dfs -ls /
Found 2 items
drwxr-xr-x  - bigdata supergroup          0 2017-08-11 21:16 /ejercicio1
drwxr-xr-x  - bigdata supergroup          0 2017-08-11 21:12 /tmp
bigdata@bigdata:~/hadoop/ejercicios/ejercicio1$ █
```

Ejecución del comando wordcount:

```
hadoop jar /home/bigdata/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.8.0.jar
wordcount /ejercicio1 /out
```

```
bigdata@bigdata:~/hadoop$ hadoop jar /home/bigdata/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.8.0.jar wordcount /ejercicio1 /out
17/06/16 13:45:09 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
17/06/16 13:45:10 INFO input.FileInputFormat: Total input files to process : 1
17/06/16 13:45:10 INFO mapreduce.JobSubmitter: number of splits:1
17/06/16 13:45:11 INFO mapreduce.Job: Submitting tokens for job: job_1497612245395_0001
17/06/16 13:45:11 INFO impl.YarnClientImpl: Submitted application application_1497612245395_0001
17/06/16 13:45:11 INFO mapreduce.Job: The url to track the job: http://bigdata:8088/proxy/application_1497612245395_0001/
17/06/16 13:45:11 INFO mapreduce.Job: Running job: job_1497612245395_0001
17/06/16 13:45:23 INFO mapreduce.Job: Job job_1497612245395_0001 running in uber mode : false
17/06/16 13:45:29 INFO mapreduce.Job: map 0% reduce 0%
17/06/16 13:45:36 INFO mapreduce.Job: map 100% reduce 0%
17/06/16 13:45:36 INFO mapreduce.Job: map 100% reduce 100%
17/06/16 13:45:36 INFO mapreduce.Job: Job job_1497612245395_0001 completed successfully
17/06/16 13:45:36 INFO mapreduce.Job: Counters: 49
  File System Counters
    FILE: Number of bytes read=82
    FILE: Number of bytes written=273095
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=157
    HDFS: Number of bytes written=48
    HDFS: Number of read operations=6
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=2
  ...
  Job Counters
    mapred.job.end-to-end.ctime=2
    Launched map tasks=1
    Launched reduce tasks=1
    Data-local map tasks=1
    Total time spent by all maps in occupied slots (ms)=3874
    Total time spent by all reduces in occupied slots (ms)=3980
    Total time spent by all map tasks (ms)=3874
    Total time spent by all reduce tasks (ms)=3980
    Total vcore-milliseconds taken by all map tasks=3874
    Total vcore-milliseconds taken by all reduce tasks=3980
    Total megabyte-milliseconds taken by all map tasks=3966976
    Total megabyte-milliseconds taken by all reduce tasks=4075520
  Map-Reduce Framework
    Map input records=2
    Map output records=10
    Map output bytes=88
    Map output materialized bytes=82
    Input split bytes=109
    Combine input records=10
    Combine output records=7
    Reduce input groups=7
    Reduce shuffle bytes=82
    Reduce input records=7
    Reduce output records=7
    Spilled Records=14
    Shuffled Maps =1
    Failed Shuffles=0
    Merged Map outputs=1
    GC time elapsed (ms)=106
    CPU time spent (ms)=780
    Physical memory (bytes) snapshot=343625728
    Virtual memory (bytes) snapshot=3786981376
    Total committed heap usage (bytes)=222429184
  Shuffle Errors
    BAD_ID=0
    CONNECTION=0
    IO_ERROR=0
    WRONG_LENGTH=0
    WRONG_MAP=0
    WRONG_REDUCE=0
  File Input Format Counters
    Bytes Read=48
  File Output Format Counters
    Bytes Written=48
bigdata@bigdata:~/hadoop$
```

Revisión de la salida:

`bin/hdfs dfs -cat /out/*`

```
bigdata@bigdata:~/hadoop$ bin/hdfs dfs -cat /out/*
Esta      2
de        1
ejemplo   1
es        2
linea     2
otra      1
una      1
bigdata@bigdata:~/hadoop$
```

X. HDFS

HDFS es un sistema de archivos diseñado para grandes sistemas de datos distribuidos sobre frameworks como MapReduce. HDFS abstracta al usuario del modo en el que los datos están disponibles y distribuidos sobre múltiples máquinas, de modo que se tiene la ilusión de estar trabajando con un único archivo.



Anotación: Características

- HDFS almacena bloques de menos de 64 MB de tamaño, frente a los 4-32 KB de la mayoría de sistemas de archivos.
- HDFS está optimizado para aportar rendimiento sobre latencia, es decir, es muy eficiente en la transmisión y lectura de las peticiones de archivos de gran tamaño, pero no en la búsqueda de muchas peticiones pequeñas.
- HDFS está optimizado para las cargas de trabajo que generalmente son de una escritura y múltiples lecturas.
- 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.
- En lugar de controlar fallos de disco empleando técnicas de redundancias físicas en las matrices de discos o estrategias similares, HDFS utiliza la replicación. Cada uno de los bloques que comprenden un archivo se almacena en varios nodos del clúster, y el HDFS NameNode monitoriza constantemente cada DataNode para asegurarse de que se sigue manteniendo el factor de replicación deseado. Si esto no sucede, programa la copia de un bloque dentro del clúster.

10.1. Arquitectura HDFS

El sistema de archivos distribuidos Hadoop (HDFS) es un sistema diseñado para ejecutarse en hardware de bajo coste. Tiene muchas similitudes con los sistemas de archivos distribuidos pero también presenta diferencias significativas. Como se ha visto anteriormente, HDFS es altamente tolerante a fallos, proporcionando un alto rendimiento de acceso a datos de la aplicación, lo que resulta adecuado para aplicaciones que manejan grandes conjuntos de datos. HDFS fue construido originalmente como infraestructura para el proyecto de motor de búsqueda en la web de Apache Nutch y es parte del proyecto Apache Hadoop Core.

HDFS tiene una arquitectura maestro/esclavo. Un clúster HDFS consta de un solo NameNode que es un servidor maestro que gestiona el espacio de nombres del sistema de archivos y regula el acceso a los archivos de los clientes. Además, hay una serie de DataNodes, por lo general uno en cada nodo del clúster, que gestionan el almacenamiento asociado a cada uno de los nodos que se ejecutan en el clúster. HDFS proporciona un espacio de nombres del sistema de archivos y permite que los datos de usuario se almacenen en ficheros. Internamente, un archivo se divide en uno o más bloques y estos bloques se almacenan en un conjunto de DataNodes. El NameNode ejecuta operaciones del espacio de nombres del sistema de archivos como abrir, cerrar y renombrar archivos y directorios. También determina la asignación de bloques para los DataNodes. Los DataNodes son responsables de atender las solicitudes de lectura y escritura de los clientes del sistema de archivos. Los DataNodes también se encargan de la creación de bloques, eliminación y replicación según instrucciones del NameNode.

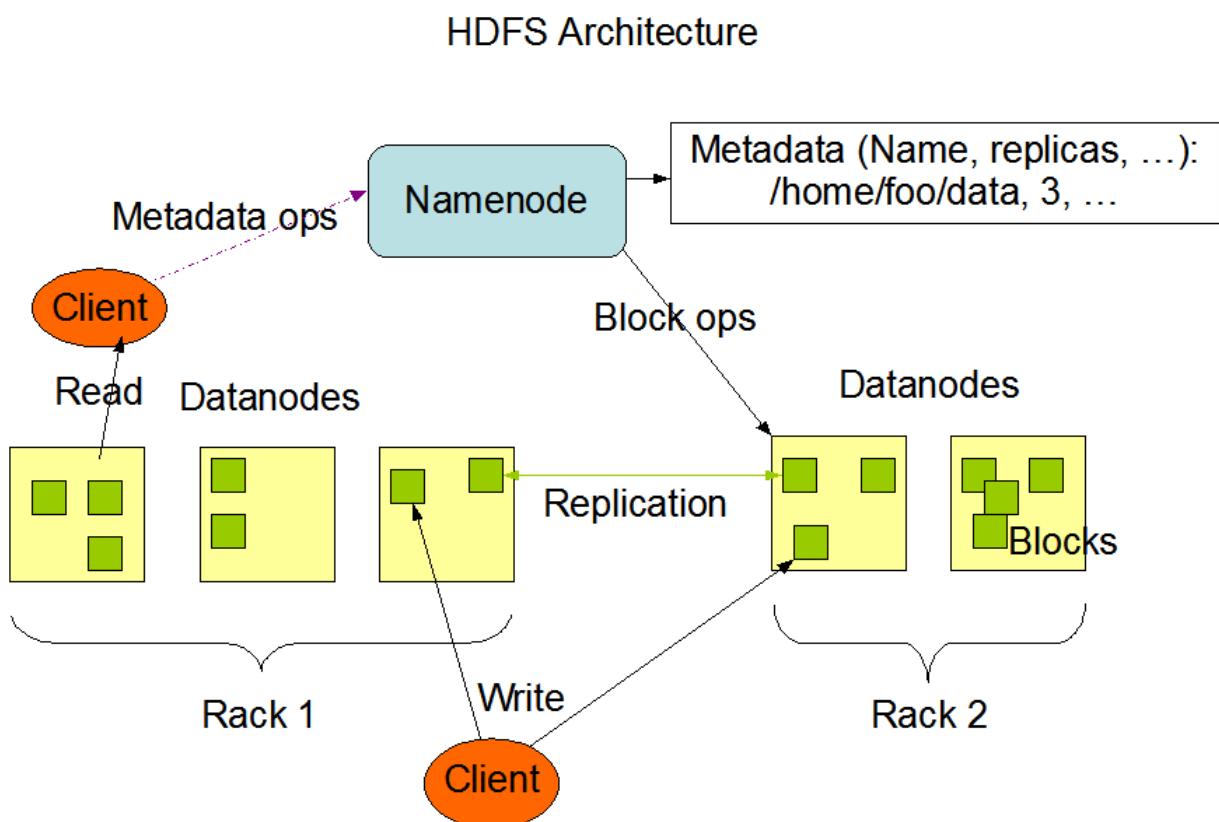


Figura 16. HDFS Architecture. *Fuente:* Fundación Apache.

El NameNode y el DataNode son piezas de *software* diseñadas para ejecutarse en máquinas de recursos básicos. Estas máquinas suelen ejecutar un sistema operativo GNU/Linux. HDFS se construye utilizando el lenguaje Java de modo que cualquier máquina que soporte Java puede ejecutar el *software* del NameNode o del DataNode. Un despliegue típico consiste en un equipo dedicado en el que solo se ejecuta el *software* NameNode y cada una de las otras máquinas del clúster ejecuta una instancia del *software* DataNode. La arquitectura no impide la ejecución de varios DataNodes en la misma máquina, pero en una implementación real no suele emplearse.

La existencia de un único NameNode en un clúster simplifica en gran medida la arquitectura del sistema. El NameNode es el árbitro y repositorio para todos los metadatos HDFS. El sistema está diseñado de tal manera que los datos de usuario nunca fluyan a través del NameNode.

10.2. Replicación de datos

HDFS está diseñado para almacenar de forma fiable archivos muy grandes a través de las máquinas en un clúster. Almacena cada archivo como una secuencia de bloques, todos del mismo tamaño excepto el último. Los bloques de un archivo se replican para obtener tolerancia frente a fallos. El tamaño del bloque y el factor de replicación son configurables por archivo. También es configurable el número de réplicas de un archivo. El factor de replicación se puede especificar en el momento de creación del archivo o se puede cambiar más adelante. Los archivos en HDFS son de una sola escritura y tienen estrictamente un escritor en cualquier momento.

Block Replication

```
Namenode (Filename, numReplicas, block-ids, ...)
/users/sameerp/data/part-0, r:2, {1,3}, ...
/users/sameerp/data/part-1, r:3, {2,4,5}, ...
```

Datanodes

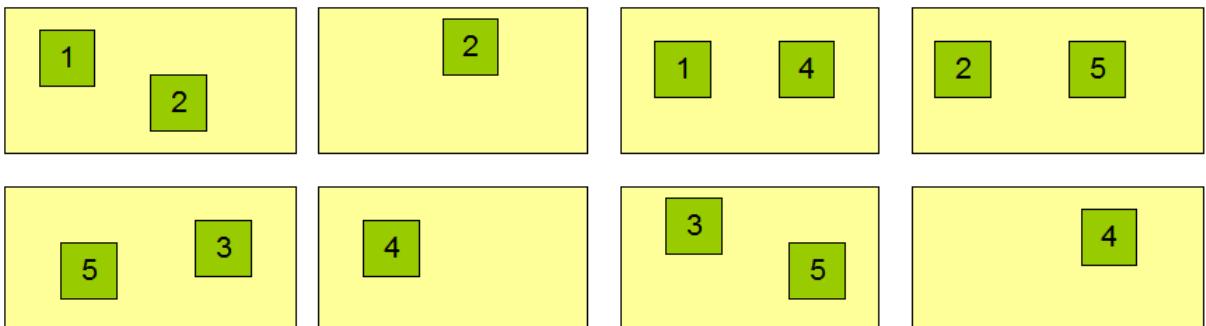


Figura 17. Replicación de bloques. *Fuente:* Fundación Apache.

El NameNode toma todas las decisiones con respecto a la replicación de bloques. Se recibe periódicamente un mensaje de confirmación de actividad y un informe de los bloques de cada uno de los DataNodes del clúster. La recepción del mensaje de confirmación implica que el DataNode está funcionando correctamente. Cada informe sobre los bloques contiene una lista de todos los bloques en un DataNode.

La copia de réplicas es fundamental para la fiabilidad y el rendimiento HDFS. La optimización de la copia de las réplicas es una característica que distingue HDFS de la mayoría de otros sistemas de archivos distribuidos. El propósito de la política de copia de réplica en rack es mejorar la fiabilidad de los datos, la disponibilidad y la utilización de ancho de banda de la red. El equipo de Hadoop está trabajando para mejorar el funcionamiento del sistema de réplica. Los objetivos a corto plazo de aplicación de esta política son validarla en los sistemas de producción, aprender más acerca de su comportamiento, construir una base para poner a prueba y la investigación de políticas más sofisticadas.

Para el caso común en el que el factor de replicación es tres, la política de replicación de HDFS consiste en poner una réplica en un nodo del rack local, otra en un nodo diferente del mismo rack y la última en un nodo diferente en otro rack. La probabilidad de fallo de un rack es mucho menor que la de fallo de un nodo. Con esta política, las réplicas de un archivo no se distribuyen uniformemente a través de los racks. Un tercio de las réplicas están en un nodo, dos tercios de las réplicas están en un mismo rack y el otro tercio se distribuye uniformemente a través de los racks restantes. De este modo se mejora el rendimiento de escritura sin comprometer la fiabilidad de los datos o el rendimiento de lectura frente a otras soluciones de otros sistemas distribuidos.

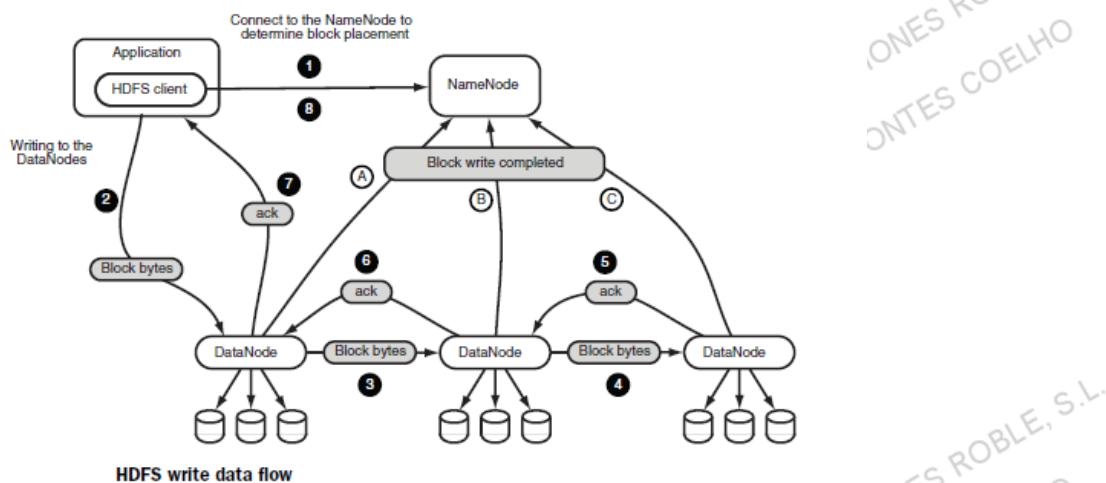


Figura 18. Proceso de escritura en HDFS. Fuente: Fundación Apache.

10.3. El nodo secundario

A pesar de lo que su nombre podría indicar, el SecondaryNameNode no es un NameNode en el sentido de que los DataNodes no pueden conectarse al nodo secundario y no puede reemplazar al nodo primario en caso de error.

El único propósito del SecondaryNameNode es establecer puntos de control. Periódicamente descarga una imagen del NameNode, así como los *logs*, y los une en una nueva imagen que vuelve a cargar sobre el nodo primario.

Por lo tanto, si el NameNode falla y se puede reiniciar en el mismo nodo físico, no es necesario parar los DataNodes sino solo reiniciar el NameNode. Si no se puede usar el antiguo nodo, entonces es necesario copiar la última imagen en otro nodo. La última imagen puede encontrarse en el nodo que se usaba como nodo primario antes del fallo o en el SecondaryNameNode. Este último contendrá el último punto de control con los *logs* asociados, es decir, las últimas modificaciones del NameSpace pueden recuperarse de dichos *logs*. En este caso, también será necesario reiniciar todo el clúster.

El inicio del proceso de los puntos de control en el nodo secundario se controla mediante dos parámetros de configuración:

- **dfs.namenode.checkpoint.period**, que por defecto está establecido a una hora y especifica el máximo retraso entre dos puntos de control consecutivos
- **dfs.namenode.checkpoint.txns**, que por defecto está establecido a un millón y define el número de transacciones sin comprobar en el NameNode que forzará un *checkpoint* urgente, incluso sin que se haya alcanzado el periodo de chequeo.

10.4. Arrancar Hadoop

Antes de poder ejecutar programas sobre los datos almacenados en HDFS, es necesario almacenar los datos en el sistema de archivos.

HDFS tiene como directorio por defecto /user/\$USER, donde \$USER es el *login* del usuario conectado.

1

Lo primero que haremos es formatear el espacio de nombres (se recomienda previamente parar los demonios ./sbin/stop-dfs.sh ./sbin/stop-yarn.sh). Este proceso solo se debe realizar una vez, al principio de la carga de datos, ya que elimina todo el contenido del HDFS.

\$ bin/hdfs namenode -format

```
bigdata@bigdata:~/hadoop$ bin/hdfs namenode -format
17/06/16 12:07:57 INFO namenode.NameNode: STARTUP_MSG:
/*****
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   user = bigdata
STARTUP_MSG:   host = bigdata/127.0.1.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.8.0
STARTUP_MSG:   classpath = /home/bigdata/hadoop/etc/hadoop:/home/bigdata/hadoop/share/hadoop/common/lib/junit-4.11.jar:/home/bigdata/hadoop/share/hadoop/common/lib/jaxb-api-2.2.2.jar:/home/bigdata/hadoop/share/hadoop/common/lib/htrace-core4-4.0.1-incubating.jar:/home/bigdata/hadoop/share/hadoop/common/lib/jsp-api-2.1.jar:/home/bigdata/hadoop/share/hadoop/common/lib/jcip-annotations-1.0.jar:/home/bigdata/hadoop/share/hadoop/common/lib/jackson-jaxrs-1.9.13.jar:/home/bigdata/hadoop/share/hadoop/common/lib/stax-api-1.0-2.jar:/home/bigdata/hadoop/share/hadoop/common/lib/hadoop-auth-2.8.0.jar:/home/bigdata/hadoop/share/hadoop/common/lib/java-xmlbuilder-0.4.jar:/home/bigdata/hadoop/share/hadoop/common/lib/netty-3.6.2.Final.jar:/home/bigdata/hadoop/share/hadoop/common/lib/jettison-1.1.jar:/home/bigdata/hadoop/share/hadoop/common/lib/jersey-server-1.9.jar:/home/bigdata/hadoop/share/hadoop/common/lib/commons-io-2.4.jar:/home/bigdata/hadoop/share/hadoop/common/lib/snappy-java-1.0.4.1.jar:/home/bigdata/hadoop/share/hadoop/common/lib/slf4j-api-1.7.10.jar:/home/bigdata/hadoop/share/hadoop/common/lib/jets3t-0.9.0.jar:/home/bigdata/hadoop/share/hadoop/common/lib/apacheds-i18n-2.0.0-M15.jar:/home/bigdata/hadoop/share/hadoop/common/lib/commons-beanutils-1.7.0.jar:/home/bigdata/hadoop/share/hadoop/common/lib/curator-client-2.7.1.jar:/home/bigdata/hadoop/share/hadoop/common/lib/paranamer-2.3.jar:/home/bigdata/hadoop/share/hadoop/common/lib/curator-recipes-2.7.1.jar:/home/bigdata/hadoop/share/hadoop/common/lib/apacheds-kerberos-codec-2.0.0-M15.jar:/home/bigdata/hadoop/share/hadoop/common/lib/commons-configuration-1.6.jar:/home/bigdata/hadoop/share/hadoop/common/lib/hamcrest-1.3.jar
17/06/16 12:07:59 INFO util.GSet: capacity      = 2^15 = 32768 entries
Re-format filesystem in Storage Directory /home/bigdata/hadoop_store/hdfs/namenode ? (Y or N) Y
17/06/16 12:08:10 INFO namenode.FSImage: Allocated new BlockPoolId: BP-358608126-127.0.1.1-1497607690850
17/06/16 12:08:10 INFO common.Storage: Storage directory /home/bigdata/hadoop_store/hdfs/namenode has been successfully formatted.
17/06/16 12:08:11 INFO namenode.FSImageFormatProtobuf: Saving image file /home/bigdata/hadoop_store/hdfs/namenode/current/fsimage.ckpt_00000000000000000000 using no compression
17/06/16 12:08:11 INFO namenode.FSImageFormatProtobuf: Image file /home/bigdata/hadoop_store/hdfs/namenode/current/fsimage.ckpt_00000000000000000000 of size 324 bytes saved in 0 seconds.
17/06/16 12:08:11 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
17/06/16 12:08:11 INFO util.ExitUtil: Exiting with status 0
17/06/16 12:08:11 INFO namenode.NameNode: SHUTDOWN_MSG:
/*****
SHUTDOWN_MSG: Shutting down NameNode at bigdata/127.0.1.1
*****
```

Una vez formateado, arrancaremos los distintos demonios:

```
$ start-dfs.sh
```

```
bigdata@bigdata:~/hadoop$ ./sbin/start-dfs.sh
Starting namenodes on [localhost]
localhost: starting namenode, logging to /home/bigdata/hadoop/logs/hadoop-bigdata-namenode-bigdata.out
localhost: starting datanode, logging to /home/bigdata/hadoop/logs/hadoop-bigdata-datanode-bigdata.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/bigdata/hadoop/logs/hadoop-bigdata-secondarynamenode-bigdata.out
```

```
$ jps
```

```
bigdata@bigdata:~/hadoop$ jps
4289 Jps
3841 NameNode
4178 SecondaryNameNode
3993 DataNode
bigdata@bigdata:~/hadoop$
```

10.4.1. Comandos básicos de HDFS

Una vez tenemos arrancados los demonios de HDFS y formateado el disco, podemos empezar a emplear los comandos HDFS invocándolos desde el *script bin/hdfs*.

Comandos de usuario

- **hdfs dfs [GENERIC_OPTIONS] [COMMAND_OPTIONS]**: sirve para ejecutar un comando del sistema de archivos soportado por Hadoop. Algunos de los posibles comandos son¹²:
 - appendToFile
 - cat
 - chgrp
 - chmod
 - chown
 - copyFromLocal
 - copyToLocal
 - count
 - cp
 - du
 - dus
 - expunge
 - get

- getfacl
 - getfattr
 - getmerge
 - ls
 - lsr
 - mkdir
 - moveFromLocal
 - moveToLocal
 - mv
 - put
 - rm
 - rmr
 - setfacl
 - setfattr
 - setrep
 - stat
 - tail
 - test
 - text
 - touchz
- hdfs fetchdt [GENERIC_OPTIONS] [--webservice <namenode_http_addr>] <path>: obtener la delegación de *token* del NameNode.
 - hdfs fsck [GENERIC_OPTIONS] <path> [-list-corruptfileblocks | [-move | -delete | -openforwrite] [-files [-blocks [-locations | -racks]]]] [-includeSnapshots]: ejecuta la comprobación de disco.
 - hdfs version: muestra la versión.

¹²Estos comandos de usuario se pueden ver con más detalle en la página web de Apache Hadoop. [En línea] URL disponible en: <http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/FileSystemShell.html>

Comandos de administrador

- hdfs balancer [-threshold <threshold>] [-policy <policy>]: lanza la utilidad de balanceador del clúster.
- hdfs namenode: lanza un NameNode en HDFS.
- hdfs datanode [-regular | -rollback | -rollingupgrade rollback]: lanza un DataNode en HDFS.
- hdfs secondarynamenode [-checkpoint [force]] | [-format] | [-geteditsize]: lanza un nodo secundario.
- hdfs dfsadmin [GENERIC_OPTIONS]: ejecuta un cliente de dfsadmin.
- hdfs mover [-p <files/dirs> | -f <local file name>]: utilidad de migración de datos.

10.5. Ejemplo de uso de comandos básicos HDFS

A continuación, se muestra un ejemplo de uso de los comandos básicos de HDFS.

1

Comprobación de la información que hay en el sistema de archivos:

```
$ hdfs dfs -ls /
```

```
bigdata@bigdata:~/hadoop$ hdfs dfs -ls /
bigdata@bigdata:~/hadoop$
```

2

Creación de un directorio para los usuarios de la máquina:

```
$ hdfs dfs -mkdir /user
```

```
bigdata@bigdata:~/hadoop$ hdfs dfs -mkdir /user
```

3

Creación de un subdirectorio para nuestro usuario:

```
$ hdfs dfs -mkdir /user/bigdata
```

```
bigdata@bigdata:~/hadoop$ hdfs dfs -mkdir /user/bigdata
```

4

Creación de un fichero de ejemplo y asociación al usuario:

```
$ sudo nano ejemplo.txt
```

```
bigdata@bigdata:~/hadoop$ sudo nano ejemplo.txt  
[sudo] password for bigdata:
```

Para ver el código que necesitas, pulsa [aquí](#).

GNU nano 2.5.3

Archivo: ejemplo.txt

```
Ejemplo para mostrar el funcionamiento  
de HDFS y Map Reduce  
1. Este es un ejemplo de fichero.  
2. Otra linea de fichero para usar en map reduce con HDFS.  
3. Escribiremos varias líneas  
4. para después poder emplear  
5. comandos como TAIL  
6. o HEAD  
7. y así mostrar su funcionamiento  
8.  
9.  
10.  
11.  
12.  
13.  
14.  
15.  
17.  
18.
```

5

Movimiento del fichero a HDFS:

```
$ hdfs dfs -put ejemplo.txt /user/bigdata
```

```
bigdata@bigdata:~/hadoop$ hdfs dfs -put ejemplo.txt /user/bigdata  
bigdata@bigdata:~/hadoop$ █
```

6

Ejecución de un listado del directorio recursivo para ver si se han subido correctamente los datos:

```
$ hdfs dfs -ls -R /
```

```
bigdata@bigdata:~/hadoop$ hdfs dfs -ls -R /
drwxr-xr-x  - bigdata supergroup          0 2017-06-16 12:24 /user
drwxr-xr-x  - bigdata supergroup          0 2017-06-16 12:25 /user/bigdata
-rw-r--r--  1 bigdata supergroup        466 2017-06-16 12:25 /user/bigdata/ejemplo.txt
bigdata@bigdata:~/hadoop$
```

7

Una vez se ha subido un fichero a la estructura de directorios, es posible trabajar con él mediante los comandos correspondientes.

Para descargar un fichero, se emplea el comando get indicando como segundo parámetro la ruta en la que queremos dejarlo:

```
$ hdfs dfs -get /user/bigdata/ejemplo.txt /tmp
```

```
bigdata@bigdata:~/hadoop$ hdfs dfs -get /user/bigdata/ejemplo.txt /tmp
bigdata@bigdata:~/hadoop$
```

8

Consulta de ficheros:

```
$ hdfs dfs -cat /user/bigdata/ejemplo.txt
```

```
bigdata@bigdata:~/hadoop$ hdfs dfs -cat /user/bigdata/ejemplo.txt
Ejemplo para mostrar el funcionamiento
de HDFS y Map Reduce
1. Este es un ejemplo de fichero.
2. Otra linea de fichero para usar en map reduce con HDFS.
3. Escribiremos varias lineas
4. para despues poder emplear
5. comandos como TAIL
6. o HEAD
7. y asi mostrar su funcionamiento
8.
9.
10.
11.
12.
13.
14.
15.
17.
18.
19.
20.
21.
22.
23.
24.
25.
```

```
$ hdfs dfs -cat /user/bigdata/ejemplo.txt | head
```

```
bigdata@bigdata:~/hadoop$ hdfs dfs -cat /user/bigdata/ejemplo.txt|head
Ejemplo para mostrar el funcionamiento
de HDFS y Map Reduce
1. Este es un ejemplo de fichero.
2. Otra linea de fichero para usar en map reduce con HDFS.
3. Escribiremos varias lineas
4. para despues poder emplear
5. comandos como TAIL
6. o HEAD
7. y asi mostrar su funcionamiento
8.
bigdata@bigdata:~/hadoop$
```

```
$ hdfs dfs -cat /user/bigdata/ejemplo.txt |tail
```

```
bigdata@bigdata:~/hadoop$ hdfs dfs -cat /user/bigdata/ejemplo.txt|tail
41.
42.
43.
44.
45.
46.
47.
48.
49.
50. Fin del ejemplo
bigdata@bigdata:~/hadoop$
```

9

Eliminar un fichero:

```
$ hdfs dfs -rm /user/bigdata/ejemplo.txt
```

```
bigdata@bigdata:~/hadoop$ hdfs dfs -rm /user/bigdata/ejemplo.txt
Deleted /user/bigdata/ejemplo.txt
bigdata@bigdata:~/hadoop$
```

10

Listar directorios otra vez:

```
$ hdfs dfs -ls -R /
```

```
bigdata@bigdata:~/hadoop$ hdfs dfs -ls -R /
drwxr-xr-x  - bigdata supergroup          0 2017-06-16 12:24 /user
drwxr-xr-x  - bigdata supergroup          0 2017-06-16 12:30 /user/bigdata
bigdata@bigdata:~/hadoop$
```

10.6. API de programación de HDFS

Para subir grandes ficheros, se recomienda usar la API de programación de HDFS.

En el siguiente enlace se muestra la API Java Hadoop Apache Package org.apache.hadoop.fs:
<http://hadoop.apache.org/docs/current/api/org/apache/hadoop/fs/package-summary.html>

A continuación, se muestra un ejemplo de subida de ficheros escrito en Java que recibe como entrada un directorio de entrada y la ruta en la que se va a dejar el fichero en HDFS y como resultado ejecuta la subida de todos los ficheros que existen en dicho directorio y los mezcla para dejarlos en un único fichero (en el indicado como segundo parámetro).

Para ver el código que necesitas, pulsa [aquí](#).

10.7. Ejemplos para ejercicio de evaluación

Para poder terminar de asimilar los conceptos de HDFS, se plantea un ejercicio que introduce el concepto de MapReduce.

Para simplificar el aprendizaje, hemos dividido el ejercicio en dos apartados:

Ejemplo de uso de HDFS

En este ejercicio, se trabajará con los comandos básicos de HDFS.

1

Paso 1. Crear un directorio HDFS, subir un fichero y mostrarlo

La sintaxis es la siguiente:

- `hdfs dfs -mkdir <paths>`: recibe un *uri path* como argumento y crea uno o varios directorios.

Ejemplo:

```
hdfs dfs -mkdir /user/bigdata/dir1 /user/bigdata/dir2
```

```
hdfs dfs -mkdir hdfs://localhost:9000/user/bigdata/dir3
```

```
bigdata@bigdata:~/hadoop$ hdfs dfs -mkdir /user/bigdata/dir1 /user/bigdata/dir2
bigdata@bigdata:~/hadoop$ hdfs dfs -mkdir hdfs://localhost:9000/user/bigdata/dir3
bigdata@bigdata:~/hadoop$
```

- `hdfs dfs -ls <args>`: lista el contenido de un directorio. Para un fichero, muestra sus estadísticas

Ejemplos:

```
bigdata@bigdata:~/hadoop$ hdfs dfs -put ejemplo.txt /user/bigdata/dir1
```

```
hdfs dfs -put ejemplo.txt /user/bigdata/dir1
```

```
hdfs dfs -ls /user/bigdata/dir1 /user/bigdata/dir2
```

```
bigdata@bigdata:~/hadoop$ hdfs dfs -ls /user/bigdata/dir1 /user/bigdata/dir2
Found 1 items
-rw-r--r-- 1 bigdata supergroup      466 2017-06-16 12:32 /user/bigdata/dir1/ejemplo.txt
bigdata@bigdata:~/hadoop$
```

```
hdfs dfs -ls /user/bigdata/dir1/ejemplo.txt
```

```
bigdata@bigdata:~/hadoop$ hdfs dfs -ls /user/bigdata/dir1/ejemplo.txt
-rw-r--r-- 1 bigdata supergroup      466 2017-06-16 12:32 /user/bigdata/dir1/ejemplo.txt
bigdata@bigdata:~/hadoop$
```

```
hdfs dfs -ls hdfs://localhost:9000/user/bigdata/dir1/
```

```
bigdata@bigdata:~/hadoop$ hdfs dfs -ls hdfs://localhost:9000/user/bigdata/dir1/
Found 1 items
-rw-r--r-- 1 bigdata supergroup      466 2017-06-16 12:32 hdfs://localhost:9000/user/bigdata/dir1/ejemplo.txt
bigdata@bigdata:~/hadoop$
```

- `hdfs dfs -du URI`: muestra el tamaño de los archivos y directorios que contiene un directorio dado o el tamaño de un archivo si el parámetro de entrada es un archivo.

Ejemplo:

```
hdfs dfs -du /user/bigdata/ /user/bigdata/dir1/ejemplo.txt
```

```
bigdata@bigdata:~/hadoop$ hdfs dfs -du /user/bigdata/ /user/bigdata/dir1/ejemplo.txt
466  /user/bigdata/dir1
0    /user/bigdata/dir2
0    /user/bigdata/dir3
466  /user/bigdata/dir1/ejemplo.txt
bigdata@bigdata:~/hadoop$
```

2

Paso 2. Subir y descargar ficheros de HDFS

- hdfs dfs -put <localsrc> ... <HDFS_dest_Path>: copia uno o múltiples ficheros de la ruta local al sistema de archivos de Hadoop (HDFS).

Ejemplo:

```
echo "ejemplo1" > /tmp/ejemplo1.txt
```

```
echo "ejemplo2" > /tmp/ejemplo2.txt
```

```
ls /tmp/ejemplo*.txt
```

```
hdfs dfs -put /tmp/ejemplo1.txt /tmp/ejemplo2.txt /user/bigdata/dir3/
```

```
bigdata@bigdata:~/hadoop$ echo "ejemplo1" > /tmp/ejemplo1.txt
bigdata@bigdata:~/hadoop$ echo "ejemplo2" > /tmp/ejemplo2.txt
bigdata@bigdata:~/hadoop$ ls /tmp/ejemplo*.txt
/tmp/ejemplo1.txt /tmp/ejemplo2.txt /tmp/ejemplo.txt
bigdata@bigdata:~/hadoop$ hdfs dfs -put /tmp/ejemplo1.txt /tmp/ejemplo2.txt /user/bigdata/dir3/
bigdata@bigdata:~/hadoop$
```

- hdfs dfs -get <hdfs_src> <localdst>: copia o descarga un fichero al sistema local de archivos.

Ejemplo:

```
hdfs dfs -get /user/bigdata/dir3/ejemplo1.txt /home/bigdata
```

```
bigdata@bigdata:~/hadoop$ hdfs dfs -get /user/bigdata/dir3/ejemplo1.txt /home/bigdata
bigdata@bigdata:~/hadoop$
```

Paso 3. Otros comandos

hdfs dfs -getmerge <src> <localdst> [addnl]: recibe un directorio de archivos como entrada y concatena los archivos en un único fichero en el destino local.

Ejemplo:

```
hdfs dfs -getmerge /user/bigdata/dir3 ./EjemploMerge.txt
```

Opcional:

addnl: se puede activar para que añada una nueva línea al final de cada archivo.

```
bigdata@bigdata:~/hadoop$ hdfs dfs -ls /user/bigdata/dir3
Found 2 items
-rw-r--r-- 1 bigdata supergroup          9 2017-06-16 12:36 /user/bigdata/dir3/ejemplo1.txt
-rw-r--r-- 1 bigdata supergroup          9 2017-06-16 12:36 /user/bigdata/dir3/ejemplo2.txt
bigdata@bigdata:~/hadoop$ hdfs dfs -getmerge /user/bigdata/dir3 ./EjemploMerge.txt
bigdata@bigdata:~/hadoop$ cat EjemploMerge.txt
ejemplo1
ejemplo2
bigdata@bigdata:~/hadoop$
```

Paso 4. Acceso a la ayuda

Se puede emplear el comando Help para listar todos los comandos soportados por Hadoop Data File System (HDFS).

Ejemplo:

Hdfs dfs -help

```
bigdata@bigdata:~/hadoop$ hdfs dfs -help
Usage: hadoop fs [generic options]
      [-appendToFile <localsrc> ... <dst>]
      [-cat [-ignoreCrc] <src> ...]
      [-checksum <src> ...]
      [-chgrp [-R] GROUP PATH...]
      [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
      [-chown [-R] [OWNER][:GROUP]] PATH...
      [-copyFromLocal [-f] [-p] [-l] [-d] <localsrc> ... <dst>]
      [-copyToLocal [-f] [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
      [-count [-q] [-h] [-v] [-t [<storage type>]] [-u] [-x] <path> ...]
      [-cp [-f] [-p | -p[topax]] [-d] <src> ... <dst>]
      [-createSnapshot <snapshotDir> [<snapshotName>]]
      [-deleteSnapshot <snapshotDir> <snapshotName>]
      [-df [-h] [<path> ...]]
      [-du [-s] [-h] [-x] <path> ...]
      [-expunge]
      [-find <path> ... <expression> ...]
      [-get [-f] [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
      [-getfacl [-R] <path>]
      [-getattr [-R] {-n name | -d} [-e en] <path>]
      [-getmerge [-nl] [-skip-empty-file] <src> <localdst>]
      [-help [cmd ...]]
      [-ls [-C] [-d] [-h] [-q] [-R] [-t] [-S] [-r] [-u] [<path> ...]]
      [-mkdir [-p] <path> ...]
      [-moveFromLocal <localsrc> ... <dst>]
      [-moveToLocal <src> <localdst>]
      [-mv <src> ... <dst>]
      [-put [-f] [-p] [-l] [-d] <localsrc> ... <dst>]
      [-renameSnapshot <snapshotDir> <oldName> <newName>]
      [-rm [-f] [-r|-R] [-skipTrash] [-safely] <src> ...]
      [-rmdir [--ignore-fail-on-non-empty] <dir> ...]
      [-setfacl [-R] [{-b|-k} {-m|-x <acl_spec>} <path>]|[--set <acl_spec> <path>]]
      [-setattr {-n name [-v value] | -x name} <path>]
      [-setrep [-R] [-w] <rep> <path> ...]
      [-stat [format] <path> ...]
      [-tail [-f] <file>]
      [-test [-defsz] <path>]
      [-text [-ignoreCrc] <src> ...]
      [-touchz <path> ...]
      [-truncate [-w] <length> <path> ...]
      [-usage [cmd ...]]]
```

Ejemplo de uso de MapReduce

En un sistema Linux, el comando ls sirve para listar el contenido del directorio raíz:

\$ ls /

```
bigdata@bigdata:~/hadoop$ ls
bin      ejemplo.txt  include  libexec   logs      README.txt  share
EjemploMerge.txt  etc        lib       LICENSE.txt NOTICE.txt  sbin
bigdata@bigdata:~/hadoop$
```

Hadoop proporciona con DFS una herramienta para mostrar las funcionalidades básicas de un sistema de archivos, como por ejemplo la instrucción ls.

```
$ hdfs dfs -ls /
```

```
bigdata@bigdata:~/hadoop$ hdfs dfs -ls /
Found 1 items
drwxr-xr-x  - bigdata supergroup          0 2017-06-16 12:24 /user
bigdata@bigdata:~/hadoop$
```

Vamos a crear un directorio en el que después crearemos un par de ficheros de texto sobre los que después trabajaremos.

```
$ sudo mkdir wc-in
```

```
$ sudo chmod -R 777 wc-in
```

```
$ sudo echo "bla bla" > wc-in/a.txt
```

```
$ sudo echo "bla wa wa" > wc-in/b.txt
```

```
bigdata@bigdata:~/hadoop$ sudo mkdir wc-in
bigdata@bigdata:~/hadoop$ sudo chmod -R 777 wc-in
bigdata@bigdata:~/hadoop$ echo "bla bla" > wc-in/a.txt
bigdata@bigdata:~/hadoop$ echo "bla wa wa" > wc-in/b.txt
bigdata@bigdata:~/hadoop$
```

Copiamos los ficheros anteriores a un directorio en HDFS. Esta vez emplearemos el comando copyFromLocal que es similar al comando put con la salvedad de que el fichero tiene que tener una referencia local.

```
$ hdfs dfs -mkdir /user/bigdata/dir4
```

```
$ hdfs dfs -copyFromLocal wc-in /user/bigdata/dir4
```

```
bigdata@bigdata:~/hadoop$ hdfs dfs -mkdir /user/bigdata/dir4
bigdata@bigdata:~/hadoop$
```

```
bigdata@bigdata:~/hadoop$ hdfs dfs -copyFromLocal wc-in /user/bigdata/dir4
```

Vamos a emplear un ejemplo de MapReduce para saber el número de palabras que contienen los ficheros creados anteriormente.

```
$ hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.8.0.jar wordcount /user/bigdata/dir4/wc-in /user/bigdata/dir4/wc-out
```

```

bigdata@bigdata:~/hadoop$ hadoop jar SHADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.8.0.jar wordcount /user/bigdata/dir4/wc-in /user/bigdata/dir4/wc-out
17/08/12 19:51:04 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
17/08/12 19:51:04 INFO input.FileInputFormat: Total input files to process : 2
17/08/12 19:51:04 INFO mapreduce.JobSubmitter: number of splits:2
17/08/12 19:51:05 INFO mapreduce.JobSubmitter: Submitting application for job: job_1502490164835_0013
17/08/12 19:51:05 INFO impl.YarnClientImpl: Submitted application application_1502490164835_0013
17/08/12 19:51:06 INFO mapreduce.Job: The url to track the job: http://bigdata:8088/proxy/application_1502490164835_0013/
17/08/12 19:51:06 INFO mapreduce.Job: Running job: job_1502490164835_0013
17/08/12 19:51:14 INFO mapreduce.Job: Job job_1502490164835_0013 running in uber mode : false
17/08/12 19:51:14 INFO mapreduce.Job: map 0% reduce 0%
17/08/12 19:51:27 INFO mapreduce.Job: map 100% reduce 0%
17/08/12 19:51:34 INFO mapreduce.Job: map 100% reduce 100%
17/08/12 19:51:34 INFO mapreduce.Job: Job job_1502490164835_0013 completed successfully
17/08/12 19:51:35 INFO mapreduce.Job: Counters: 49
    File System Counters
        FILE: Number of bytes read=35
        FILE: Number of bytes written=409602
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=250
        HDFS: Number of bytes written=11
        HDFS: Number of read operations=9
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=2
    Job Counters

```

Los resultados pueden verse a través del sistema de archivos:

```
$ hdfs dfs -cat /user/bigdata/dir4/wc-out/*
```

bla 3

wa 2

```

bigdata@bigdata:~/hadoop$ hdfs dfs -cat /user/bigdata/dir4/wc-out/*
bla      3
wa      2
bigdata@bigdata:~/hadoop$
```

Para archivos grandes, no se pueden emplear los comandos -more, -head o -tail. En su lugar hay que usar la tubería (pipe "|") junto con el comando cat.

```
hdfs dfs -cat /user/bigdata/dir4/wc-out/* | more
```



Es importante comprender bien los ejemplos para poder abordar los ejercicios de evaluación de este tema.

XI. MAP REDUCE

11.1. Procesamiento paralelo

El concepto de **procesamiento paralelo** se relaciona con el diseño de programas con dos características principales: **corren en múltiples procesadores** (o núcleos) y todos ellos **cooperan para resolver un único problema**.

1

No se debe confundir el **procesamiento** paralelo con el **distribuido**. Por ejemplo, un servidor web, como el de Google, normalmente corre en un clúster formado por **varias máquinas** multinúcleo que atienden y procesan **miles de peticiones** cada segundo de forma simultánea. El navegador web muestra el contenido procesado que llega así al usuario final. Sin embargo, cada una de estas máquinas atiende peticiones individuales, no colaboran para resolver una única tarea. En este caso, por tanto, no se cumple la segunda característica mencionada. Esto es lo que se denomina **procesamiento distribuido**, no paralelo.

En la actualidad existen numerosas aplicaciones en las que se utiliza procesamiento paralelo. Google, por su parte, popularizó el paradigma MapReduce para el procesamiento paralelo de grandes volúmenes de datos y es, quizás, el ejemplo más significativo. Sin embargo, el **procesamiento paralelo se emplea** en muchas otras áreas como las **matemáticas computacionales** (álgebra lineal numérica, soluciones numéricas de ecuaciones diferenciales, optimización, combinatoria, teoría de grafos...), **procesamiento científico** (predicción meteorológica, predicción de huracanes, modelado climatológico, astrofísica, bioinformática...), **ingeniería** (simulación de túneles de viento, análisis de elementos finitos, análisis de circuitos...), finanzas (modelado de mercado, búsqueda de tendencias...), **analítica BigData** (minería de datos, indexación web, caracterización de usuarios..), etc.

2

En cualquier caso, el uso del procesamiento paralelo no ha estado siempre tan extendido. A principios de los años ochenta la computación paralela era cara, cada proveedor ofrecía sus propias soluciones de **hardware** y sus propios lenguajes de programación, de forma que su uso se limitaba a grandes compañías y centros de investigación.

El cambio se produjo a finales del siglo XX debido a la importante reducción de costes de implantación de redes ethernet locales, el abaratamiento de los ordenadores personales, el desarrollo de los protocolos de comunicación TCP/IP y la distribución de Linux como sistema operativo libre. Además, en 1995 se publicó el **paper Beowulf**, en el que se detallaba cómo aunar todos los recursos listados para crear un sistema completo, denominado Beowulf, que sentaba las **bases** de lo que actualmente se conoce como **clúster**. Por otra parte, en esta época se comenzaron a utilizar lenguajes como Fortran, C y C++ como estándares para el desarrollo de *software* de procesamiento paralelo. También se publicaron los estándares **Message Passing Interface** (MPI) y **OpenMP** (1994 y 1997, respectivamente), que se convirtieron en estándares de facto de programación en clústers de equipos de procesamiento paralelo. En cualquier caso, las máquinas multinúcleo no eran una realidad: lo más habitual al implementar sistemas de procesamiento paralelo era construir clústers con múltiples equipos con un único procesador.

3

El segundo hito que marcó el cambio sucedió en **2004**. Los fabricantes de microprocesadores explotaron la **ley de Moore** y comenzaron a desarrollar procesadores **doblando el número de transistores** y la **frecuencia de reloj** cada dos años. Sin embargo, en 2004 se llegó a los 3 GHz y **allímito en la disipación de calor**. De esta forma ya no era posible aumentar la frecuencia de reloj y únicamente quedaba la opción de **incrementar el número de transistores**. Así, se comenzaron a **diseñar micros** formados por **múltiples procesadores operando en paralelo**, con dos relojes, ya que teóricamente dos procesadores a 3 GHz eran equivalentes a uno trabajando a 6. El número de núcleos de procesamiento implementados en un micro ha ido incrementándose hasta la actualidad y hoy ya es posible encontrar procesadores de 8 núcleos relativamente económicos.

Además, la popularización de los videojuegos ha generado también un aumento considerable de las capacidades de memoria principal de los equipos y se encuentran fácilmente dotaciones de 8 o 16 GB. De esta forma se ha llegado a que una única máquina esté dotada de capacidad suficiente para equiparse a los clústers empleados en los años ochenta. Gracias a todo ello, hay disponibles sistemas de procesamiento paralelo prácticamente en cualquier lugar del mundo.

4

Por otra parte, las aplicaciones modernas que corren sobre estos sistemas se desarrollan en lenguajes nuevos como **Java** y empleando nuevos paradigmas de programación como **MapReduce**. El ejemplo más significativo es la librería MapReduce de Hadoop, escrita en Java y diseñada para ejecutar tareas de procesamiento paralelo en conjuntos de datos BigData.

No obstante, se ha llegado a un punto de tales **necesidades de procesamiento** que incluso un equipo equiparable a un clúster de los años ochenta **no es suficiente** para realizar determinadas tareas sobre los volúmenes de datos que se manejan en la actualidad. Por ello se deben combinar formando **clústers** de equipos **multiprocesamiento**. En la actualidad, estos centros de procesamiento se denominan **supercomputadores**. En noviembre de 2014, el mayor supercomputador del mundo era el National Supercomputer Center de Guangzhou en China; consta de 3.120.000 núcleos y funciona a una frecuencia de pico de 54.902 tflops por segundo, consumiendo una potencia de 17.808 KW.

5

En cualquier caso, **los clústers siguen siendo sistemas caros** y no están al alcance de cualquier empresa o centro que lo requiera. En los últimos años se ha desarrollado como alternativa lo que se denomina *cloud computing*. Empresas como Amazon proveen servicios de computación en la nube, como es el caso de EC2. En este modelo, los **nodos de procesamiento se alquilan** en función de las necesidades de cada cliente en lo que a número y tiempo se refiere. Además, presenta la ventaja de que **suprime** las tareas de **mantenimiento**.

No obstante, la creciente capacidad de procesamiento paralelo no tiene sentido si no **se desarrolla software capaz de sacarle rendimiento**. Para desarrollar este tipo de *software* es necesario conocer, a su vez, en qué se basa el procesamiento paralelo. Esto es lo que se discute en el siguiente epígrafe.

11.2. Fundamentos

El procesamiento paralelo se puede caracterizar por tres dimensiones bien definidas: *hardware*, *software* y aplicación.

11.2.1. Dimensión hardware

Los actores fundamentales de la computación paralela relacionados con el *hardware* son los siguientes:

Nodo

Ordenador independiente que posee su propia CPU, su propia memoria principal y su propia interfaz de red. Un sistema de procesamiento paralelo puede consistir en un único nodo o varios. En caso de ser un sistema **multinodo**, este se denomina *clúster*.

Núcleo

Unidad de procesamiento de un nodo, ejecuta secuencias de instrucciones. Un nodo puede tener un único núcleo o varios y estos, a su vez, pueden ejecutar más de una secuencia de instrucciones; es lo que se denomina “*multihilo*” o *hyperthreaded*.

Aceleradores

Procesadores independientes de propósito específico que pueden ejecutar operaciones junto a la CPU. Un ejemplo común son los GPU (*Graphics Processing Units*), que suelen estar formados, a su vez, por varios núcleos; o las FPGA (*Field Programmable Gate Array*), chips digitales cuyos circuitos lógicos pueden ser reconfigurados según los requerimientos y permiten crear procesadores de gran velocidad personalizados.

Memoria principal

Celdas de almacenamiento aleatorio en las que se alojan las secuencias de instrucciones y variables de ejecución. Debido a que la circuitería de las memorias principales es más lenta que la del núcleo de procesamiento, se inserta una memoria intermedia, denominada caché, dividida a su vez en dos niveles relacionados con la velocidad de lectura, mucho más rápida que la memoria principal pero de menor tamaño.

11.2.2. Dimensión software

1

Un programa paralelo consiste en un **conjunto de hilos que realizan operaciones simultáneamente** corriendo en los núcleos de los nodos. Las **operaciones** realizadas por cada hilo de procesamiento pueden estar **desacopladas, semi acopladas o totalmente acopladas**, en función de si los hilos se comunican o coordinan entre ellos para obtener el resultado o no.

Así, cuando se desarrolla un *software* de procesamiento paralelo se debe tener muy presente el hecho de que el resultado debe ser correcto, y es de suma importancia cuando las tareas desarrolladas por cada hilo están acopladas total o parcialmente.

Obtener soluciones correctas en un entorno multihilo puede ser una tarea desafiante. El hecho de que los hilos comparten espacios de memoria facilita la programación de *software* paralelo pero también es un punto de fallo habitual ya que los hilos pueden compartir datos de formas que el programador podría no imaginar. Si la salida de un programa cambia según cambia la gestión de los hilos, entonces se están produciendo lo que se conoce como “*condiciones carrera*”. Es uno de los principales puntos de fallo ya que, además, es prácticamente imposible asegurar que un determinado programa no contiene condiciones carrera, podría funcionar correctamente las mil primeras veces y fallar en la siguiente.



Considérese por ejemplo un programa que calcula dos resultados (A y B) y, posteriormente, los combina para proporcionar una respuesta final (Res). Supóngase además que el programa ejecuta dichas instrucciones empleando dos hilos paralelos que ejecutan las siguientes instrucciones:

Hilo 1	Hilo 2
$A = \text{BigJob}()$	$B = \text{BigJob}()$
$\text{Res} += A$	$\text{Res} += B$

En caso de que el cálculo de uno de los valores, supóngase el de A, requiera mucho más tiempo que el otro, supóngase el de B, este programa generaría, probablemente, la respuesta adecuada. Sin embargo, si ambos procesos toman tiempos similares, el resultado puede ser imprevisible. Por ejemplo, si A = 1, B = 2, y Res = 3 inicialmente, entonces los posibles resultados serían:

Valor final de Res	Secuencia de operaciones
6	Si los cálculos de A y B llevan tiempos muy diferentes.
5	Si el Hilo 1 lee Res pero mientras realiza la suma el Hilo 2 lee Res, hace la suma y escribe el resultado antes de que el Hilo 1 termine.
4	Si el Hilo 2 lee Res, pero mientras está realizando la suma el Hilo 1 lee Res, hace la suma y escribe el resultado antes de que el Hilo 2 termine.

Así queda demostrado que la corrección del dato devuelto no es un hecho trivial.

2

En cualquier caso, el desarrollo de este tipo de *software* se fundamenta en la idea de ejecutar una serie de instrucciones lo más rápido posible y es, por tanto, una cuestión de rendimiento.

En este sentido, el rendimiento consta de dos aspectos fundamentales: latencia *latency* y flujo o *throughput*. La **latencia** se relaciona con el **tiempo necesario para completar una determinada tarea** y resulta crítica en aplicaciones interactivas como movimientos de ratón, etc. El **flujo**, por su parte, se relaciona con el **número de tareas** que se pueden ejecutar en una **determinada cantidad de tiempo**. En este caso, una tarea individual puede requerir mucho tiempo para su ejecución, pero desde este punto de vista no tiene importancia si el tiempo total consumido para completar un conjunto de tareas es reducido. Un ejemplo de ello es la renderización de una secuencia de *frames* de una película animada: no tiene importancia si un *frame* en concreto requiere mucho tiempo para renderizarse, lo importante es el tiempo total.

En cualquier caso, tanto si lo que se busca es disminuir la latencia como aumentar el flujo, lo que importa es el **rendimiento global**. Es habitual medir el rendimiento del *software* paralelo en lo que se denomina *speed-up*, una ratio que relaciona el tiempo de ejecución de *software* paralelo con el tiempo de ejecución de la mejor implementación secuencial disponible del mismo algoritmo:

$$S = ts / tp$$

donde **ts** es el **tiempo de ejecución secuencial** y **tp** el **tiempo de ejecución paralelo**.

Si el programa de ejecución paralela es perfecto, entonces la ratio anterior escala linealmente con el número de elementos de procesamiento (*P*), es decir, *S* se iguala a *P*. En este caso, doblar el número de núcleos de procesamiento implicaría la reducción a la mitad del tiempo de ejecución.

3

No obstante, la situación de linealidad perfecta anterior no suele darse; en la mayor parte de los casos hay fracciones de los algoritmos que no pueden ejecutarse en paralelo. La fracción secuencial en la solución de un problema y su impacto sobre la ratio *S* es lo que se denomina "*ley Amdahl*". Considérese un problema con un tiempo total de ejecución *T(s)*, con una parte que solo puede ser resuelta secuencialmente (*fs*) y otra con solución paralela (*fp*). La parte secuencial puede relacionarse, por ejemplo, con I/O, costes de inicio, procesamiento secuencial de resultados paralelos, etc. Así, según se incrementa el número de procesadores disponibles para ejecutar el algoritmo, tan solo la parte paralela se ve beneficiada, de forma que el tiempo de procesamiento sigue la siguiente ecuación:

$$T(p) = (fs + fp / P) T(s) = (fs + (1-fs) / P) T(s)$$

Sustituyendo la expresión anterior en la ecuación de ratio *S*, se obtiene:

$$S = T(s) / (fs + (1-fs) / P) T(s) = 1 / (fs + (1-fs) / P)$$

De forma que si *P* tiende a infinito, se obtiene que

$$S = 1/fs$$

Las ecuaciones anteriores muestran que el impacto que tiene la parte secuencial de un algoritmo es vital cuando se está desarrollando un *software* de procesamiento paralelo. De hecho si, por ejemplo, se paralleliza el 80 % de un algoritmo, la mejor ratio alcanzable será 5, independientemente del número de procesadores disponibles. Es por tanto un requisito imprescindible tener en cuenta la ley Amdahl cuando se desarrolla este tipo de *software*.

11.3. Aplicación

Las aplicaciones de procesamiento paralelo se caracterizan por dos dimensiones ortogonales: CPU pequeña - CPU grande y pocos datos - muchos datos.

11.3.1. CPU pequeña y pocos datos

Una aplicación con bajos requerimientos de CPU y que trabaja con pocos datos suele requerir pocos cálculos (ciclos de CPU) para completarse con cada dato. En cualquier caso, los cálculos pueden realizarse en paralelo. Una hoja de cálculo es un ejemplo de ello, trabaja con muy pocos datos (los valores contenidos en las celdas) y se realizan pocas operaciones (las fórmulas de las celdas). Sin embargo, los cálculos pueden realizarse en paralelo puesto que no hay dependencias entre celdas.

11.3.2. CPU grande y pocos datos

En este caso también se trabaja con pocos datos, pero se requieren muchos ciclos de CPU para completar los cálculos sobre los mismos. Realizar los cálculos en paralelo puede acelerar considerablemente la ejecución. Las aplicaciones criptográficas entran dentro de este marco de procesamiento, la minería Bitcoin¹³ es un ejemplo. Un bloque bitcoin es una pieza de dinero electrónico, ocupa pocos kilobytes de datos pero determinar su valor (lo que se denomina "minar el bitcoin") requiere calcular la función hash SHA256 muchas veces. Estas operaciones pueden realizarse en paralelo, de hecho es una práctica habitual.

¹³Bitcoinmining.com: "What is Bitcoin Mining?". [En línea] URL disponible en: <https://www.bitcoinmining.com/>

11.3.3. CPU pequeña y muchos datos

Una aplicación de este tipo requiere poco tiempo de CPU para obtener el resultado de cada dato, pero la operación se aplica a una gran cantidad de ellos. Debido a esto, la aplicación puede requerir mucho tiempo para concluir y el procesamiento paralelo puede mejorar el rendimiento considerablemente. MapReduce, desarrollado por Google e implementado por Apache Hadoop, es un paradigma de procesamiento paralelo para aplicaciones sobre una gran cantidad de datos.

11.3.4. CPU grande y muchos datos

En este tipo de aplicaciones se realizan numerosos cálculos sobre una gran cantidad de datos. Las aplicaciones científicas que corren en supercomputadores se incluyen en este tipo. Un ejemplo extremo es el programa LAMMPS¹⁴, que simula el movimiento de los átomos desde los principios de la física y corre sobre el supercomputador Keneeland en el Laboratorio Nacional de Oak Ridge¹⁵ de EE. UU. Keeneland es un clúster de medio tamaño que consta de 120 nodos con dos núcleos y tres aceleradores GPU por nodo.

¹⁴LAMMPS Molecular Dynamics Simulator. [En línea] URL disponible en: <http://lammps.sandia.gov/>

¹⁵Página web del Laboratorio Nacional de Oak Ridge. [En línea] URL disponible en: <https://www.ornl.gov/>

11.4. Ejemplo sencillo de MapReduce

MapReduce es un paradigma de programación *software* que abarca las dimensiones desarrolladas en los epígrafes anteriores. Como se ha visto en otras ocasiones, se basa en dos procedimientos marcados, un procedimiento *map* que genera un conjunto de datos tipo clave–valor y otra *reduce* que combina, de la forma que se establezca, los resultados de la función *map*. La figura 19 muestra de forma resumida el funcionamiento de MapReduce:

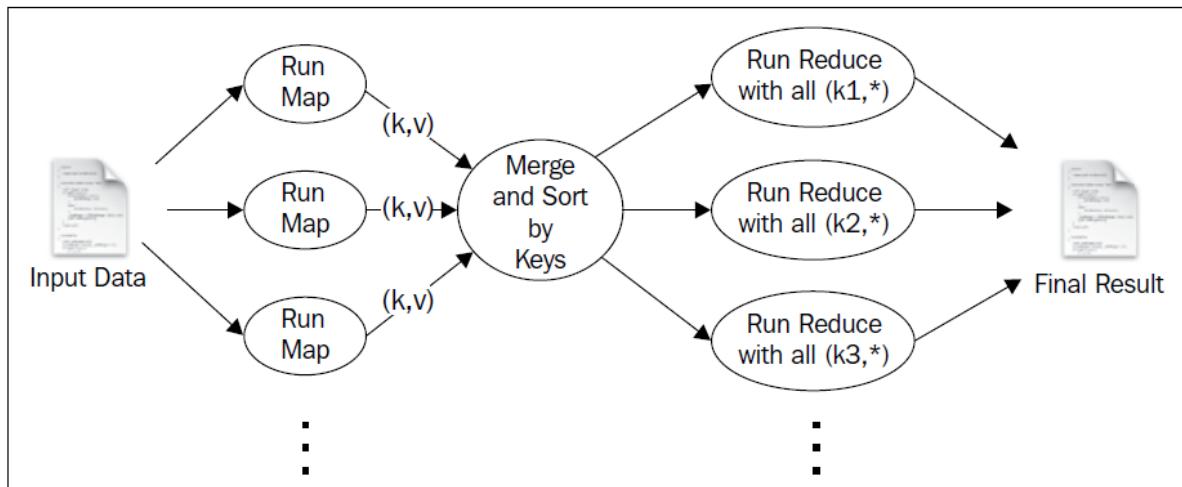


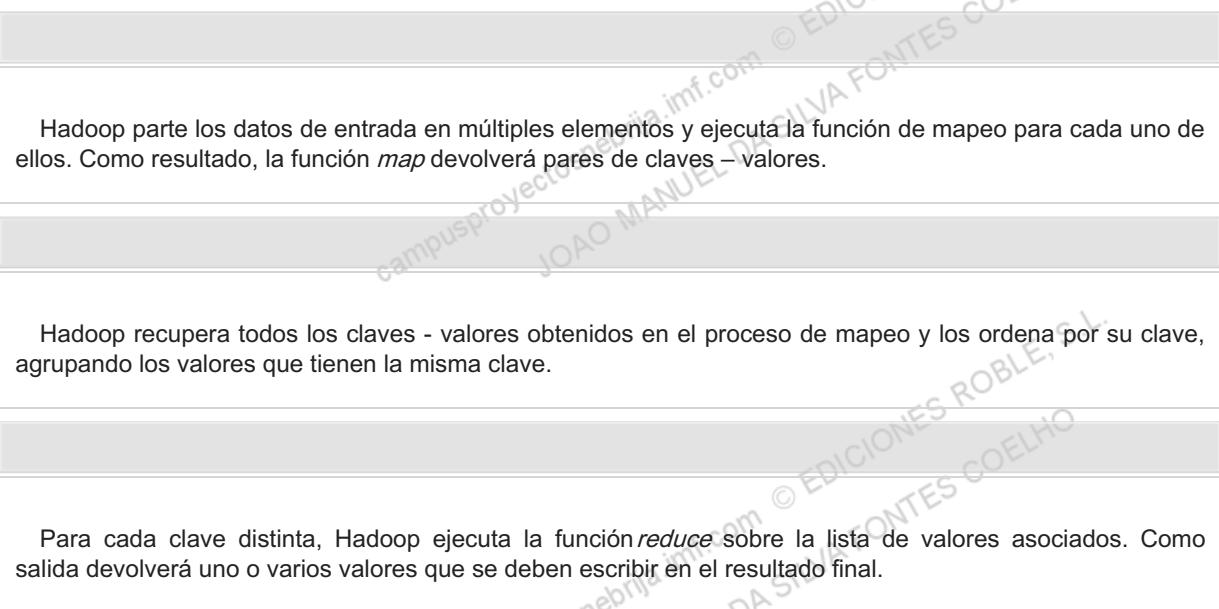
Figura 19. Funcionamiento de MapReduce.

Fuente: Fundación Apache.

Para poder ejecutar un proceso en MapReduce, hace falta:

- Datos de entrada.
- Una función de mapeo.
- Una función *reduce*.
- Datos de salida.

El proceso ejecutado consta de los siguientes pasos:



Hadoop MapReduce es un *framework* que permite escribir aplicaciones que procesan grandes cantidades de datos en paralelo sobre clústers de *hardware* básico en un entorno confiable y tolerante a fallos.

El *framework* de MapReduce consiste en un demonio JobTracker maestro y un demonio esclavo TaskTracker en cada uno de los nodos del clúster. El demonio maestro es el responsable de la programación de las tareas de los componentes de los *jobs* en los esclavos, de monitorizarlos y de reejecutarlos cuando falla alguna de las tareas. Los demonios esclavos ejecutan las tareas ordenadas por el demonio maestro.

Aunque el *framework* de Hadoop ha sido implementado en Java TM, las aplicaciones de MapReduce no tienen por qué estar escritas en Java.

11.5. Explicación del modelo MapReduce

Antes hemos utilizado un .jar que contaba el número de veces que una palabra aparecía en un texto. Este es el ejemplo estándar de “Hola Mundo” de MapReduce, que se compone de tres fases.

En la fase de mapeo, lee cada una de las líneas del texto, una cada vez. Después trocea cada palabra en la cadena y para cada palabra, muestra la palabra y un contador que indica el número de veces que la palabra ha aparecido.

En la fase de mezcla, empleará la palabra como clave y hará *unhash* con los registros para prepararlos para la fase de reducción.

En la fase de reducción, realizará la suma del número de veces que cada palabra ha sido vista y la escribirá junto con la palabra como salida.



Vamos a intentar explicarlo con el siguiente ejemplo:

En un lugar de la Mancha,

de cuyo nombre no quiero acordarme,

no ha mucho tiempo que vivía un hidalgo

de los de lanza en astillero,

adarga antigua, rocín flaco y galgo corredor.

Se asume que cada una de las líneas se envía a una tarea de mapeo distinta. En realidad, a cada mapeo se le suele asignar una cantidad mayor de información, pero se ha simplificado por motivos de claridad. Además asumiremos que en la fase de reducción se van a tener solo dos reductores que dividen las palabras de A-L y de M-Z. Por lo tanto, el flujo de datos quedaría de la siguiente forma:

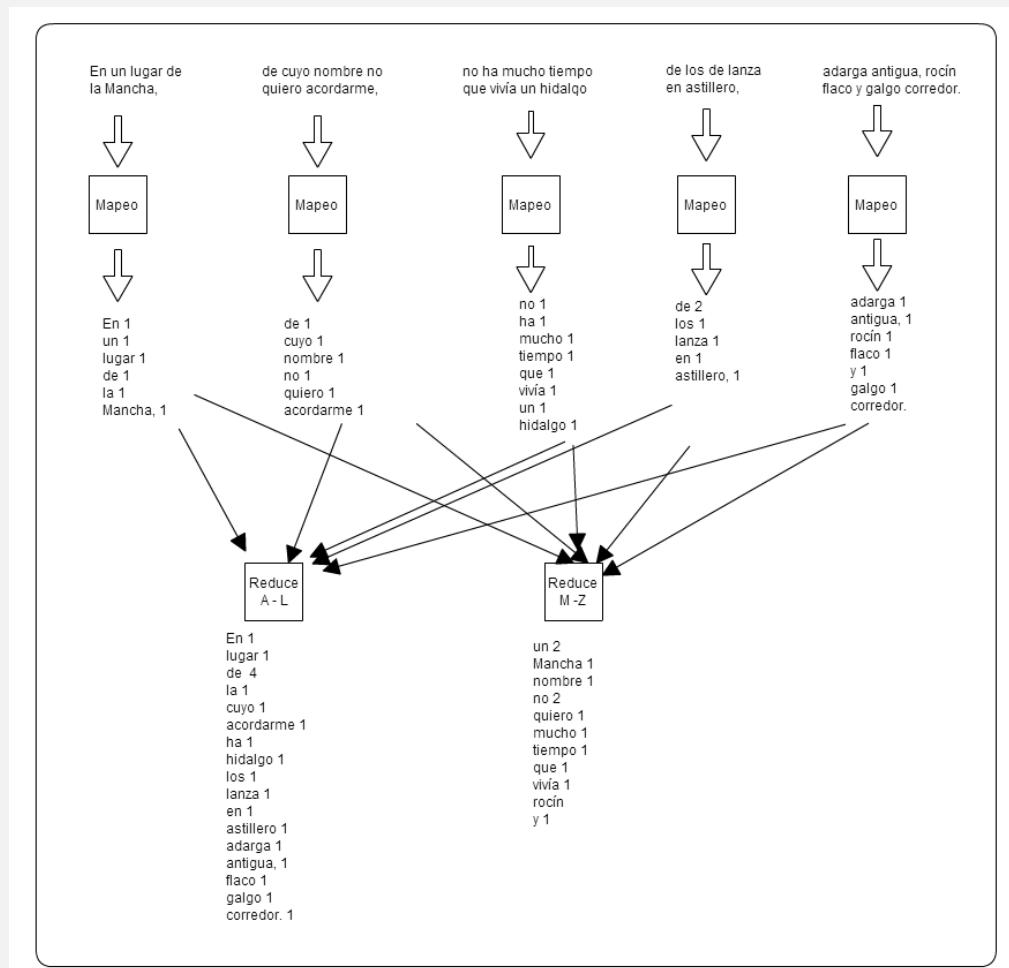


Figura 20. Ejemplo sencillo de MapReduce. Fuente: Fundación Apache.

11.6. Ejemplo

En este epígrafe se presenta un ejemplo completo de MapReduce. Para ello se parte del ejemplo wordcount proporcionado por la documentación de Hadoop y, dada una serie de datos de entrada, se verá cómo el sistema realiza los cálculos del número de palabras que aparecen en los mismos y el número de veces que se repite cada una de ellas.

11.6.1. Datos de entrada

El ejemplo va a partir de una serie de ficheros de texto con una información asociada a los mismos, la creación de dichos ficheros es como sigue:

```
$ mkdir mapreduce
$ cd mapreduce/
$ sudo mkdir wc-in
$ sudo chmod -R 777 wc-in
$ sudo echo -e "Hello World: Bye World" > wc-in/a.txt
$ sudo echo -e "Hello Hadoop Goodbye Hadoop" > wc-in/b.txt
```

```
bigdata@bigdata:~$ mkdir mapreduce
bigdata@bigdata:~$ cd mapreduce/
bigdata@bigdata:~/mapreduce$ sudo mkdir wc-in
bigdata@bigdata:~/mapreduce$ sudo chmod -R 777 wc-in
bigdata@bigdata:~/mapreduce$ sudo echo -e "Hello World: Bye World" > wc-in/a.txt
bigdata@bigdata:~/mapreduce$ sudo echo -e "Hello Hadoop Goodbye Hadoop" > wc-in/b.txt
bigdata@bigdata:~/mapreduce$
```

A. Mapeo

El punto de partida del proceso de MapReduce es la fase de mapeo. Durante esta fase, se capturan los datos de entrada y se transforman en un par clave valor que se empleará en el proceso intermedio. Los registros transformados no tienen por qué ser el mismo número de registros que los de entrada. Un par de entrada puede devolver cero elementos mapeados o múltiples pares, e incluso pueden ser de distinto tipo.

La implementación del mapeo se envía mediante la clase indicada en la tarea con la función `setMapperClass`. El *framework* de Hadoop llamará a la función `map` (`WritableComparable`, `Writable`, `Context`) para cada uno de los elementos clave valor que recibe en la entrada.



En este ejemplo, se procesará cada línea del fichero de entrada y posteriormente se dividirá en *tokens* separados por espacios en blanco.

Para ver el código que necesitas, pulsa [aquí](#).

Como resultado del ejemplo se obtiene la siguiente información para cada uno de los ficheros.

Para el primer fichero (ejemplo de MapReduce: “Hello, World. Bye, World”):

< Hello, 1>

< World., 1>

< Bye, 1>

< World., 1>

El mapeo del segundo fichero ("Hello, Hadoop, Goodbye, Hadoop") devuelve lo siguiente:

< Hello, 1>

< Hadoop, 1>

< Goodbye, 1>

< Hadoop, 1>

B. Mezcla

Todos los pares clave - valor intermedios resultantes de la fase de mapeo se emplearán como entrada a la función de mezcla o agrupación. Esta función se controla mediante el método setCombinerClass:

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

En este caso, el proceso de mezcla o combinación emplea la misma clase que el proceso de reducción.

El código asociado a esta clase se muestra a continuación (para cada una de las palabras, suma el número de repeticiones de la misma):

Para ver el código que necesitas, pulsa [aquí](#).

La salida del primer bloque será:

< Bye, 1>

< Hello, 1>

< World., 2>

La salida del segundo bloque será:

< Goodbye, 1>

< Hadoop, 2>

< Hello, 1>

C. Reducción

Por último, se aplica la fase de reducción para, partiendo de los elementos de la mezcla, obtener el resultado. En este caso la función se especifica mediante el método setReducerClass.

Para ver el código que necesitas, pulsa [aquí](#).

Como resultado, se obtiene lo siguiente:

```
< Bye, 1>
< Goodbye, 1>
< Hadoop, 2>
< Hello, 2>
< World., 2>
```

11.6.2. Resultado

A continuación se presenta el ejemplo completo. En primer lugar, se parte del siguiente fichero Java que contiene las funciones *map* y *reduce*.

Para ver el código que necesitas, pulsa [aquí](#).

Debemos generar un jar encargado de ejecutar las funciones anteriores. Para ello se siguen los siguientes pasos.

1

1. Crear un fichero wordcount.java con el código anterior:

```
$ sudo mkdir /home/bigdata/mapreduce
```

```
$ cd /home/bigdata/mapreduce
```

```
$ sudo nano WordCount.java
```

```
bigdata@bigdata:~/mapreduce$ cd /home/bigdata/mapreduce
bigdata@bigdata:~/mapreduce$ sudo nano WordCount.java
bigdata@bigdata:~/mapreduce$
```

2

2. Compilar el fichero:

```
$ sudo javac -classpath $HADOOP_HOME/share/hadoop/common/hadoop-common-2.8.0.jar:$HADOOP_HOME/share/hadoop/common/lib/hadoop-annotations-2.8.0.jar:$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-core-2.8.0.jar /home/bigdata/mapreduce/WordCount.java
```

```
bigdata@bigdata:~/mapreduce$ sudo javac -classpath $HADOOP_HOME/share/hadoop/common/hadoop-common-2.8.0.jar:$HADOOP_HOME/share/hadoop/common/lib/hadoop-annotations-2.8.0.jar:$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-core-2.8.0.jar /home/bigdata/mapreduce/WordCount.java
bigdata@bigdata:~/mapreduce$ ls
WordCount.class WordCount$IntSumReducer.class WordCount.java WordCount$TokenizerMapper.class
bigdata@bigdata:~/mapreduce$
```

3

3. Crear un jar:

```
$ sudo jar cf wc.jar WordCount*.class
```

```
bigdata@bigdata:~/mapreduce$ sudo jar cf wc.jar WordCount*.class
bigdata@bigdata:~/mapreduce$ ls
wc.jar WordCount$IntSumReducer.class WordCount$TokenizerMapper.class
WordCount.class WordCount.java
bigdata@bigdata:~/mapreduce$
```

4

4. Crear los ficheros de entrada de datos:

```
$ sudo mkdir wc-in
```

```
$ sudo chmod -R 777 wc-in
```

```
$ sudo echo "Hello World. Bye World." > wc-in/a.txt
```

```
$ sudo echo "Hello Hadoop Goodbye Hadoop" > wc-in/b.txt
```

```
bigdata@bigdata:~/mapreduce$ sudo mkdir wc-in
bigdata@bigdata:~/mapreduce$ sudo chmod -R 777 wc-in
bigdata@bigdata:~/mapreduce$ sudo echo "Hello World. Bye World." > wc-in/a.txt
bigdata@bigdata:~/mapreduce$ sudo echo "Hello Hadoop Goodbye Hadoop" > wc-in/b.txt
bigdata@bigdata:~/mapreduce$
```

```
$ hdfs dfs -mkdir /user/bigdata/wc-in
```

```
$ hdfs dfs -put wc-in/* /user/bigdata/wc-in
```

```
bigdata@bigdata:~/mapreduce$ hdfs dfs -put wc-in/* /user/bigdata/wc-in
bigdata@bigdata:~/mapreduce$
```

5

5. Ejecución del jar:

```
$ hadoop jar wc.jar WordCount wc-in wc-out
```

```
bigdata@bigdata:~/mapreduce$ hadoop jar wc.jar WordCount wc-in wc-out
17/08/12 19:58:17 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
17/08/12 19:58:17 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
17/08/12 19:58:18 INFO mapreduce.JobSubmitter: number of splits:2
17/08/12 19:58:18 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1502490164835_0015
17/08/12 19:58:19 INFO mapreduce.Job: The url to track the job: http://bigdata:8088/proxy/application_1502490164835_0015/
17/08/12 19:58:19 INFO mapreduce.Job:  waiting for 0 sec for map tasks
17/08/12 19:58:27 INFO mapreduce.Job: Job job_1502490164835_0015 running in uber mode : false
17/08/12 19:58:27 INFO mapreduce.Job: map 0% reduce 0%
17/08/12 19:58:31 INFO mapreduce.Job: map 100% reduce 0%
17/08/12 19:58:42 INFO mapreduce.Job: map 100% reduce 100%
17/08/12 19:58:43 INFO mapreduce.Job: Job job_1502490164835_0015 completed successfully
17/08/12 19:58:43 INFO mapreduce.Job: Counters
    FILE: Number of bytes read=92
    FILE: Number of bytes written=408983
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=273
    HDFS: Number of bytes written=50
    HDFS: Number of read operations=9
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=2
Job Counters
```

```
$ hdfs dfs -cat /user/bigdata/wc-out/part-r-00000
```

```
bigdata@bigdata:~/mapreduce$ hdfs dfs -cat /user/bigdata/wc-out/part-r-00000
Bye      1
Goodbye 1
Hadoop   2
Hello    2
World.   2
bigdata@bigdata:~/mapreduce$
```

11.7. Interfaz web

Por último, se comprobará el correcto funcionamiento de la interfaz web. Para ello, se siguen los siguientes pasos.

1

1. Arrancar el jobhistory:

```
$ sbin/mr-jobhistory-daemon.sh start historyserver
```

2

2. Acceder mediante un navegador a la url <http://localhost:19888>

Submit Time	Start Time	Finish Time	job ID	Name	User	Queue	State	Maps Total	Maps Completed	Reduces Total	Reduces Completed
2014-12-23 16:26:38 CET	2014-12-23 16:26:49 CET	2014-12-23 16:29:08 CET	job_1419258670929_0004	word count	bigdata	default	SUCCEEDED	2	2	1	1

11.8. Ejemplo II

Este ejemplo realiza la suma del número de líneas que contienen los ficheros que recibe como entrada:

Para ver el código que necesitas, pulsa [aquí](#).

1

A continuación se exponen los pasos para ejecutar el ejemplo. Lo primero que debemos hacer es generar el fichero .java "LineCount.java" como se puede apreciar en la siguiente captura de pantalla.

sudo nano LineCount.java

```
bigdata@bigdata:~/mapreduce$ sudo nano LineCount.java
```

Dentro de este fichero deberemos pegar el código del ejemplo.

```
import java.io.IOException;
import java.util.*;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;
public class LineCount {
    public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable> {
        private final static IntWritable uno= new IntWritable(1);
        private Text word = new Text("Total Líneas");
        public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
            output.collect(word, uno);
        }
    }
    public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {
        public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
            int sum = 0;
            while (values.hasNext()) {
                sum += values.next().get();
            }
            output.collect(key, new IntWritable(sum));
        }
    }
    public static void main(String[] args) throws Exception {
        JobConf conf = new JobConf(LineCount.class);
        conf.setJobName('LineCount');
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);
        conf.setMapperClass(Map.class);
        conf.setCombinerClass(Reduce.class);
        conf.setReducerClass(Reduce.class);
        conf.setInputFormat(TextInputFormat.class);
        conf.setOutputFormat(TextOutputFormat.class);
        FileInputFormat.setInputPaths(conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(conf, new Path(args[1]));
        JobClient.runJob(conf);
    }
}
```

2

Una vez creado "LineCount.java", procederemos a crear un directorio llamado "line-count-in" ejecutando el comando siguiente:

sudo mkdir line-count-in

sudo chmod -R 777 line-count-in

```
bigdata@bigdata:~/mapreduce$ sudo mkdir line-count-in
bigdata@bigdata:~/mapreduce$ sudo chmod -R 777 line-count-in
```

Una vez creado el directorio, pasaremos a crear los ficheros .txt (a, b, c, d) y su contenido ejecutando los siguientes comandos:

```
sudo echo -e "uno\ndos\ntres" > line-count-in/a.txt  
sudo echo -e "dos\ndos\ndos" > line-count-in/b.txt  
sudo echo -e "tres\ntres\ntres" > line-count-in/c.txt  
sudo echo -e "tres\nncuatro\nncinco\nnseis" > line-count-in/d.txt
```

```
bigdata@bigdata:~/mapreduce$ sudo echo -e "uno\ndos\ntres" > line-count-in/a.txt  
bigdata@bigdata:~/mapreduce$ sudo echo -e "dos\ndos\ndos" > line-count-in/b.txt  
bigdata@bigdata:~/mapreduce$ sudo echo -e "tres\ntres\ntres" > line-count-in/c.txt  
bigdata@bigdata:~/mapreduce$ sudo echo -e "tres\nncuatro\nncinco\nnseis" > line-count-in/d.txt
```

3

Para comprobar que hemos creado correctamente los ficheros, ejecutaremos los comandos siguientes comprobando la salida:

```
cat line-count-in/a.txt
```

```
cat line-count-in/b.txt
```

```
cat line-count-in/c.txt
```

```
cat line-count-in/d.txt
```

```
bigdata@bigdata:~/mapreduce$ cat line-count-in/a.txt  
uno  
dos  
tres  
bigdata@bigdata:~/mapreduce$ cat line-count-in/b.txt  
dos  
dos  
dos  
bigdata@bigdata:~/mapreduce$ cat line-count-in/c.txt  
tres  
tres  
tres  
bigdata@bigdata:~/mapreduce$ cat line-count-in/d.txt  
tres  
cuatro  
cinco  
seis
```

Una vez tengamos todos los ficheros correctamente creados, los subiremos a HDFS con el siguiente comando:

```
hdfs dfs -put line-count-in/ /user/bigdata/
```

```
bigdata@bigdata:~/mapreduce$ hdfs dfs -put line-count-in/ /user/bigdata/
```

4

Para comprobar que se subieron correctamente, ejecutaremos:

```
hdfs dfs -ls /user/bigdata/line-count-in
```

```
bigdata@bigdata:~/mapreduce$ hdfs dfs -ls /user/bigdata/line-count-in
Found 4 items
-rw-r--r-- 1 bigdata supergroup          13 2017-11-19 18:06 /user/bigdata/line-count-in/a.txt
-rw-r--r-- 1 bigdata supergroup          12 2017-11-19 18:06 /user/bigdata/line-count-in/b.txt
-rw-r--r-- 1 bigdata supergroup          15 2017-11-19 18:06 /user/bigdata/line-count-in/c.txt
-rw-r--r-- 1 bigdata supergroup          23 2017-11-19 18:06 /user/bigdata/line-count-in/d.txt
bigdata@bigdata:~/mapreduce$
```

Como se puede apreciar en la siguiente imagen, antes de ejecutar nuestro ejemplo debemos obtener nuestra aplicación en un .class ejecutando el siguiente comando:

```
sudo javac -classpath $HADOOP_HOME/share/hadoop/common/hadoop-common-2.8.0.jar:$HADOOP_HOME/share/hadoop/common/lib/hadoop-annotations-2.8.0.jar:$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-core-2.8.0.jar /home/bigdata/mapreduce/LineCount.java
```

```
bigdata@bigdata:~/mapreduce$ sudo javac -classpath $HADOOP_HOME/share/hadoop/common/hadoop-common-2.8.0.jar:$HADOOP_HOME/share/hadoop/common/lib/hadoop-annotations-2.8.0.jar:$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-core-2.8.0.jar /home/bigdata/mapreduce/LineCount.java
bigdata@bigdata:~/mapreduce$
```

Una vez obtenemos el .class, generaremos el .jar con el comando siguiente:

```
sudo jar cf LineCount.jar LineCount*.class
```

```
bigdata@bigdata:~/mapreduce$ sudo jar cf LineCount.jar LineCount*.class
bigdata@bigdata:~/mapreduce$
```

Llegados a este punto, ya estamos listos para ejecutar nuestro ejemplo. Para ello lanzaremos el siguiente comando:

```
hadoop jar /home/bigdata/mapreduce/LineCount.jar LineCount line-count-in/ line-count-out
```

```
bigdata@bigdata:~/mapreduce$ hadoop jar /home/bigdata/mapreduce/LineCount.jar LineCount line-count-in/ line-count-out
```

```
bigdata@bigdata:~/mapreduce$ hadoop jar /home/bigdata/mapreduce/LineCount.jar LineCount line-count-in/ line-count-out
17/11/19 18:14:03 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8082
17/11/19 18:14:04 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8082
17/11/19 18:14:05 WARN mapred.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
17/11/19 18:14:05 INFO mapred.FileInputFormat: Total input files to process : 4
17/11/19 18:14:06 INFO mapred.FileInputFormat: Number of splits: 4
17/11/19 18:14:06 INFO mapreduce.JobSubmissionHandler: Submitting tokens for job: job_1504456518480_0002
17/11/19 18:14:06 INFO impl.YarnClientImpl: Submitted application application_1504456518480_0002
17/11/19 18:14:06 INFO mapreduce.Job: The url to track the job: http://bigdata:8088/proxy/application_1504456518480_0002/
17/11/19 18:14:06 INFO mapreduce.Job: Running job: job_1504456518480_0002
17/11/19 18:14:06 INFO mapreduce.Job: Job job_1504456518480_0002 running in uber mode : false
17/11/19 18:14:16 INFO mapreduce.Job: map 100% reduce 0%
17/11/19 18:14:35 INFO mapreduce.Job: map 100% reduce 0%
17/11/19 18:14:42 INFO mapreduce.Job: map 100% reduce 100%
17/11/19 18:14:42 INFO mapreduce.Job: Job job_1504456518480_0002 completed successfully
17/11/19 18:14:42 INFO mapreduce.Job: Counters: 49
File System Counters
  FILE: Number of bytes read=82
  FILE: Number of bytes written=683142
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=47
  HDFS: Number of bytes written=16
  HDFS: Number of read operations=15
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=2
Job Counters
  Launched map tasks=4
  Launched reduce tasks=1
  Data-local map tasks=4
  Total time spent by all maps in occupied slots (ms)=68138
  Total time spent by all reduces in occupied slots (ms)=4089
  Total time spent by all map tasks (ms)=68138
  Total time spent by all reduce tasks (ms)=4089
  Total vcore-milliseconds taken by all map tasks=8138
  Total vcore-milliseconds taken by all reduce tasks=4089
  Total vcore-milliseconds taken by all tasks=8138
Map-Reduce Framework
  Map input records=13
  Map output records=13
  Map output bytes=221
  Map output materialized bytes=100
  Input split bytes=52
  Combine input records=13
  Combine output records=4
  Reduce input groups=1
  Reduce shuffle bytes=100
  Reduce input records=4
  Reduce output records=1
  Spilled Records=8
  Shuffled Maps=1
  Failed Shuffles=0
  Merged Map outputs=4
  GC time elapsed (ms)=1087
  CPU time spent (ms)=1840
  Physical memory (bytes) snapshot=1000497152
  Virtual memory (bytes) snapshot=694208000
  Total committed heap usage (bytes)=700776448
Shuffle Bytes=0
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0
  File Input Format Counters
    Bytes Read=683
  File Output Format Counters
    Bytes Written=16
bigdata@bigdata:~/mapreduce$
```

6

Nuestro ejemplo generará dos ficheros, para comprobarlo ejecutaremos:

```
hdfs dfs -ls /user/bigdata/line-count-out
```

```
bigdata@bigdata:~/mapreduce$ hdfs dfs -ls /user/bigdata/line-count-out
Found 2 items
-rw-r--r-- 1 bigdata supergroup          0 2017-11-19 18:14 /user/bigdata/line-count-out/_SUCCESS
-rw-r--r-- 1 bigdata supergroup        16 2017-11-19 18:14 /user/bigdata/line-count-out/part-00000
bigdata@bigdata:~/mapreduce$
```

El fichero que contiene el resultado es "part-00000", para consultar su contenido ejecutaremos:

```
hdfs dfs -cat /user/bigdata/line-count-out/part-00000
```

```
bigdata@bigdata:~/mapreduce$ hdfs dfs -cat /user/bigdata/line-count-out/part-00000
Total Lineas   13
bigdata@bigdata:~/mapreduce$
```

11.9. Ejemplo de MapReduce con Python: Mapper

1

Creamos una carpeta en la que almacenaremos los ficheros de mapeo y reducción:

```
mkdir /home/bigdata/ejemplosMapReduce
```

```
mkdir /home/bigdata/ejemplosMapReduce/python
```

```
cd /home/bigdata/ejemplosMapReduce/python
```

```
bigdata@bigdata:~$ mkdir /home/bigdata/ejemplosMapReduce
bigdata@bigdata:~$ mkdir /home/bigdata/ejemplosMapReduce/python
bigdata@bigdata:~$ cd /home/bigdata/ejemplosMapReduce/python
bigdata@bigdata:~/ejemplosMapReduce/python$
```

```
sudo nano mapper.py
```

```
bigdata@bigdata:~/ejemplosMapReduce/python$ sudo nano mapper.py
```

Para ver el código que necesitas, pulsa [aquí](#).

2

```
#!/usr/bin/env python
import sys
# input comes from STDIN (standard input)
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    words = line.split()
    # increase counters
    for word in words:
        # write the results to STDOUT (standard output);
        # what we output here will be the input for the
        # Reduce step, i.e. the input for reducer.py
        # tab-delimited; the trivial word count is 1
        print '%s\t%s' % (word, 1)
```

```
#!/usr/bin/env python
import sys
# input comes from STDIN (standard input)
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    words = line.split()
    # increase counters
    for word in words:
        # write the results to STDOUT (standard output);
        # what we output here will be the input for the
        # Reduce step, i.e. the input for reducer.py
        # tab-delimited; the trivial word count is 1
        print '%s\t%s' % (word, 1)
```

3

Asignamos los permisos de ejecución con el siguiente comando:

```
sudo chmod +x /home/bigdata/ejemplosMapReduce/python/mapper.py
```

```
bigdata@bigdata:~/ejemplosMapReduce/python$ sudo chmod +x /home/bigdata/ejemplosMapReduce/python/mapper.py
```

```
sudo nano reducer.py
```

```
bigdata@bigdata:~/ejemplosMapReduce/python$ sudo nano reducer.py
```

Para ver el código que necesitas, pulsa [aquí](#).

```
GNU nano 2.7.4                               Archivo: reducer.py                                Modificado
#!/usr/bin/env python
from operator import itemgetter
import sys
current_word = None
current_count = 0
word = None
# input comes from STDIN
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # parse the input we got from mapper.py
    word, count = line.split('\t', 1)
    # convert count (currently a string) to int
    try:
        count = int(count)
    except ValueError:
        # count was not a number, so silently ignore/discard this line
        continue
    # this IF-switch only works because Hadoop sorts map output by key (here: word) before it is passed to the
    if current_word == word:
        current_count += count
    else:
        if current_word:
            # write result to STDOUT
            print '%s\t%s' % (current_word, current_count)
        current_count = count
        current_word = word
# do not forget to output the last word if needed!
if current_word == word:
    print '%s\t%s' % (current_word, current_count)
```

4

Asignamos los permisos de ejecución con el siguiente comando:

```
sudo chmod +x /home/bigdata/ejemplosMapReduce/python/reducer.py
```

```
bigdata@bigdata:~/ejemplosMapReduce/python$ sudo chmod +x /home/bigdata/ejemplosMapReduce/python/reducer.py
```

Comprobamos que funcionan con un ejemplo sencillo al que concatenamos ordenación:

```
echo "1 2 22 3 2 333 4 1" | /home/bigdata/ejemplosMapReduce/python/mapper.py
```

```
echo "1 2 22 3 2 333 4 1" | /home/bigdata/ejemplosMapReduce/python/mapper.py | sort -k1,1
```

```
bigdata@bigdata:~/ejemplosMapReduce/python$ echo "1 2 22 3 2 333 4 1" | /home/bigdata/ejemplosMapReduce/python/mapper.py
1      1
2      1
22     1
3      1
2      1
333    1
4      1
1      1
bigdata@bigdata:~/ejemplosMapReduce/python$ echo "1 2 22 3 2 333 4 1" | /home/bigdata/ejemplosMapReduce/python/mapper.py | sort -k1,1
1      1
1      1
2      1
2      1
22     1
3      1
333    1
4      1
```

5

Creamos unos ficheros y los subimos a una nueva estructura en HDFS:

```
mkdir /home/bigdata/ejemplosMapReduce/python/wc-in-local
```

```
echo "Hello World. Bye World." > /home/bigdata/ejemplosMapReduce/python/wc-in-local/a.txt
```

```
echo "Hello Hadoop Goodbye Hadoop" > /home/bigdata/ejemplosMapReduce/python/wc-in-local/b.txt
```

```
bigdata@bigdata:~/ejemplosMapReduce/python$ mkdir /home/bigdata/ejemplosMapReduce/python/wc-in-local
bigdata@bigdata:~/ejemplosMapReduce/python$ echo "Hello World. Bye World." > /home/bigdata/ejemplosMapReduce/python/wc-in-local/a.txt
bigdata@bigdata:~/ejemplosMapReduce/python$ echo "Hello Hadoop Goodbye Hadoop" > /home/bigdata/ejemplosMapReduce/python/wc-in-local/b.txt
```

Creamos la nueva carpeta en HDFS:

```
hdfs dfs -mkdir -p /user/bigdata/wc-in-python
```

```
bigdata@bigdata:~/ejemplosMapReduce/python$ hdfs dfs -mkdir -p /user/bigdata/wc-in-python
```

6

Subimos los ficheros:

```
hdfs dfs -put /home/bigdata/ejemplosMapReduce/python/wc-in-local/* /user/bigdata/wc-in-python
```

```
bigdata@bigdata:~/ejemplosMapReduce/python$ hdfs dfs -put /home/bigdata/ejemplosMapReduce/python/wc-in-local/* /user/bigdata/wc-in-python
```

```
hadoop jar /home/bigdata/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.8.0.jar -mapper /home/bigdata/ejemplosMapReduce/python/mapper.py -reducer /home/bigdata/ejemplosMapReduce/python/reducer.py -input /user/bigdata/wc-in-python/* -output /user/bigdata/wc-output-python
```

```
bigdata@bigdata:~/ejemplosMapReduce/python$ hadoop jar /home/bigdata/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.8.0.jar -mapper /home/bigdata/ejemplosMapReduce/python/mapper.py -reducer /home/bigdata/ejemplosMapReduce/python/reducer.py -input /user/bigdata/wc-in-python/* -output /user/bigdata/wc-output-python
```

```
packageJobJar: [/tmp/hadoop-unjar2723194120675579548/] [] /tmp/streamjob4759422607733764516.jar tmpDir=null
17/11/19 18:27:41 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
17/11/19 18:28:23 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
17/11/19 18:28:24 INFO mapred.FileInputFormat: Total input files to process : 2
17/11/19 18:28:24 INFO mapreduce.JobSubmitter: number of splits:2
17/11/19 18:28:24 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1504456518480_0004
17/11/19 18:28:25 INFO impl.YarnClientImpl: Submitted application application_1504456518480_0004
17/11/19 18:28:25 INFO mapreduce.Job: The url to track the job: http://bigdata:8088/proxy/application_1504456518480_0004/
17/11/19 18:28:25 INFO mapreduce.Job: Running job: job_1504456518480_0004
17/11/19 18:28:34 INFO mapreduce.Job: Job job_1504456518480_0004 running in uber mode : false
17/11/19 18:28:34 INFO mapreduce.Job: map 0% reduce 0%
17/11/19 18:28:46 INFO mapreduce.Job: map 100% reduce 0%
17/11/19 18:28:54 INFO mapreduce.Job: map 100% reduce 100%
17/11/19 18:28:55 INFO mapreduce.Job: Job job_1504456518480_0004 completed successfully
17/11/19 18:28:55 INFO mapreduce.Job: Counters: 49
File System Counters
    FILE: Number of bytes read=90
    FILE: Number of bytes written=415490
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=262
    HDFS: Number of bytes written=42
    HDFS: Number of read operations=9
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=2
Job Counters
    Launched map tasks=2
    Launched reduce tasks=1
    Data-local map tasks=2
    Total time spent by all maps in occupied slots (ms)=20272
    Total time spent by all reduces in occupied slots (ms)=4617
    Total time spent by all map tasks (ms)=20272
    Total time spent by all reduce tasks (ms)=4617
    Total vcore-milliseconds taken by all map tasks=20272
    Total vcore-milliseconds taken by all reduce tasks=4617
    Total megabyte-milliseconds taken by all map tasks=20758528
    Total Megabyte-milliseconds taken by all reduce tasks=4727808
```

```

Map-Reduce Framework
  Map input records=2
  Map output records=8
  Map output bytes=68
  Map output materialized bytes=96
  Input split bytes=210
  Combine input records=0
  Combine output records=0
  Reduce input groups=5
  Reduce shuffle bytes=96
  Reduce input records=8
  Reduce output records=5
  Spilled Records=16
  Shuffled Maps =2
  Failed Shuffles=0
  Merged Map outputs=2
  GC time elapsed (ms)=342
  CPU time spent (ns)=1390
  Physical memory (bytes) snapshot=562249728
  Virtual memory (bytes) snapshot=5823672320
  Total committed heap usage (bytes)=381878272
Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=52
File Output Format Counters
  Bytes Written=42
17/11/19 18:28:55 INFO streaming.StreamJob: Output directory: /user/bigdata/wc-output-python
bigdata@bigdata:~/ejemplosMapReduce/python$ 

```

hdfs dfs -ls /user/bigdata/wc-output-python

```

bigdata@bigdata:~/ejemplosMapReduce/python$ hdfs dfs -ls /user/bigdata/wc-output-python
Found 2 items
-rw-r--r-- 1 bigdata supergroup      0 2017-11-19 18:28 /user/bigdata/wc-output-python/_SUCCESS
-rw-r--r-- 1 bigdata supergroup    42 2017-11-19 18:28 /user/bigdata/wc-output-python/part-00000
bigdata@bigdata:~/ejemplosMapReduce/python$ 

```

*hdfs dfs -cat /user/bigdata/wc-output-python/**

```

bigdata@bigdata:~/ejemplosMapReduce/python$ hdfs dfs -cat /user/bigdata/wc-output-python/*
Bye      1
Goodbye 1
Hadoop   2
Hello    2
World.   2

```

XII. Resumen



En esta unidad hemos estudiado qué es Hadoop, algo de su historia y por qué es un *software* tan vinculado al mundo del Big Data. Asimismo, se han repasado sus componentes principales:

- ➔ HDFS: sistema de archivos distribuido que utiliza Hadoop para almacenar la información.
- ➔ YARN/MapReduce: componente responsable de la gestión y distribución del trabajo en el clúster.

Por último, se han hecho ejercicios simples para familiarizarnos con el entorno en las máquinas virtuales facilitadas al efecto.

Ejercicios

Ejercicio 1



Objetivo: poner en práctica los conocimientos adquiridos sobre la herramienta para resolver el ejercicio.

Partiendo de las letras de las siguientes canciones:

Nine Inch Nails, "March of the Pigs"

step right up march push

crawl right up on your knees

please greed feed (no time to hesitate)

I want a little bit I want a piece of it I think he's losing it

I want to watch it come down

don't like the look of it don't like the taste of it don't like the smell of it

I want to watch it come down

all the pigs are all lined up

I give you all that you want

take the skin and peel it back

now doesn't that make you feel better?

shove it up inside surprise! lies

stains like the blood on your teeth

bite chew suck away the tender parts

I want to break it up I want to smash it up I want to fuck it up

I want to watch it come down

maybe afraid of it let's discredit it let's pick away at it

I want to watch it come down

now doesn't that make you feel better?

the pigs have won tonight

now they can all sleep soundly

and everything is all right

Pink Floyd, "Pigs (Three Different Ones)"

Big man, pig man, ha ha, charade you are

You well heeled big wheel, ha ha, charade you are

And when your hand is on your heart

You're nearly a good laugh

Almost a joker With your head down in the pig bin

Saying "keep on digging" Pig stain on your fat chin

What do you hope to find?

When you're down in the pig mine

You're nearly a laugh You're nearly a laugh

But you're really a cry.

Bus stop rat bag, ha ha, charade you are

You fucked up old hag, ha ha, charade you are

You radiate cold shafts of broken glass

You're nearly a good laugh Almost worth a quick grin

You like the feel of steel You're hot stuff with a hat pin

And good fun with a hand gun

You're nearly a laugh You're nearly a laugh

But you're really a cry.

Hey you Whitehouse, ha ha, charade you are

You house proud town mouse, ha ha, charade you are

You're trying to keep our feelings off the street

You're nearly a real treat All tight lips and cold feet

And do you feel abused?!.....!.....!

You gotta stem the evil tide

And keep it all on the inside

Mary you're nearly a treat Mary you're nearly a treat

But you're really a cry.

Dave Matthews Band, "Pig"

Isn't it strange

How we move our lives for another day

Like skipping a beat

What if a great wave should wash us all away

Just thinking out loud

Don't mean to dwell on this dying thing

But looking at blood

It's alive right now

Deep and sweet within

Pouring through our veins

Intoxicate moving wine to tears

Drinking it deep

Then an evening spent dancing

It's you and me

This love will open our world

From the dark side we can see a glow of something bright

There's much more than we see here

Don't burn the day away

Is this not enough

This blessed sip of life

Is it not enough

Staring down at the ground

Oh then complain and pray more from above

Greedy little pig

Stop just watch your world trickle away

Oh it's your problem now

It'll all be dead and gone in a few short years

Just love will open our eyes

Just love will put the hope in our minds

Much more than we could ever know

Don't burn the day away

Come sister my brother

Shake up your bones shake up your feet

I'm saying open up

And let the rain come pouring in

Wash out this tired notion

That the best is yet to come

But while you're dancing on the ground

Don't think of when you're gone

Love love what more is there

We need the light of love in here

Don't beat your head

Dry your eyes

Let the love in there

There are bad times

But that's ok

Just look for love in it

Don't burn the day away

Look

Here are we

On this starry night staring into space

And I must say

I feel as small as dust

Lying down here

What point could there be troubling

Head down wondering what will become of me

Why concern we cannot see

But no reason to abandon it

Time is short but that's all right

Maybe I'll go in the middle of the night

Take your hands from your eyes, my love

Everything must end some time

Don't burn the day away

Come sister my brother

Shake up your bones shake up your feet

I'm saying open up

And let the rain come flooding in

Wash out this tired notion

That the best is yet to come

But while you're dancing on the ground

Don't think of when you're gone

Love love what more is there

We need the light of love in here

Don't beat your head

Dry your eyes

Let the love in there

There are bad times

But that's ok

Just look for love in it

Se pide

1. Crear ficheros de texto en la máquina Ubuntu.
2. Subir los ficheros a HDFS.
3. Ejecutar un job wordcount y resolver las siguientes cuestiones:
 - a. Acceder al jobhistory e indicar los valores de "Status" y "Map Total" del job con el nombre "word count".
 - b. Palabra que se repite más veces.
 - c. Palabra que se repite seis veces.
 - d. Cuántas veces aparece la palabra "eyes".

Solución

1. Crear ficheros de texto en la máquina Ubuntu

- ➔ Crear un directorio nuevo para la práctica 1:

```
bigdata@bigdata:~/hadoop$ mkdir practica1
bigdata@bigdata:~/hadoop$ ls
bin EjemploMerge.txt ejemplo.txt ejercicios etc include lib libexec LICENSE.txt logs NOTICE.txt practica1 README.txt sbin share
bigdata@bigdata:~/hadoop$
```

- ➔ Crear los ficheros con la letra de las canciones:

```
bigdata@bigdata:~/hadoop/practica1$ sudo nano cancion1.txt
bigdata@bigdata:~/hadoop/practica1$ sudo nano cancion2.txt
bigdata@bigdata:~/hadoop/practica1$ sudo nano cancion3.txt
bigdata@bigdata:~/hadoop/practica1$
```

```
GNU nano 2.2.6                                         Archivo: cancion1.txt

NINE INCH NAILS LYRICS "March Of The Pigs"

step right up march push
crawl right up on your knees
please greed feed (no time to hesitate)
I want a little bit I want a piece of it I think he's losing it
I want to watch it come down
don't like the look of it don't like the taste of it don't like the smell of it
I want to watch it come down
all the pigs are all lined up
I give you all that you want
take the skin and peel it back
now doesn't that make you feel better?
shove it up inside surprise! lies
stains like the blood on your teeth
bite chew suck away the tender parts
I want to break it up I want to smash it up I want to fuck it up
I want to watch it come down
maybe afraid of it let's discredit it let's pick away at it
I want to watch it come down
now doesn't that make you feel better?
the pigs have won tonight
now they can all sleep soundly
and everything is all right■
```

```
GNU nano 2.2.6                                         Archivo: cancion2.txt

PINK FLOYD "Pigs (Three Different Ones)"
Big man, pig man, ha ha, charade you are
You well heeled big wheel, ha ha, charade you are
And when your hand is on your heart
You're nearly a good laugh
Almost a joker
With your head down in the pig bin
Saying "keep on digging"
Pig stain on your fat chin
What do you hope to find?
When you're down in the pig mine
You're nearly a laugh
You're nearly a laugh
But you're really a cry.
Bus stop rat bag, ha ha, charade you are
You fucked up old hag, ha ha, charade you are
You radiate cold shafts of broken glass
You're nearly a good laugh
Almost worth a quick grin
You like the feel of steel
You're hot stuff with a hat pin
And good fun with a hand gun
You're nearly a laugh
You're nearly a laugh
But you're really a cry.
Hey you Whitehouse, ha ha, charade you are
You house proud town mouse, ha ha, charade you are
You're trying to keep our feelings off the street
```

```
GNU nano 2.2.6                                         Archivo: cancion3.txt

DAVE MATTHEWS BAND "Pig"
Isn't it strange
How we move our lives for another day
Like skipping a beat
What if a great wave should wash us all away
Just thinking out loud
Don't mean to dwell on this dying thing
But looking at blood
It's alive right now
Deep and sweet within
Pouring through our veins
Intoxicate moving wine to tears
Drinking it deep
Then an evening spent dancing
It's you and me
This love will open our world
From the dark side we can see a glow of something bright
There's much more than we see here
Don't burn the day away
Is this not enough
This blessed sip of life
Is it not enough
Staring down at the ground
Oh then complain and pray more from above
Greedy little pig
Stop just watch your world trickle away
Oh it's your problem now
```

2. Subir los ficheros a HDFS

- Añadir los ficheros a HDFS:

```
hdfs dfs -copyFromLocal practica1 /practica1
```

- Comprobar si se han añadido correctamente:

```
hadoop fs -ls /practica1
```

```
bigdata@bigdata:~/hadoop$ hdfs dfs -copyFromLocal practica1 /practica1
16/03/06 12:11:56 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
bigdata@bigdata:~/hadoop$ hadoop fs -ls /
16/03/06 12:12:29 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 5 items
drwxr-xr-x  - bigdata supergroup      0 2016-03-02 21:47 /ejercicio1
drwxr-xr-x  - bigdata supergroup      0 2016-03-02 21:51 /out
drwxr-xr-x  - bigdata supergroup      0 2016-03-06 12:12 /practica1
drwx-----  - bigdata supergroup      0 2016-03-02 21:48 /tmp
drwxr-xr-x  - bigdata supergroup      0 2016-03-05 12:26 /user
bigdata@bigdata:~/hadoop$ hadoop fs -ls /practica1
16/03/06 12:12:48 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 3 items
-rw-r--r--  1 bigdata supergroup     883 2016-03-06 12:12 /practica1/cancion1.txt
-rw-r--r--  1 bigdata supergroup    1149 2016-03-06 12:12 /practica1/cancion2.txt
-rw-r--r--  1 bigdata supergroup   2287 2016-03-06 12:12 /practica1/cancion3.txt
```

3. Ejecutar un wordcount y resolver las siguientes cuestiones

- Ejecutar wordcount:

```
hadoop jar /home/bigdata/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.8.0.jar wordcount /practica1 /outcanciones
```

```
bigdata@bigdata:~/hadoop$ hadoop jar /home/bigdata/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.1.jar wordcount /practica1 /outcanciones
16/03/06 12:20:13 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
16/03/06 12:20:18 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
16/03/06 12:20:21 INFO input.FileInputFormat: Total input paths to process : 3
16/03/06 12:20:21 INFO mapreduce.JobSubmitter: number of splits:3
16/03/06 12:20:23 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1457173731478_0004
16/03/06 12:20:24 INFO impl.YarnClientImpl: Submitted application application_1457173731478_0004
16/03/06 12:20:24 INFO mapreduce.Job: The url to track the job: http://bigdata:8088/proxy/application_1457173731478_0004/
16/03/06 12:20:24 INFO mapreduce.Job: Running job: job_1457173731478_0004
16/03/06 12:20:52 INFO mapreduce.Job: Job job_1457173731478_0004 running in uber mode : false
16/03/06 12:21:51 INFO mapreduce.Job: map 0% reduce 0%
16/03/06 12:21:54 INFO mapreduce.Job: map 100% reduce 0%
16/03/06 12:22:14 INFO mapreduce.Job: map 100% reduce 100%
16/03/06 12:22:15 INFO mapreduce.Job: Job job_1457173731478_0004 completed successfully
16/03/06 12:22:16 INFO mapreduce.Job: Counters
      File System Counters
          FILE: Number of bytes read=4806
          FILE: Number of bytes written=472145
          FILE: Number of read operations=0
          FILE: Number of large read operations=0
          FILE: Number of write operations=0
          HDFS: Number of bytes read=4646
          HDFS: Number of bytes written=2789
          HDFS: Number of read operations=12
          HDFS: Number of large read operations=0
          HDFS: Number of write operations=2
```

```
Map input records=141
Map output records=860
Map output bytes=7680
Map output materialized bytes=4818
Input split bytes=327
Combine input records=860
Combine output records=419
Reduce input groups=363
Reduce shuffle bytes=4818
Reduce input records=419
Reduce output records=363
Spilled Records=838
Shuffled Maps =3
Failed Shuffles=0
Merged Map outputs=3
GC time elapsed (ms)=1944
CPU time spent (ms)=7380
Physical memory (bytes) snapshot=781418496
Virtual memory (bytes) snapshot=3198332928
Total committed heap usage (bytes)=540868608
Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=4319
File Output Format Counters
  Bytes Written=2789
```

Procesamiento de datos con Hadoop

1

A. Acceder al jobhistory e indicar los valores de "Status" y "Map Total" del job con el nombre "word count":

- ➔ Acceder al jobhistoy:

The screenshot shows the Hadoop Job History UI at localhost:8088/cluster. The main page displays a table of jobs. One job, **application_1457173731478_0004**, is highlighted with a red box. This job was submitted by **bigdata** and has a status of **SUCCEEDED**. The table includes columns for ID, User, Name, Application Type, Queue, Start Time, Finish Time, State, Final Status, Progress, and Tracking UI.

Job Overview details for job_1457173731478_0004:

- Job Name: word count
- User Name: bigdata
- Queue: default
- Status: SUCCEEDED**
- Uberized: false
- Submitted: Sun Mar 06 12:20:24 CET 2016
- Started: Sun Mar 06 12:20:51 CET 2016
- Finished: Sun Mar 06 12:22:14 CET 2016
- Elapsed: 1mins, 22sec

Diagnostics:

Average Map Time	57sec
Average Shuffle Time	16sec
Average Merge Time	0sec
Average Reduce Time	2sec

ApplicationMaster tasks:

Attempt Number	Start Time	Node	Logs
1	Sun Mar 06 12:20:28 CET 2016	bigdata:8042	logs

Task Type distribution:

Task Type	Total	Complete
Map	3	3
Reduce	1	1
Attempts	Failed	Killed
Maps	0	0
Reduces	0	1

- ➔ Comprobar que el Status es SUCCEEDED y el Map Total es 3.

2

B. Palabra que se repite más veces:

- ➔ Revisar la salida:

```
bigdata@bigdata:~/hadoop$ bin/hdfs dfs -cat /outcanciones/*
16/03/06 12:34:47 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
"March" 1
"Pig" 1
"Pigs" 1
"keep" 1
(Three 1
(no 1
.....!.....!.....!.....! 1
All 1
Almost 2
And 7
BAND 1
Big 1
Bus 1
But 9
Come 2
DAVE 1
Deep 1
Different 1
Don't 9
Drinking 1
Dry 2
Everything 1
FLOYD 1
From 1
Greedy 1
Head 1
```

Para obtener la palabra que más se repite, ordenar la salida por la segunda columna numérica en orden descendente y nos quedamos con la primera:

```
hdfs dfs -cat /outcanciones/* | sort -k2nr | head -1
```

```
bigdata@bigdata:~/hadoop$ hdfs dfs -cat /outcanciones/* | sort -k2nr | head -1
16/03/06 13:12:20 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
the 33
```

El resultado es la palabra “**the**”, con **33 apariciones**.

3

C. Palabra que se repite seis veces

- ➔ Para obtener las palabras que se repiten seis veces, usamos el comando grep:

```
hdfs dfs -cat /outcanciones/* | grep '6'
```

```
bigdata@bigdata:~/hadoop$ hdfs dfs -cat /outcanciones/* | grep '6'
16/03/06 13:31:04 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform
ble
You 6
charade 6
ha 6
ha, 6
laugh 6
our 6
we 6
```

Obtenemos las palabras: “You”, “charade”, “ha”, “laugh”, “our” y “we”.

4

D. Cuántas veces aparece la palabra "eyes"

- Obtenemos las veces que aparece la palabra "eyes" con el siguiente comando:

```
hdfs dfs -cat /outcanciones/* | grep '\<eyes.*\>'
```

```
bigdata@bigdata:~/hadoop$ hdfs dfs -cat /outcanciones/* | grep '\<eyes.*\>'  
16/03/06 13:20:38 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable  
eyes      3  
eyes,     1
```

Vemos que aparece **cuatro veces**.

Ejercicio 2



Objetivo: Poner en práctica los conocimientos adquiridos sobre la herramienta para resolver el ejercicio.

A

Ejecutar el programa sudoku con los siguientes parámetros:

- Programa: sudoku
- Path: fichero



En la ruta del fichero habrá que indicar la ruta en la que se encuentre un fichero que tendrá la siguiente información:

```
? 4 6 3 ? ? 8 2 5
? ? ? ? ? 2 4 ? ?
8 2 ? ? 6 ? ? ? 7 ?
7 ? ? 4 ? ? ? ? 2
3 8 ? ? ? ? 6 ? ? 9
? 6 ? ? 2 8 ? ? ? ?
? 7 ? ? ? 5 ? 3 ?
5 3 8 ? ? 7 ? ? ?
? 9 ? ? ? ? ? ? 6
```

Indicar todos los resultados del sudoku de ejemplo.

B

OPCIONAL: Dado el siguiente sudoku, obtener el valor de las siguientes filas y columnas:

- 1,8
- 7,5

	6	8		2	14	15		5		16	10				
	5		12			9		10		3		7			
9	15			8		12		2				13		5	
13			4		6			7							
2	10		14	5	4		11				13	15			
	8	11		14						2		3			
		13	3			10							2	12	
						6		9	4					8	
12			8	11	15			3		5				2	
	11	15			9	12			13	6				14	
7					2				14	12				13	
10		2					4	15						7	
		7		12	13		5	8		9		14	15		
		6	15												
	14								10		3	6		4	
3		5			8	10	13		7		9	12		11	

Solución

1

- Crear un directorio y, en este, un fichero con el sudoku dado:

```
mkdir ejercicio2
```

```
bigdata@bigdata:~/hadoop$ mkdir ejercicio2
```

```
sudo nano sudo.txt
```

```
GNU nano 2.2.6                                         Archivo: sudo.txt

? 4 6 3 ? ? 8 2 5
? ? ? ? ? 2 4 ? ?
8 2 ? ? 6 ? ? 7 ?
7 ? ? 4 ? ? ? ? 2
3 8 ? ? ? ? 6 ? 9
? 6 ? 2 8 ? ? ? ?
? 7 ? ? ? 5 ? 3 ?
5 3 8 ? ? 7 ? ? ?
? 9 ? ? ? ? ? ? 6
```

```
bigdata@bigdata:~/hadoop$ cd ejercicio2
bigdata@bigdata:~/hadoop/ejercicio2$ ls
sudo.txt
```

- Añadir el fichero a HDFS y comprobar que se haya añadido correctamente:

Crear el directorio ejercicio2: hdfs dfs -mkdir /ejercicio2

Copiar el fichero anterior a un directorio en HDFS:

hdfs dfs -copyFromLocal ejercicio2 /ejercicio2

Comprobar que se ha copiado correctamente:

hdfs dfs -ls /

hdfs dfs -ls /ejercicio2/ejercicio2

```
bigdata@bigdata:~/hadoop$ hdfs dfs -mkdir /ejercicio2
16/03/06 18:39:47 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using ble
bigdata@bigdata:~/hadoop$ hdfs dfs -copyFromLocal ejercicio2 /ejercicio2
16/03/06 18:40:18 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using ble
bigdata@bigdata:~/hadoop$ hdfs dfs -ls /
16/03/06 18:40:31 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using ble
Found 7 items
drwxr-xr-x  - bigdata supergroup      0 2016-03-02 21:47 /ejercicio1
drwxr-xr-x  - bigdata supergroup      0 2016-03-06 18:40 /ejercicio2
drwxr-xr-x  - bigdata supergroup      0 2016-03-02 21:51 /out
drwxr-xr-x  - bigdata supergroup      0 2016-03-06 12:22 /outcanciones
drwxr-xr-x  - bigdata supergroup      0 2016-03-06 12:12 /practica1
drwx-----  - bigdata supergroup      0 2016-03-02 21:48 /tmp
drwxr-xr-x  - bigdata supergroup      0 2016-03-05 12:26 /user
bigdata@bigdata:~/hadoop$ hdfs dfs -ls /ejercicio2
16/03/06 18:40:42 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using ble
Found 1 items
drwxr-xr-x  - bigdata supergroup      0 2016-03-06 18:40 /ejercicio2/ejercicio2
bigdata@bigdata:~/hadoop$ hdfs dfs -ls /ejercicio2/ejercicio2
16/03/06 18:41:02 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using ble
Found 1 items
-rw-r--r--  1 bigdata supergroup     162 2016-03-06 18:40 /ejercicio2/ejercicio2/sudo.txt
```

- Comprobar que, si lo ejecuto sobre el sistema de ficheros HDFS, no me funciona correctamente y sin embargo lo ejecuto en local y sí obtengo respuesta:

```
bigdata@bigdata:~$ hadoop jar /home/bigdata/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.1.jar sudoku /home/bigdata/hadoop/ejercicio2/sudo.txt
Solving /home/bigdata/hadoop/ejercicio2/sudo.txt
1 4 6 3 7 9 8 2 5
9 5 7 8 1 2 4 6 3
8 2 3 5 4 9 7 1
7 1 5 4 9 6 3 8 2
3 8 2 7 5 1 6 4 9
4 6 9 2 8 3 5 1 7
6 7 1 9 4 5 2 3 8
5 3 8 6 2 7 1 9 4
2 9 4 1 3 8 7 5 6

Found 1 solutions
bigdata@bigdata:~$
```

2

OPCIONAL:

- Crear un fichero para el sudoku:

sudo nano sudokuopc.txt

```
bigdata@bigdata:~/hadoop/ejercicio2$ sudo nano sudokuopc.txt
bigdata@bigdata:~/hadoop/ejercicio2$ ls
sudokuopc.txt  sudo.txt
bigdata@bigdata:~/hadoop/ejercicio2$
```

GNU nano 2.2.6	Archivo: sudokuopc.txt
? 6 8 ? 2 14 15 ? ? 5 ? 16 10 ? ? ?	
? 5 ? 12 ? ? 9 ? 10 ? 3 ? 7 ? ? ?	
9 15 ? ? 8 ? ? 12 ? 2 ? ? ? 13 ? 5	
13 ? ? 4 ? 6 ? ? 7 ? ? ? ? ? ? ? ?	
2 10 ? 14 5 4 ? 11 ? ? ? 13 15 ? ? ?	
? 8 11 ? 14 ? ? ? ? ? 2 ? 3 ? ?	
? ? 13 3 ? ? 10 ? ? ? ? ? ? ? ? 2 12	
? ? ? ? ? ? 6 ? 9 4 ? ? ? ? ? 8	
12 ? ? 8 11 15 ? ? 3 ? ? 5 ? ? ? 2	
? 11 15 ? ? 9 12 ? ? ? 13 6 ? ? ? 14	
7 ? ? ? ? 2 ? ? ? ? ? 14 12 ? ? 13	
10 ? 2 ? ? ? ? ? 4 15 ? ? ? ? ? 7	
? ? 7 ? 12 13 ? 5 8 ? ? 9 ? 14 15 ?	
? ? 6 15 ? ? ? ? ? ? ? ? ? ? ? ? ?	
? 14 ? ? ? ? ? ? ? 10 ? 3 6 ? 4	
3 ? 5 ? ? ? 8 10 13 ? 7 ? 9 12 ? 11	

hadoop jar /home/bigdata/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.8.0.jar sudoku /home/bigdata/hadoop/ejercicio2/sudokuopc.txt

```
bigdata@bigdata:~/hadoop$ hadoop jar /home/bigdata/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.1.jar sudoku /home/bigdata/hadoop/ejercicio2/sudokuopc.txt
Solving /home/bigdata/hadoop/ejercicio2/sudokuopc.txt
1 6 8 7 2 14 15 13 9 5 12 16 10 11 4 3
14 5 16 12 1 11 9 4 10 13 3 8 7 2 6 15
9 15 3 11 8 10 7 12 6 2 14 4 1 13 16 5
13 2 10 4 3 6 5 16 7 1 15 11 14 8 12 9
2 10 12 14 5 4 1 11 16 3 8 13 15 9 7 6
4 8 11 9 14 12 16 7 15 6 5 2 13 3 10 1
6 16 13 3 9 8 10 15 14 11 1 7 4 5 2 12
15 7 1 5 13 3 2 6 12 9 4 10 11 16 14 8
12 13 14 8 11 15 4 1 3 7 16 5 6 10 9 2
5 11 15 1 7 9 12 8 2 10 13 6 16 4 3 14
7 9 4 16 10 2 6 3 1 8 11 14 12 15 5 13
10 3 2 6 16 5 13 14 4 15 9 12 8 1 11 7
11 1 7 10 12 13 3 5 8 4 6 9 2 14 15 16
8 12 6 15 4 1 14 9 11 16 2 3 5 7 13 10
16 14 9 13 15 7 11 2 5 12 10 1 3 6 8 4
3 4 5 2 6 16 8 10 13 14 7 15 9 12 1 11

Found 1 solutions
```

Observando el resultado, vemos:

Fila 1, columna 8 → **valor 13**

Fila 7, columna 5 → **valor 9**

Fila 9, columna 10 → **valor 7**

Fila 13, columna 6 → **valor 13**

Recursos

Enlaces de Interés

-  <http://www.project-voldemort.com/voldemort/quickstart.html>
NoSQL Databases
-  <http://docs.basho.com/riak/latest/>
NoSQL Databases
-  <http://zookeeper.apache.org/doc/r3.3.4/index.html>
NoSQL Databases
-  <http://hbase.apache.org/book/book.html>
NoSQL Databases
-  <http://hortonworks.com/get-started/>
MapReduce
-  <https://hive.apache.org/>
MapReduce
-  <http://pig.apache.org/>
MapReduce
-  <http://doc.mapr.com/display/MapR/Home>
MapReduce
-  <http://docs.aws.amazon.com/AmazonS3/latest/gsg/GetStartedWithS3.html>
Almacenamiento
-  <http://wiki.apache.org/hadoop/GettingStartedWithHadoop>
Almacenamiento
-  <http://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-2.8.0/hadoop-2.8.0.tar.gz>
<http://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-2.8.0/hadoop-2.8.0.tar.gz>
Proceso de instalación de Hadoop

Bibliografía

- ***Big Data Glossary.*** : Warden, Pete. Sebastopol, CA: O'Reilly Media; 2011.
- ***Big Data Now: Current Perspectives from O'Reilly Media.*** : O'Reilly Radar. Sebastopol, CA: O'Reilly Media; 2016.
- ***Cassandra: The Definitive Guide.*** : Hewitt, Eben. Sebastopol, CA: O'Reilly Media; 2010.
- ***Hadoop in Action.*** : Lam, Chuck. Shelter Island, NY: Manning; 2010.
- ***Hadoop: The Definitive Guide.*** : White, Tom. Sebastopol, CA: O'Reilly Media; 2012.
- ***HBase: The Definitive Guide.*** : George, Lars. Sebastopol, CA: O'Reilly Media; 2011.
- ***Pro Hadoop.*** : Venner, Jason. Berkeley, CA: Apress; 2009.

- **Professional NoSQL.** : Tiwari, Shashank. Hoboken, NJ: Wrox; 2011.
- **Redis Cookbook.** : Macedo, Tiago y Oliveira, Fred. Sebastopol, CA: O'Reilly Media; 2011.
- **Bibliografía complementaria. Amazon Web Services 100 Success Secrets.** : Robinson, Donald. Sebastopol, CA: O'Reilly Media; 2008.
- **Bibliografía complementaria. Del cloud computing al big data.** : Torres i Viñals, Jordi. Barcelona: Universitat Oberta de Catalunya; 2012.
- **Bibliografía complementaria. Programación en Java 2.** : Joyanes, Luis y Zahonero, Ignacio. Madrid: McGraw Hill; 2002.
- **Bibliografía complementaria. Programming Amazon EC2.** : Van Vliet, Jurg y Paganelli, Flavia. Sebastopol, CA: O'Reilly Media; 2011.
- **Bibliografía complementaria. Linux Pocket Guide.** : Barret, Daniel J. Sebastopol, CA: O'Reilly Media; 2012.
- **Bibliografía complementaria. Management Strategies for the Cloud Revolution.** : Babcock, Charls. Nueva York: McGraw Hill; 2010.
- **Bibliografía complementaria. A Practical Guide to Linux Commands, Editors, and Shell Programming.** : Sobell, Mark G. Nueva Jersey: Prentice Hall; 2012.
- **Bibliografía complementaria. Handbook of Cloud Computing.** : Furht, Borko y Escalante, Armando. Nueva York: Springer; 2010.

Glosario.

- **Batch:** Ejecución de un programa sin el control o supervisión directa del usuario. Este tipo de programas se caracterizan porque su ejecución no precisa ningún tipo de interacción con el usuario.
- **Clúster:** Conjunto de 1 a n racks.
- **Demonios:** En el ámbito de la informática, proceso que corre en segundo plano sobre el sistema operativo de la máquina.
- **Distribución:** En el ámbito del software, es un conjunto de software específico (o una colección de múltiples softwares, incluso un sistema operativo), ya compilado y configurado. Generalmente puede tomar forma de licencia (entre las más usadas está la licencia GPL u open source). También puede tomar la forma de una distribución binaria, un instalador (.exe) que puede descargarse de internet.
- **Framework:** Un framework, entorno de trabajo o marco de trabajo es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.
- **Hadoop:** Framework de procesamiento paralelo.
- **Java:** Lenguaje de programación.
- **Ley de Moore:** Término informático acuñado en la década de los sesenta a raíz del trabajo de Gordon Moore (cofundador de Intel), que establece que la velocidad del procesador o la capacidad de procesamiento total de las computadoras se duplica cada doce meses.
- **Logs:** Historial de log o registro a la grabación secuencial en un archivo de todos los acontecimientos (eventos o acciones) que afectan a un proceso particular (aplicación, actividad de una red informática, etc.). De esta forma constituye una evidencia del comportamiento del sistema.
- **Nodo:** Equipo físico que posee sus propios componentes hardware y software.

- **NoSQL:** A veces llamado “no solo SQL”. Amplia clase de sistemas de gestión de bases de datos que difieren del modelo clásico de SGBDR (sistema de gestión de bases de datos relacionales) en aspectos importantes, el más destacado que no usan SQL como lenguaje principal de consultas.
- **Query:** En informática, instrucción de consulta a una base de datos.
- **Rack:** Conjunto de 1 a n nodos que se suelen agrupar en estructuras metálicas con dicho nombre que presentan unas dimensiones establecidas por la industria y localizados en una misma red.
- **Sistemas distribuidos:** Conjunto de ordenadores o nodos separados físicamente y conectados entre sí por una red de comunicaciones
- **SQL:** Siglas en inglés de Structured Query Language