FORMACIÓN HADOOP

CURSO:

DESARROLLADOR PARA APACHE HADOOP

CAPÍTULO 3: HADOOP – CONCEPTOS BÁSICOS

Índice

Índice

1

Introducción a Hadoop

1.1 Proyecto Hadoop

- Hadoop es un proyecto open-source supervisado por Apache Software Fundation.

- Originalmente Hadoop está basado en los 'papers' publicados por Google en 2003 y 2004:

Paper #1: [2003] The Google File System

http://research.google.com/archive/gfs-sosp2003.pdf

Paper #2: [2004] MapReduce:

Simplified Data Processing on Large Cluster

http://research.google.com/archive/mapreduce-osdi04.pdf

- Los commiters de Hadoop trabajan en diferentes organizaciones:

  Incluyendo Cloudera, Yahoo, Facebook, LinkedIn, HortonWorks, Twitter, eBay...

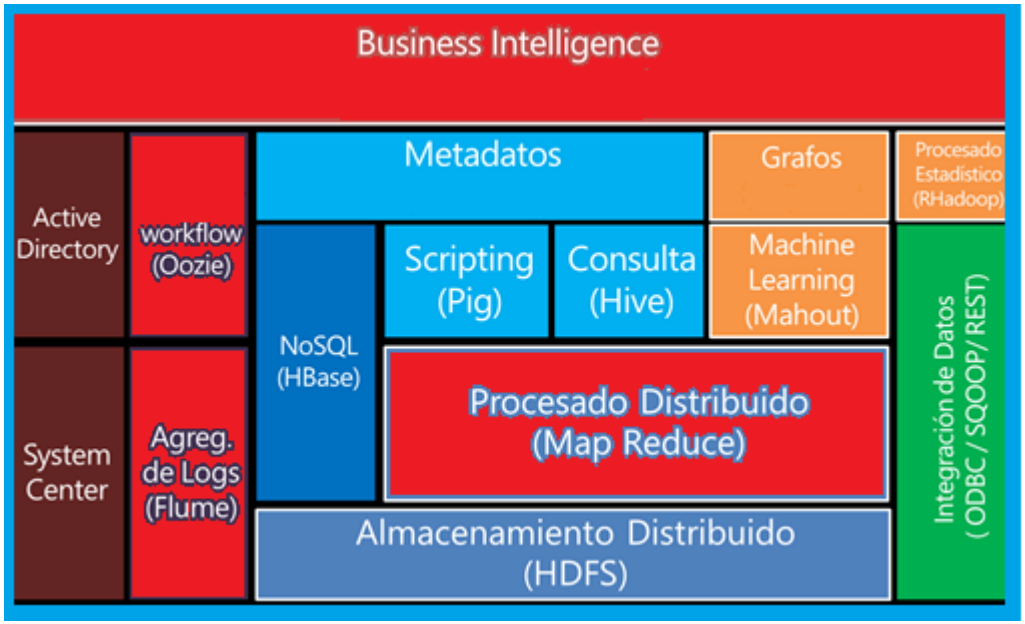
1.2 Conceptos de Hadoop

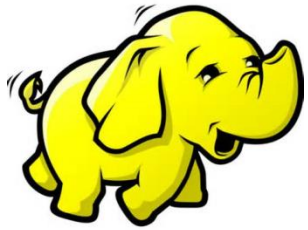
- El Core de Hadoop se divide en 2 componentes:
  1. Hadoop Distributed File System (HDFS).
  2. MapReduce.

1.2 Conceptos de Hadoop

- Hay muchos otros proyectos que se basan en el Core de Hadoop (esto es a lo que nos referimos como ecosistema de Hadoop). Algunos de estos proyectos son:

  Pig, Hive, Hbase, Flume, Oozie, Sqoop, Impala, etc (más adelante en el curso veremos varios de ellos en detalle).



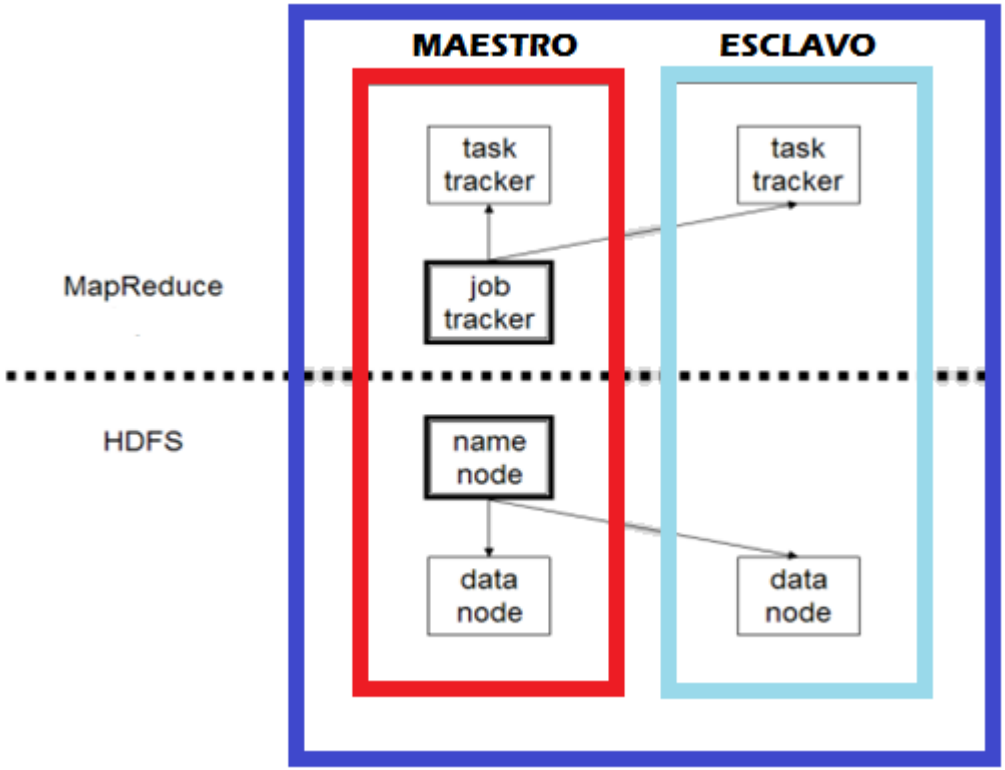


1.2 Conceptos de Hadoop

- Si lo que necesitamos es realizar procesamiento de datos a gran escala, necesitaremos dos cosas: Un lugar para almacenar grandes cantidades de datos y un sistema para el procesamiento de la misma. HDFS proporciona el almacenamiento y MapReduce ofrece una manera de procesarlo.

- Un conjunto de máquinas que ejecutan HDFS y MapReduce se conoce como cluster Hadoop.

- Un cluster puede contener desde de 1 nodo hasta una gran cantidad de nodos.



En la siguiente imagen, vemos la composición de un cluster Hadoop con los demonios de la versión 1 de MapReduce. Esta versión es la recomendada para la distribución CDH4. En la distribución CDH5 la versión recomendada de MapReduce es la versión 2 (YARN).

Como podemos observar, un cluster Hadoop se compone como mínimo de un nodo Maestro y un nodo Esclavo.

2

Cluster Hadoop

- ***Demonios perteneciente a HDFS:***
  - NameNode: Mantiene los metadatos del sistema de ficheros HDFS.
  - Secondary NameNode: Realiza funciones de mantenimiento del NameNode. No hace funciones de backup del NameNode.
  - DataNode : Almacena los bloques de datos del HDFS.

- ***Demonios pertenecientes a MapReduce V1:***
  - JobTracker: Administra los jobs MapReduce y distribuye las tareas individuales a los nodos.
  - TaskTracker : Crea una instancia y supervisa la tarea individual Map o Reduce.

- ***Demonios pertenecientes a MapReduce V2 (YARN):***
  - ResourceManager: Inicia las aplicaciones (Jobs) y gestiona los recursos. Uno por cluster.
  - ApplicationMaster: Se encarga de ejecutar las aplicaciones iniciadas por el ResourceManager. Uno por aplicación (Job).
  - NodeManager: Ejecuta las tareas Map o Reduce de cada una de las aplicaciones. Uno por nodo esclavo.
  - JobHistory: Se encarga de almacenar las estadísticas y los metadatos. Uno por cluster.

Un cluster únicamente puede tener en ejecución una de las dos versiones de MapReduce. En ningún caso es posible tener las dos versiones de MapReduce en ejecución al mismo tiempo.

2.1 Demonios Hdfs

- **Demonios HDFS:**

Existen dos formas de tener configurado nuestro cluster Hadoop:

Sin alta disponiblidad (3 demonios):

- NameNode (maestro)
- Secondary NameNode (maestro)
- DataNode (en cada uno de los esclavos)

Con alta disponiblidad (2 demonios):

- NameNode (maestro y otro NameNode en standby en otro maestro)
- DataNode(en cada uno de los esclavos)

2.1 Demonios Hdfs

Demonio NameNode (maestro):

El NameNode es el encargado de almacenar los metadatos del cluster Hadoop:

- Información de la localización de los ficheros del HDFS
- Información de permisos y usuarios
- Nombre de los bloques
- Localización de los bloques

Los metadatos son almacenados en el disco y son cargados en memoria cuando el NameNode arranca por primera vez:

- Se almacenan en el fichero fsimage (los nombres del bloque no son almacenados en el fichero)
- Los cambios realizados en los metadatos son almacenados en memoria y en el fichero edits.

2.1 Demonios Hdfs

Demonio DataNode (esclavo):

El DataNode es el encargado de acceder a los bloques almacenados en cada uno de los nodos esclavos:

- El nombre de los bloques es: blk_xxxxxxxxx
- En los nodos esclavo únicamente se almacenan los datos, no se almacena ningún tipo de información referente al bloque (nombre, parte de.., etc).

Cada uno de los bloques es almacenado en varios nodos dependiendo de la configuración de la replicación

Cada uno de los esclavos debe tener el demonio DataNode para poder comunicarse con el NameNode y poder acceder a los datos

2.1 Demonios Hdfs

Demonio Secondary NameNode (master):

El Secondary NameNode no es un NameNode en standby.

El Secondary NameNode se encarga de operaciones de mantenimiento:

- El NameNode escribe las modificaciones de los metadatos en el fichero edits.
- El Secondary NameNode periódicamente obtiene el fichero edits del NameNode y lo une con el fichero fsimage y envía el nuevo snapshot al NameNode.

El Secondary NameNode se debe poner en otro nodo maestro en un cluster de un tamaño considerable.

Checkpointing Secondary NameNode:

Cada hora o cada 1.000.000 de transacciones (por defecto) el Secondary NameNode realiza el checkpoint de los metadatos:

1. Llama al NameNode para obtener el fichero edits
2. Obtiene los ficheros fsimage y edits del NameNode
3. Carga el fichero fsimage en memory y agrega los cambios del fichero edits
4. Crea el nuevo fichero consolidado de fsimage
5. Envía el nuevo fichero consolidado al NameNode
6. El NameNode reemplaza el antiguo fsimage por el nuevo

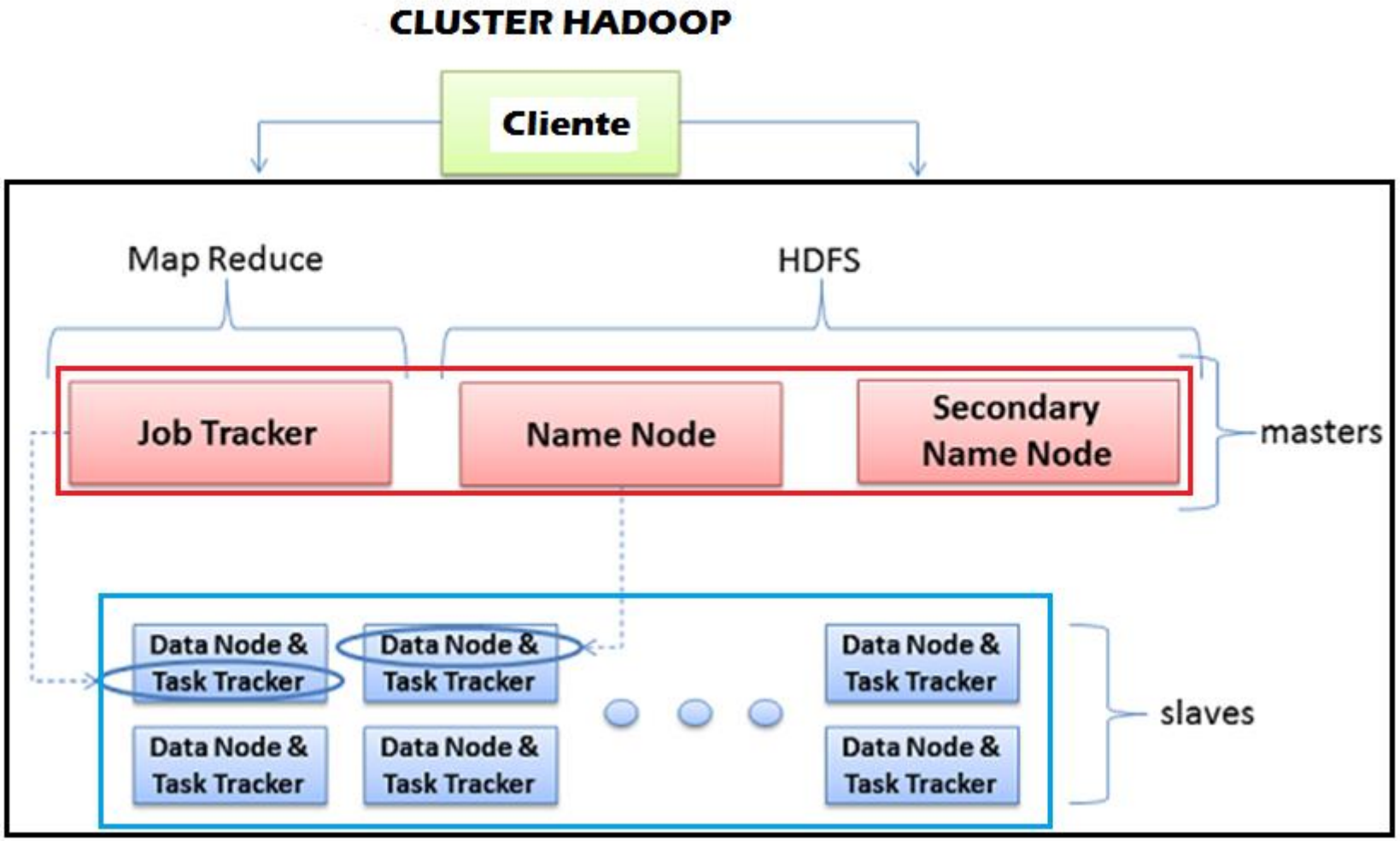
2.2 Demonios Hdfs y MapReduce V1

- Cada demonio se ejecuta en su propia máquina virtual Java (JVM).
- En un cluster real no deben ejecutarse los 5 demonios en la misma máquina.
- En un entorno *pseudoDistribuido* tendremos los 5 demonios en la misma máquina (este entorno es el que se suele utilizar para la programación y testeo de aplicaciones MapReduce).
- Los demonios deben ir situados de la siguiente manera:

  - ***Nodo Maestro (Master):*** Ejecuta el NameNode, Secondary NameNode y JobTracker. Sólo puede ejecutarse una instancia de cada demonio en el cluster.
  - ***Nodo Esclavo (Slave):*** Ejecuta el DataNode y TaskTracker. En cada slave del cluster se ejecuta una instancia de los demonios.

En casi todos los cluster pequeños, se suelen ejecutar en la misma máquina los 3 demonios (NameNode, Secondary NameNode y JobTracker). A partir de de un cluster con 20-30 Nodos esclavos, se ejecutan en máquinas diferentes.

2.2 Demonios Hdfs y MapReduce V1



2.3 Demonios Hdfs y MapReduce V2

- En un cluster real no deben ejecutarse los 7 demonios en la misma máquina.

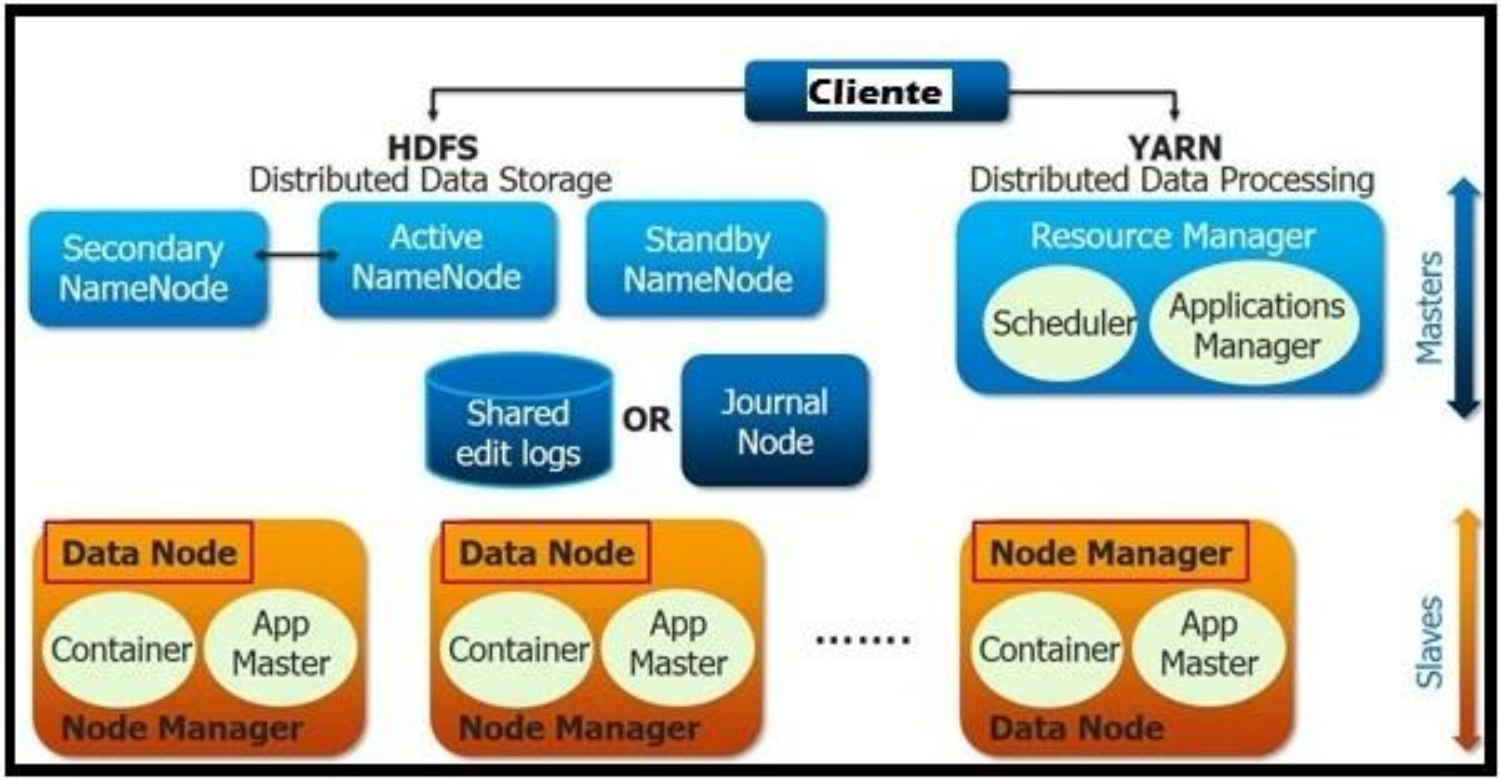
- En un entorno *pseudoDistribuido* tendremos los 7 demonios en la misma máquina (este entorno es el que se suele utilizar para la programación y testeo de aplicaciones MapReduce).

- Los demonios deben ir situados de la siguiente manera:

  - ***Nodo Maestro (Master):*** Ejecuta el NameNode, Secondary NameNode y ResourceManager.
  - ***Nodo Esclavo (Slave):*** Ejecuta el DataNode , NodeManager y ApplicationMaster.
  - **Nodo "History"** : Ejecuta el JobHistory, encargado de almacenar las métricas.

En casi todos los cluster pequeños, se suelen ejecutar en la misma máquina los 3 demonios (NameNode, Secondary NameNode y ResourceManager). A partir de de un cluster con 20-30 Nodos esclavos, se ejecutan en máquinas diferentes.

2.3 Demonios Hdfs y MapReduce V2



3

Características de Hadoop Distributed File System (HDFS)

El HDFS es el responsable del almacenamiento de los datos en el cluster.

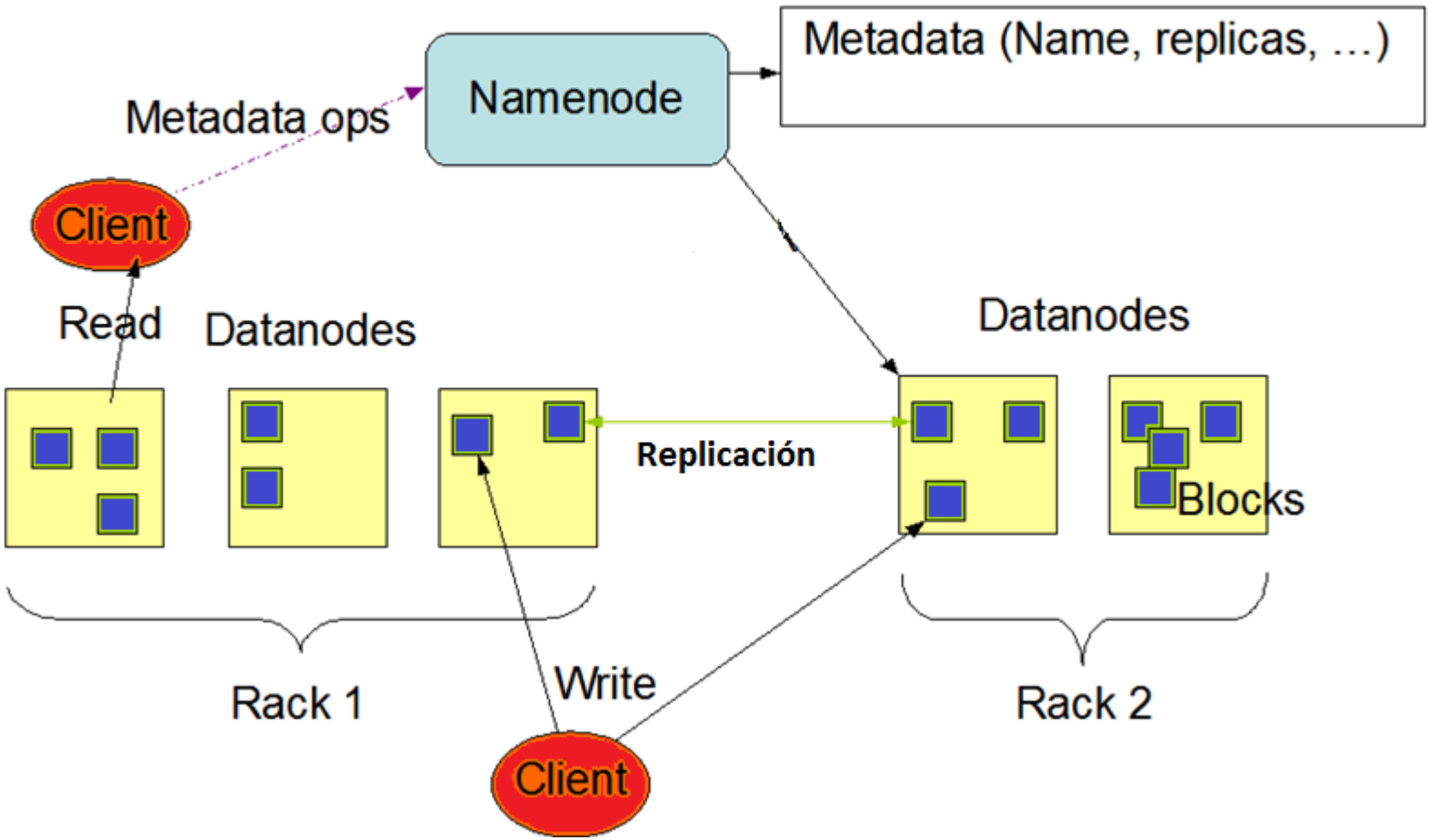
Los datos se dividen en bloques y son almacenados en varios nodos del cluster. El tamaño por defecto del bloque es de 64Mb, pero Cloudera recomienda 128Mb. Siempre que sea posible los bloques son replicados varias veces en diferentes nodos. La replicación por defecto de la distribución de Cloudera es de 3.

Dicha replicación aumenta la *fiabilidad* (debido a que hay varias copias), la *disponibilidad* (debido a que estas copias se distribuyen a diferentes máquinas) y el *rendimiento* (debido a que más copias significan más oportunidades para ***"llevar la computación a los datos"***, principal objetivo de un sistema Hadoop).

El sistema de ficheros de Hadoop (HDFS) está optimizado para la lectura de grandes cantidades de datos con el fin de reducir al mínimo el impacto global del rendimiento (latencia).

En la imagen podemos observar el almacenamiento de 1 ficheros en HDFS. En este caso la replicación es de 3, con lo que cada uno de los bloques del fichero es almacenado en los diferentes nodos del cluster.

ARQUITECTURA HDFS



Procedimiento de escritura de un fichero en HDFS:

1. El cliente conecta con el NameNode
2. El NameNode busca en sus metadatos y devuelve al cliente el nombre del bloque y la lista de los DataNode
3. El cliente conecta con el primer DataNode de la lista y empieza el envío de los datos
4. Se conecta con el segundo DataNode para realizar el envío
5. Idem con el tercer DataNode
6. Finaliza el envío
7. El cliente indica al NameNode donde se ha realizado la escritura del bloque

Procedimiento de lectura de un fichero en HDFS:

1. El cliente conecta con el NameNode
2. El NameNode devuelve una lista con los DataNode que contienen ese bloque
3. El cliente conecta con el primero de los DataNode y comienza la lectura del bloque

Fiabilidad y recuperación del HDFS:
Los DataNode envían heartbeats al NameNode cada 3 segundos para indicar su estado

Si pasado un tiempo (por defecto 5 minutos) el NameNode no recibe el heartbeat de alguno de los nodos, da por perdido ese nodo:

- El NameNode averigua que bloques había en el nodo perdido
- El NameNode busca otros DataNode donde realizar la copia de los bloques perdidos y así mantener el número de replicación. Los DataNode serán los encargados de realizarse la copia de los bloques "perdidos"

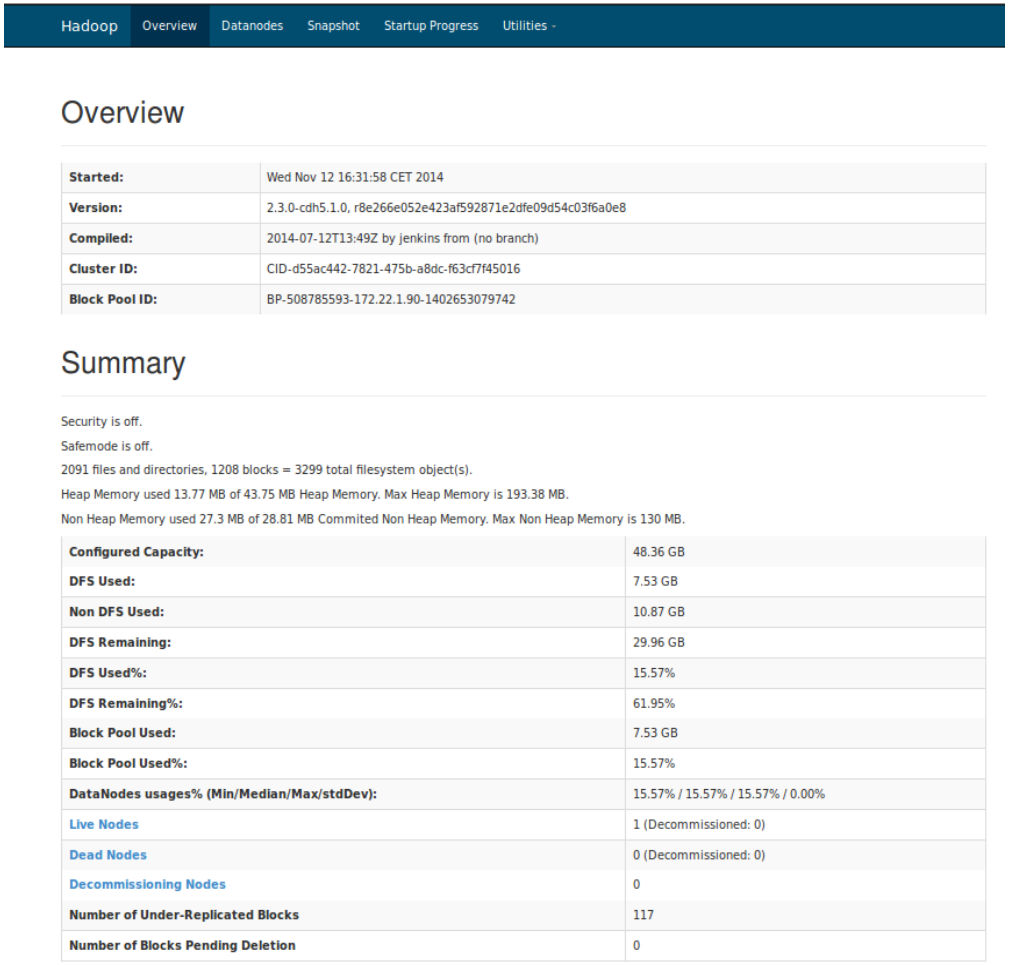
En caso de recuperar el nodo perdido, el sistema automáticamente decidirá que bloques eliminar para mantener el número de replicación de bloques.

Web UI del NameNode:
A través de la web UI del NameNode seremos capaces de ver el estado de nuestro cluster:

- Espacio total del HDFS
- Espacio ocupado del HDFS
- Espacio disponible del HDFS
- Estado de los Nodos
- Navegación por el sistema de ficheros
- ……

- Por defecto se encuentra en el puerto 50070

Web UI del NameNode:



Acceso al HDFS a través de la línea de comandos:

Nuestro sistema de fichero es muy similar a un sistema de ficheros tradicional, por lo que podemos acceder a el a través de la línea de comandos con el cliente **hadoop fs**

hadoop fs permite casi todos los comandos típicos de nuestro sistema de ficheros tradicional:

- hadoop fs -cat /file.txt
- hadoop fs -mkdir /repos
- hadoop fs -put file.txt /repos
- hadoop fs -get /repos/file.txt /tmp
- hadoop fs -ls /repos
- hadoop fs -ls -R /repos
- hadoop fs -rm /repos/file.txt

4 Conceptos de MapReduce

MapReduce no es un lenguaje de programación o incluso un marco específico, sino que con mayor precisión se podría describir como un "**modelo de programación**". Originalmente fue implementado por Google internamente, aunque posteriormente fue convertido a código abierto por Apache Hadoop.

Hay otras implementaciones MapReduce (tales como Disco y Phoenix), pero la mayoría son de propiedad o proyectos de "juguete".

Hadoop es de lejos la aplicación más utilizada de MapReduce.

4.1 MapReduce

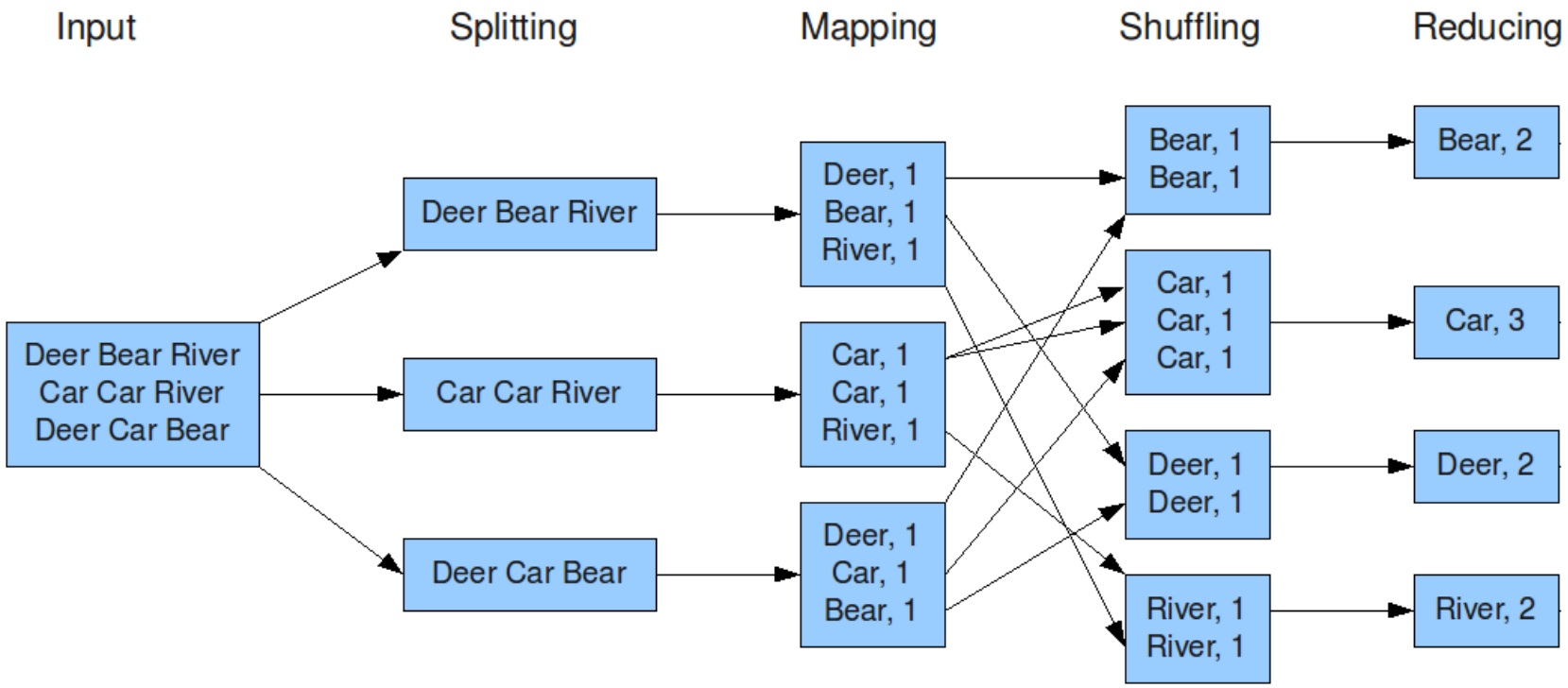
• MapReduce es el sistema utilizado para procesar los datos en el cluster Hadoop. Se compone de 2 fases: Map y Reduce. Entre las dos fases se produce la fase de shuffle and sort.

• Cada tarea Map opera con una parte de los datos, normalmente, un bloque HDFS de los datos. Una vez finalizadas todas las tareas Map, el sistema MapReduce distribuye todos los datos intermedios de los nodos para realizar la fase de Reduce.

• Como podemos observar en la imagen siguiente, dado un fichero es dividido en varios input split de entrada, estos son enviados a diferentes tareas Map. Una vez finalizadas todas las tareas Map se realiza la fase de shuffle and short, donde los datos son ordenados para posteriormente ser enviado a los nodos para realizar la fase Reduce. La fase Reduce no empieza hasta que han finalizado TODAS las tareas Map del Job correspondiente.

• En el ejemplo nos encontramos con 4 tareas Reduce, una por cada tipo de dato. El resultado final son varios ficheros, uno por cada tarea reduce ejecutada.

4.1 MapReduce



5 Funcionamiento de un cluster Hadoop

5.1 Ejecución de un Job

En este apartado vamos a ver una primera introducción del funcionamiento de un cluster Hadoop al ejecutar un Job y el funcionamiento del procesamiento de los datos por dicho Job.

Ya que el curso está enfocado para realizar la certificación de Cloudera (CCDH) y está certificación se basa en la distribución CDH4, los ejemplos se van a realizar con la versión 1 de MapReduce. Esto debe ser independiente para un desarrollador, ya que es el cluster el que se encarga de ejecutar de una manera u otra dependiendo de la versión de MapReduce que tenga configurada.

Cuando un cliente envía un trabajo, la información de configuración es empaquetada en un fichero xml. El fichero xml junto con el .jar que contiene el código del programa es enviado al JobTracker.

El jobTracker será el encargado de enviar la solicitud al TaskTracker que crea oportuno para ejecutar la tarea. Una vez el TaskTraker recibe la solicitud de ejecutar una tarea, instancia una JVM para esa tarea. El TaskTracker puede configurarse para ejecutar diferentes tareas al mismo tiempo.

Para realizar esto, habrá que tener en cuenta que disponemos de suficiente procesador y memoria.

5.2 Procesamiento de los datos

• Los ***datos intermedios*** producidos por una tarea son almacenados en el disco local de la máquina donde se ha ejecutado. Puede resultar un poco extraño, pero es debido a que al ser datos temporales, si fueran almacenados en HDFS con su correspondiente replicación, aumentaríamos el problema de nuestro cuello de botella (la red), ya que el sistema se dispondría a replicar los datos y con ello generar tráfico en la red. Los datos intermedios serán borrados una vez el Job se ha ejecutado completamente. Nunca serán borrados antes, aunque una de las tareas del job finalice, no serán borrados hasta que finalicen el resto de tareas y con ello el job.

• En la fase de ***start up*** del Reduce los datos intermedios son distribuidos a través de la red a los Reducers. Los Reducers almacenan su salida en el sistema de ficheros HDFS.

5.2 Procesamiento de los datos

Fases de una aplicación MapReduce:

① En la primera fase los datos son divididos en tantos splits como fuera necesario. Cada split genera un Map para ser procesado.

② En la segunda fase, los datos (splits) son procesados por su Map correspondiente.

③ Una vez procesados los datos por el Mapper, los datos intermedios son almacenados en el sistema de ficheros local a la espera de ser enviados a su correspondiente Reduce. Cuando se inicia la transferencia de los datos intermedios al Reduce, se realiza la tarea de shuffle and short, donde los datos son ordenados por sus correspondientes llaves para que todas las llaves sean procesadas por el mismo Reduce.

④ Se realiza el procesamiento de los datos con su Reduce.

⑤ La salida de cada uno de los Reducer es almacenada en el sistema de ficheros HDFS realizando la replicación correspondiente.

6 Otros proyectos del ecosistema Hadoop

Existen muchos otros proyectos que utilizan Hadoop core o ayudan a utilizarlo, como puede ser el caso de Sqoop, Flume o Oozie. La mayoría de los proyectos son ahora proyectos de Apache o proyectos de incubadora de Apache.

Estar en la incubadora de Apache no significa que el proyecto es inestable o inseguro para su uso en producción, sólo significa que es un proyecto relativamente nuevo en Apache y todavía se está creando la comunidad que lo rodea.

A continuación veremos una breve explicación de cada uno de los proyectos. Durante el curso veremos con mucho más detenimiento muchos de estos proyectos del ecosistema Hadoop.

6.1 Hive

Hive es una abstracción en la parte superior de MapReduce. Permite a los usuarios poder consultar los datos del cluster Hadoop sin necesidad de saber java o MapReduce utilizando el lenguaje HiveQL.

Dicho lenguaje es muy similar al SQL. Hive se ejecuta en la máquina cliente convirtiendo la consulta HiveQL en programas MapReduce los cuales son enviados al cluster Hadoop para ser ejecutados.

Ejemplo de query HiveQL:

```
SELECT stock.producto FROM stock JOIN ventas ON (stock.id = ventas.id);
```

Más información en: https://hive.apache.org/

6.2 Pig

Pig es una alternativa de abstracción para realizar programas MapReduce. Usa un lenguaje de scripting llamado PigLatin. El interprete de Pig es ejecutado en la máquina cliente transformando el código en programas MapReduce para posteriormente enviarlos al cluster Hadoop para su ejecución.

Ejemplo de un programa Pig:

```
A = LOAD 'data' AS (f1,f2,f3);
B = FOREACH A GENERATE f1 + 5;
C = FOREACH A generate f1 + f2;
```

Más información en: http://pig.apache.org/

6.3 Impala

Impala es un proyecto open-source creado por Cloudera. Su finalidad es facilitar la consulta en tiempo real de los datos almacenados en HDFS. Impala no usa MapReduce, es un demonio que es ejecutado en los nodos esclavos utilizando un lenguaje muy parecido al HiveQL.

Impala es mucho más rápido que Hive. Puede llegar a realizar consultas 40 veces más rápido que Hive aprovechando sus mismos metadatos.

http://blog.cloudera.com/blog/2012/10/cloudera-impala-real-time-queries-in-apache-hadoop-for-real/

Más información en: http://www.cloudera.com/content/cloudera/en/products-and-services/cdh/impala.html

6.4 Flume y Sqoop

Flume y Sqoop son dos productos totalmente diferentes con una cosa en común: Ambos ayudan a obtener datos en el clúster Hadoop.

Flume permite integrar los datos directamente en el clúster en tiempo real. Como su nombre lo indica, se utiliza con mayor frecuencia para los registros del servidor (servidor web, servidor de correo electrónico, Syslog , UNIX, etc), pero se pueden adaptar para leer datos de otras fuentes, como veremos más adelante en el curso.

Más información en: http://flume.apache.org/

Sqoop puede ser recordado como una contracción de "SQL a Hadoop" ya que es una herramienta para traer datos de una base de datos relacional al cluster Hadoop para su procesamiento o análisis, o para la exportación de datos que ya tienes en el cluster Hadoop a una base de datos externa para su posterior procesamiento o análisis.

Más información en: http://sqoop.apache.org/

6.5 Oozie

Como se verá más adelante en este curso, en realidad es una práctica bastante común tener como entrada de un trabajo MapReduce una salida de otro trabajo MapReduce.

Oozie es una herramienta que le ayudará a definir este tipo de flujo de trabajo para un cluster Hadoop, encargándose de la ejecución de los trabajos en el orden correcto, especificando la ubicación de la entrada y la salida, permitiendo definir lo que se desea realizar cuando el trabajo se completa correctamente o cuando se encuentra un error.

Más información en: http://oozie.apache.org/

6.6 Hbase

Hbase es la base de datos de Hadoop. Es una base de datos NoSql que realiza almacenamiento masivo de datos (Gigabytes, terabytes, petabytes de datos en una misma tabla). Hbase escala a una velocidad muy alta, ciento de miles de inserciones por segundo. Las tablas pueden tener miles de columnas aunque la mayoría de ellas estén vacías para una fila dada.

Uno de los inconvenientes de Hbase es que tiene un modelo de acceso muy limitado. Sus acciones mas comunes son de inserción de una fila, recuperación de una fila, escaneo completo o parcial de la tabla. En Hbase sólo una columna está indexada (row key).

Más información en: http://hbase.apache.org/

6.7 Spark

Spark es un programa de computación en paralelo que puede operar a través de cualquier fuente de entrada Hadoop: HDFS, HBase, Amazon S3, Avro, etc. Spark es un proyecto de código abierto creado por UC Berkeley AMPLab y en sus propias palabras, Spark inicialmente fue desarrollado para dos aplicaciones en las que es de gran ayuda almacenar los datos en memoria: algoritmos iterativos, que son comunes en el aprendizaje automático y minería de datos interactiva.

En este curso únicamente veremos las posibilidades que nos ofrece Spark una vez integrado en nuestro cluster Hadoop. En Formación Hadoop ofrecemos un curso avanzado para aprender a realizar nuestras propias aplicaciones de procesamientos de datos con Spark.

Más información en: https://spark.apache.org/

6.8 Hue

Hue es un proyecto open source que a través de una interfaz Web nos permitirá acceder a los componentes más comunes de nuestro cluster Hadoop. Será posible otorgar diferentes privilegios a cada uno de los usuarios dependiendo del rol que ocupen en el sistema.

Más información en: http://gethue.com/

Contacto

FORMACIÓN HADOOP

administracion@formacionhadoop.com

www.formacionhadoop.com

TWITTER
Twitter.com/formacionhadoop

FACEBOOK
Facebook.com/formacionhadoop

LINKEDIN
linkedin.com/company/formación-hadoop