

# ADMINISTRACIÓN DE INFRAESTRUCTURAS Y SISTEMAS INFORMÁTICOS (AISI)

## Grado en Ingeniería Informática

Roberto R. Expósito ([roberto.rey.exposito@udc.es](mailto:roberto.rey.exposito@udc.es))  
Jorge Veiga ([jorge.veiga@udc.es](mailto:jorge.veiga@udc.es))

# PRÁCTICA 6

**Apache Hadoop / Apache Spark**



# Contenidos

- **Objetivo**
- **Apache Hadoop**
- **Apache Spark**
- **Ejercicios propuestos**



# Objetivo

4

- El objetivo principal de esta práctica consiste en desplegar el **framework Apache Hadoop** junto con su **gestor de recursos/planificador (YARN)** y su **sistema de ficheros distribuido (HDFS)** en un clúster virtual basado en Debian GNU/Linux usando **Vagrant, Ansible y NFS**
  - Hadoop es un *framework* Big Data de código abierto implementado en **Java** orientado al almacenamiento y procesamiento por lotes (*batch processing*) de grandes volúmenes de datos
  - El funcionamiento de Hadoop se basa en distribuir los datos a procesar entre los nodos de un clúster de cómputo siguiendo una arquitectura *master/worker*
  - Ejecutaremos una sencilla aplicación Hadoop MapReduce (*WordCount*) para comprobar su correcto funcionamiento



<https://hadoop.apache.org>



# Objetivo

5

- El segundo objetivo será desplegar **Apache Spark** para ejecutar la misma aplicación *WordCount* con dicho *framework* sobre YARN y HDFS
  - Spark es un *framework* Big Data de código abierto implementado en **Scala** orientado al análisis de datos a gran escala, que ofrece soporte para procesamiento por lotes (*batch processing*) y procesamiento de flujos de datos (*stream processing*), entre otras cosas
  - Su despliegue en un clúster de cómputo está basado en una arquitectura *master/worker* similar a Hadoop, pero requiriendo en este caso de un sistema de ficheros distribuido externo (p.e. HDFS, Amazon S3, Apache HBase)
    - Spark no proporciona una implementación propia de sistema de ficheros
  - Spark también soporta un modo de despliegue en local que permite su ejecución en una sola máquina usando un sistema de ficheros local existente



<https://spark.apache.org>

# Justificación de la práctica

- La realización de esta práctica se justificará de la siguiente forma:
  - Documento en formato PDF que incluya las capturas de pantalla indicadas para demostrar la realización de la **parte principal de cada ejercicio**
    - **Debes incluir capturas similares a las mostradas en las transparencias:**
      - 28 (EJ1); 34 (EJ2); 42 (EJ3); 46 (EJ4); 49, 59, 61 (EJ5);
      - 63 (EJ6); 68, 69 (EJ7); 70 (EJ8)



Busca este icono en la parte superior derecha



- **ENTREGA** a través de Moodle: **08/05 (15:30)**
- **ES OBLIGATORIO** usar la nomenclatura que se propone para nombrar los recursos y debe apreciarse sin confusión en las capturas aportadas
  - **NO RECORTES** las capturas de pantalla, **debe verse toda la información** que sea relevante para comprobar el trabajo realizado
- **NO** seguir estas normas **IMPLICA UNA CALIFICACIÓN “C”** en esta práctica



# Contenidos

- Objetivo
- Apache Hadoop
- Apache Spark
- Ejercicios propuestos



# Apache Hadoop

8

- Hadoop se compone de tres componentes principales:



**MapReduce**: motor de procesamiento de datos distribuido inspirado en el paradigma propuesto por **Google**



**Data Processing**



**YARN (Yet Another Resource Negotiator)**: gestor de los recursos (p.e. CPU, memoria) de los nodos del clúster Hadoop y planificador de los trabajos (~aplicaciones) que se ejecutan en el mismo



**Resource Management**



**HDFS (Hadoop Distributed File System)**: sistema de ficheros distribuido para el almacenamiento de los datos



**Data Storage**

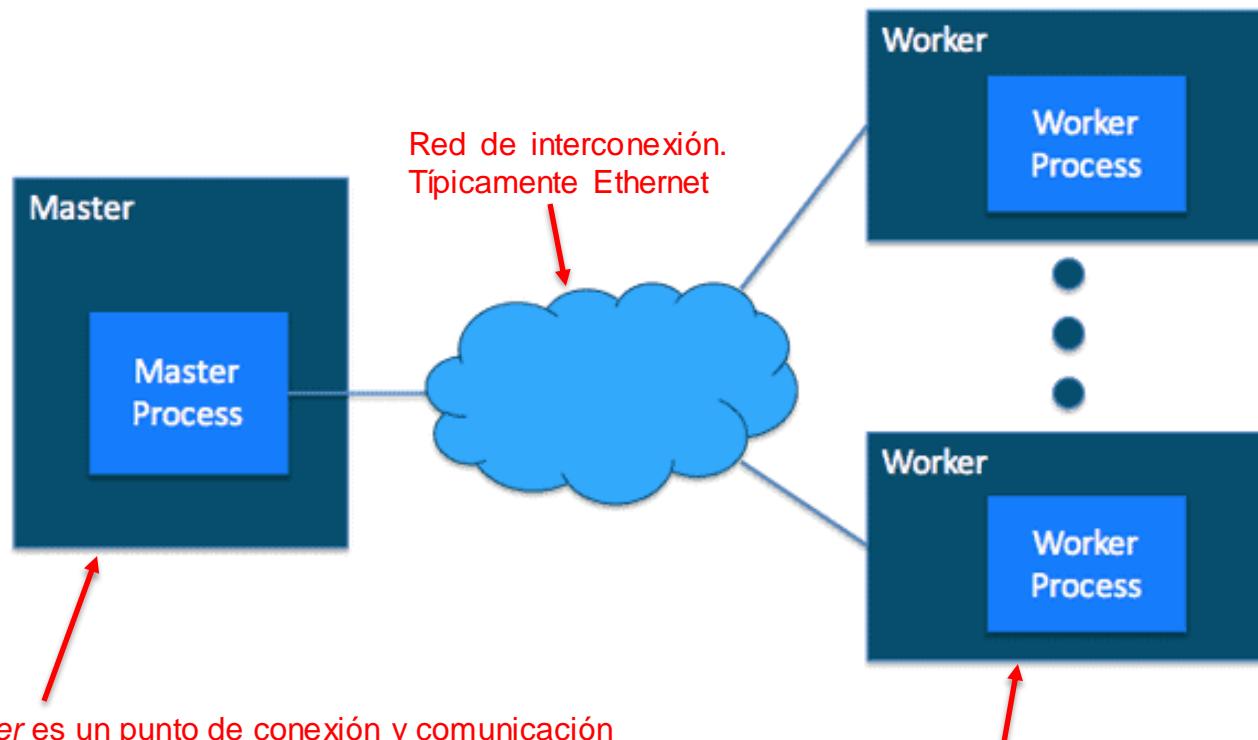




# Apache Hadoop

9

- Cada componente de Hadoop se configura en base a una **arquitectura master/worker** ejecutando los procesos/servicios correspondientes según el tipo de nodo



El nodo *master* es un punto de conexión y comunicación para los clientes que usen el clúster. Recibe sus peticiones y coordina el trabajo a realizar en el resto de nodos (en los *workers*). Suele actuar también como gestor/planificador de los recursos

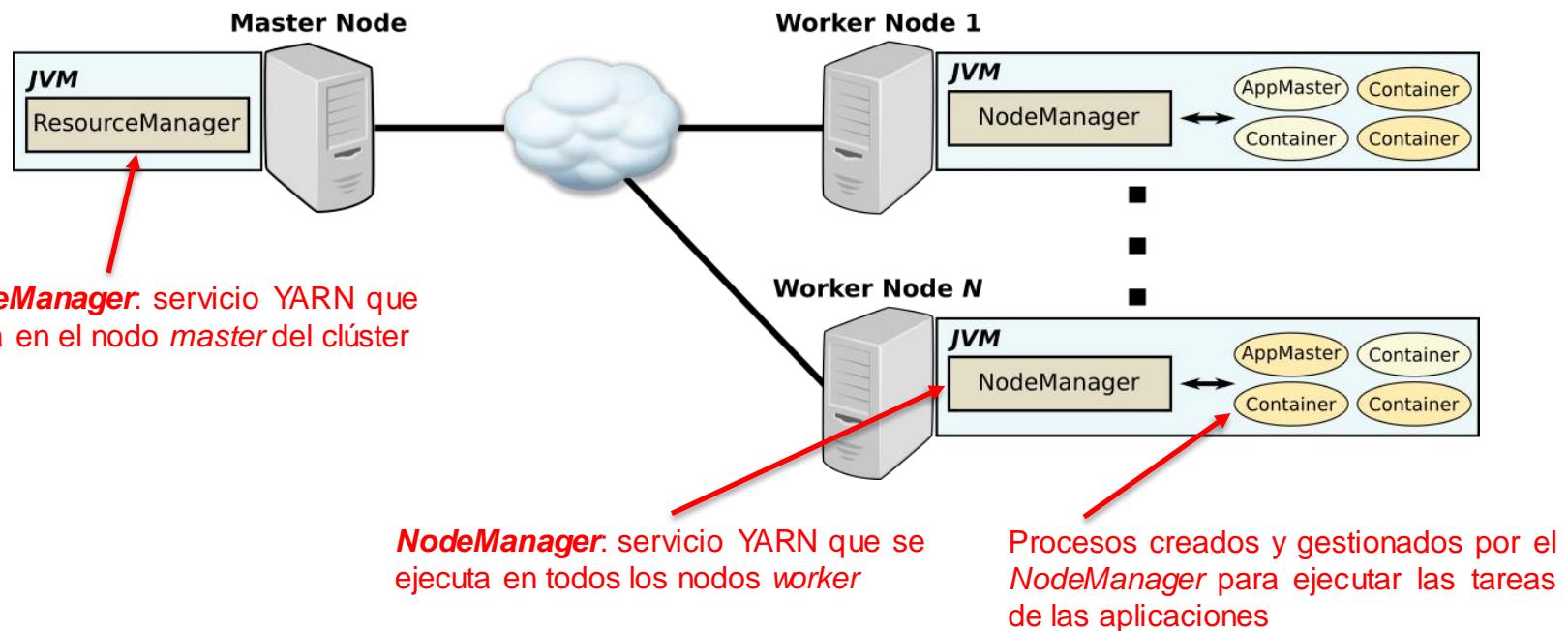
Los nodos *worker* proporcionan los recursos computacionales (CPU, memoria, disco) para la ejecución de las tareas de procesamiento que reciben desde el nodo *master*



# YARN

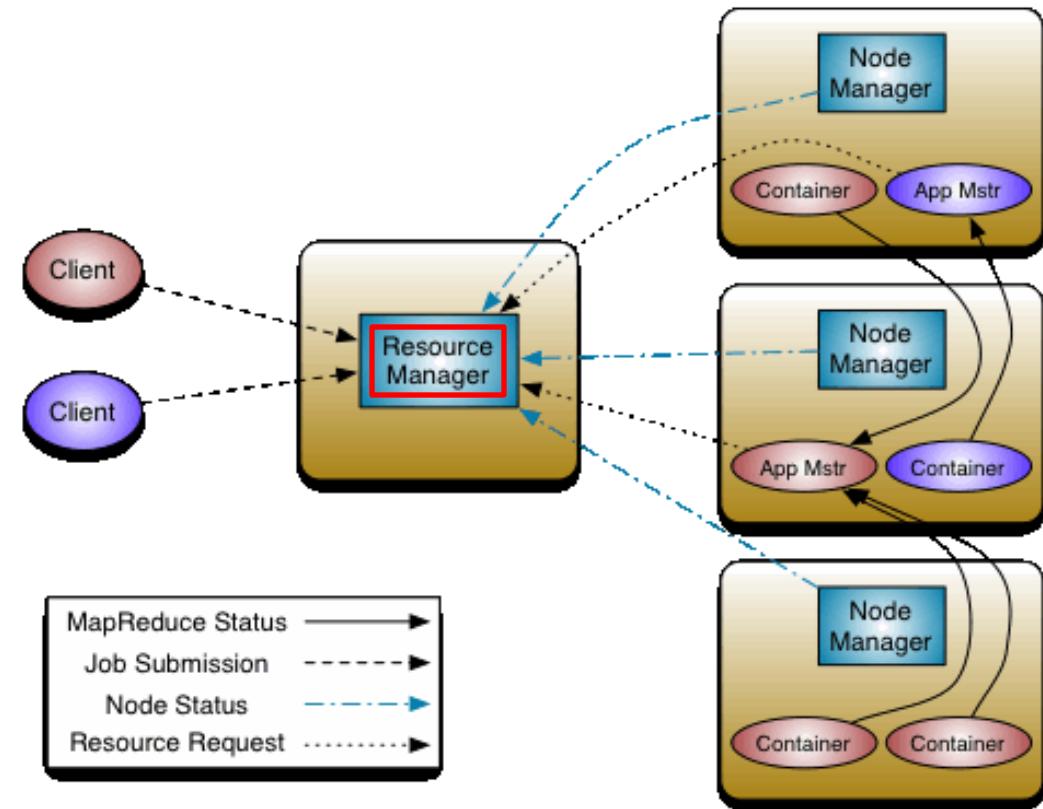
10

- Servicios principales de gestor/planificador de Hadoop
  - Se ejecutan como **procesos Java**



**ResourceManager:**

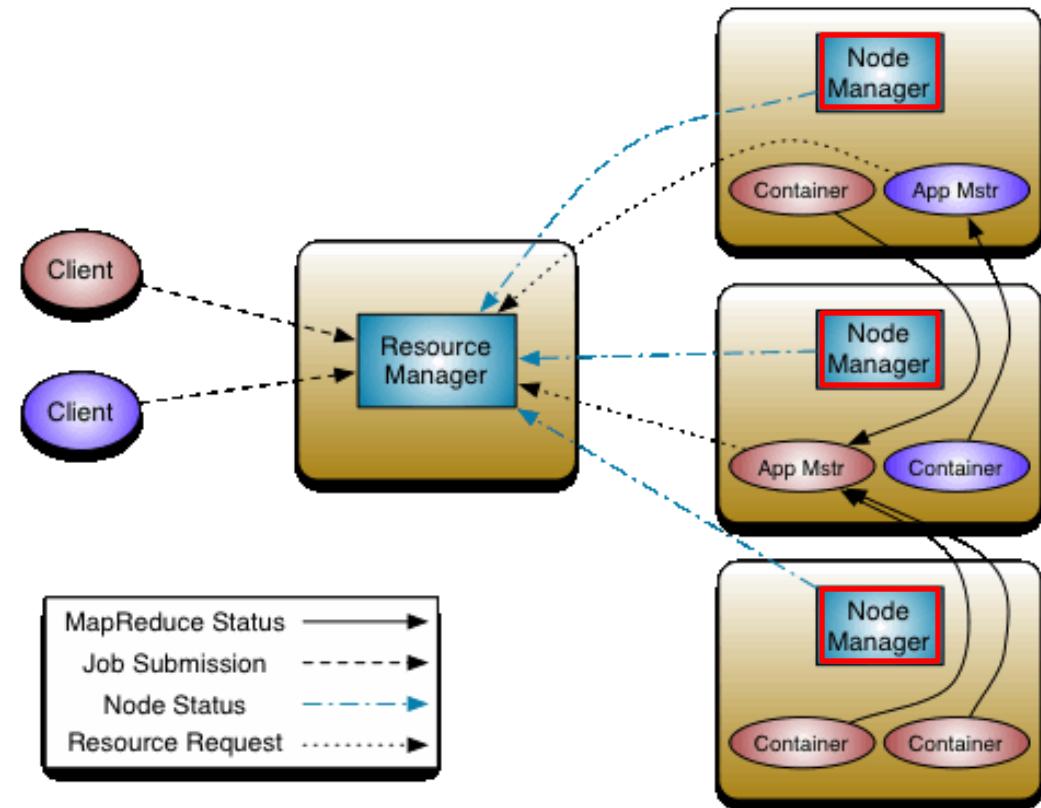
- Por un lado, integra un **planificador** responsable de dividir y asignar los recursos del clúster entre las aplicaciones ejecutadas por los clientes. La planificación se realiza en función de los requisitos de recursos especificados por las aplicaciones y basándose en la noción abstracta de un **contenedor de recursos** que proporciona una cantidad determinada de CPU, memoria, disco, etc.
- Por otro lado, también es responsable de aceptar envíos de trabajos (aplicaciones) por parte de los clientes y de negociar el primer contenedor que ejecuta el proceso **ApplicationMaster** específico de cada aplicación.





## NodeManagers:

- Se registra con el *ResourceManager* y le envía de forma periódica mensajes *heartbeat* reportando el estado de salud del nodo.
- Es responsable de crear los contenedores de recursos e iniciarlos bajo petición por parte del *ApplicationMaster* específico de cada aplicación que se ejecuta en el clúster, monitorizando además el uso de recursos (CPU, memoria, disco, red) de los contenedores e informando periódicamente al *ResourceManager* al respecto de los mismos.
- También realiza la gestión de los *logs* y se encarga de eliminar los contenedores de recursos de acuerdo a las instrucciones que recibe desde el planificador.



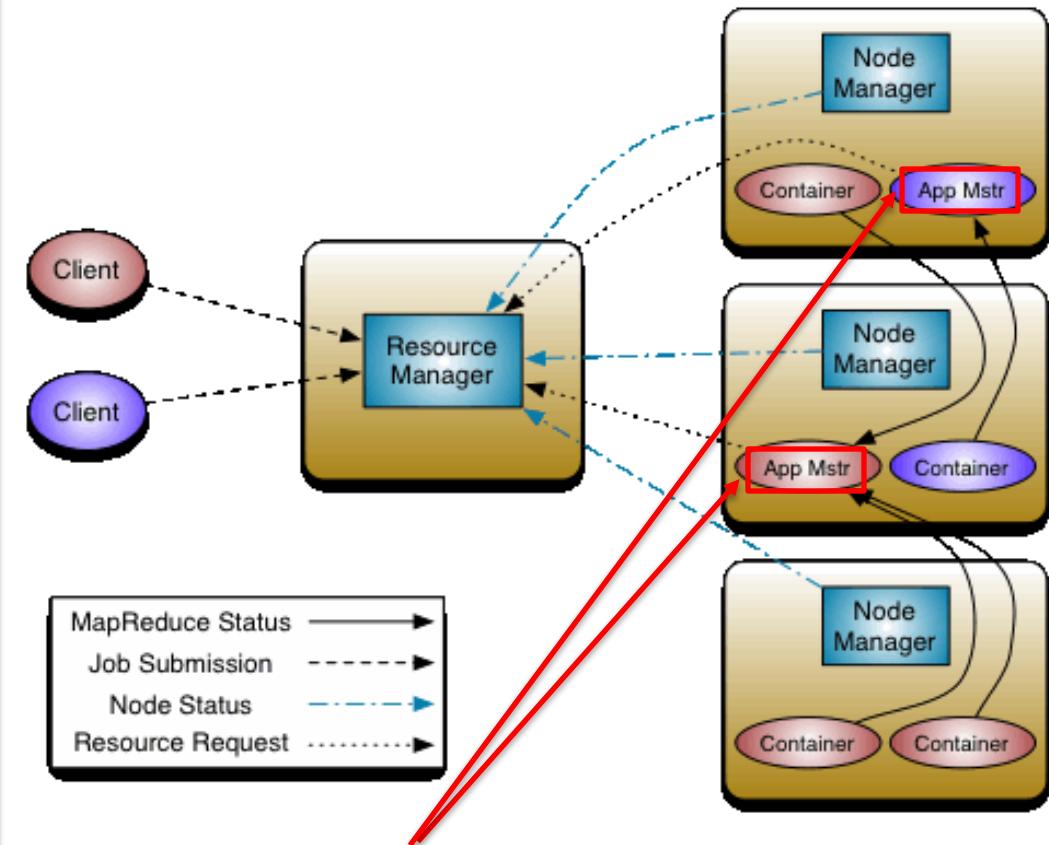


### ***ApplicationMaster:***

- Es el primer contenedor que se crea para cada aplicación que se ejecuta en el clúster, siendo responsable de negociar con el planificador los contenedores de recursos apropiados para la aplicación que gestiona, así como de monitorizar su estado y progreso durante la ejecución.

### ***Container:***

- Colección de recursos físicos como núcleos de CPU, memoria RAM y disco en un nodo *worker* del clúster.
- Los contenedores ejecutan tareas específicas de una aplicación de acuerdo a lo especificado por su correspondiente *ApplicationMaster*. Por ejemplo, los contenedores de una aplicación Hadoop MapReduce ejecutan las tareas Map y Reduce (procesos Java) encargadas del procesamiento de los datos.



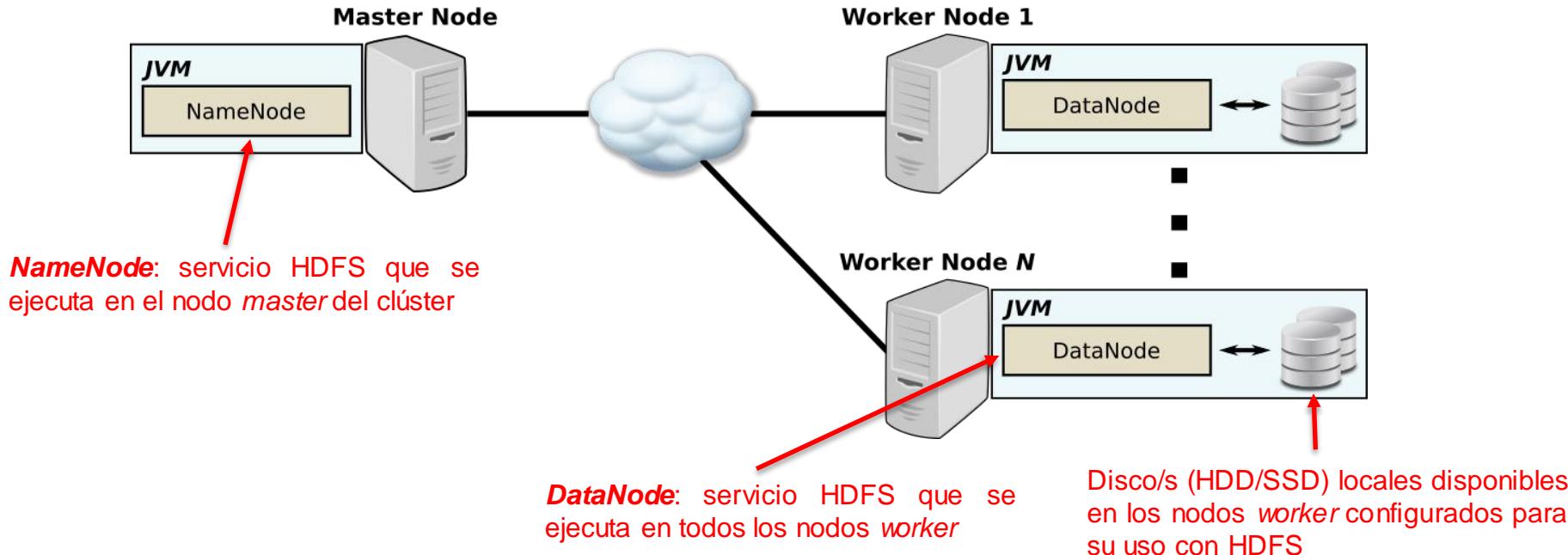
En el ejemplo de la figura se están ejecutando dos aplicaciones, una de ellas (la del cliente en color violeta) consta de un único contenedor (además de su *ApplicationMaster*), mientras que la otra (cliente rojo) consta de tres contenedores que se ejecutan en diferentes nodos del clúster



# HDFS

14

- Servicios principales del sistema de ficheros distribuido de Hadoop
  - Se ejecutan como **procesos Java**

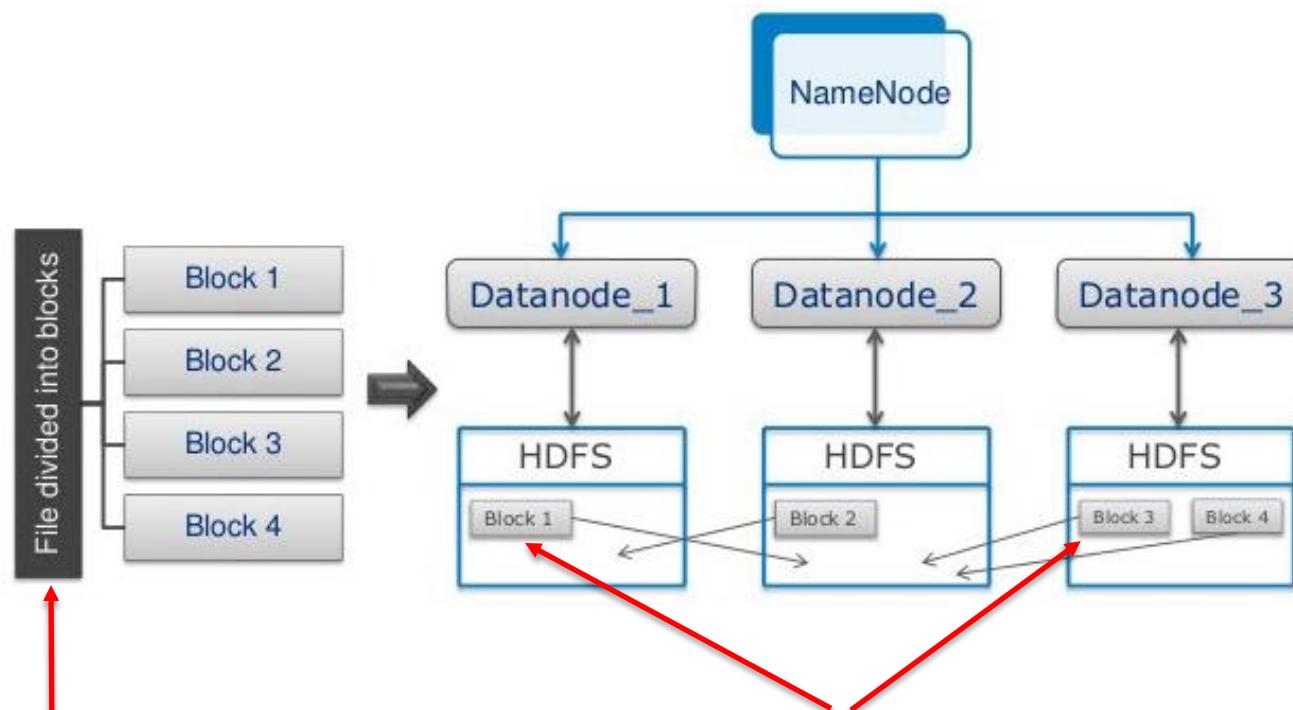




# HDFS

15

- Los ficheros almacenados en HDFS se **dividen** de forma **transparente** en un conjunto de **bloques** de tamaño fijo (p.e. 128 MiB)
  - El **tamaño de bloque** es configurable (fichero **hdfs-site.xml**)

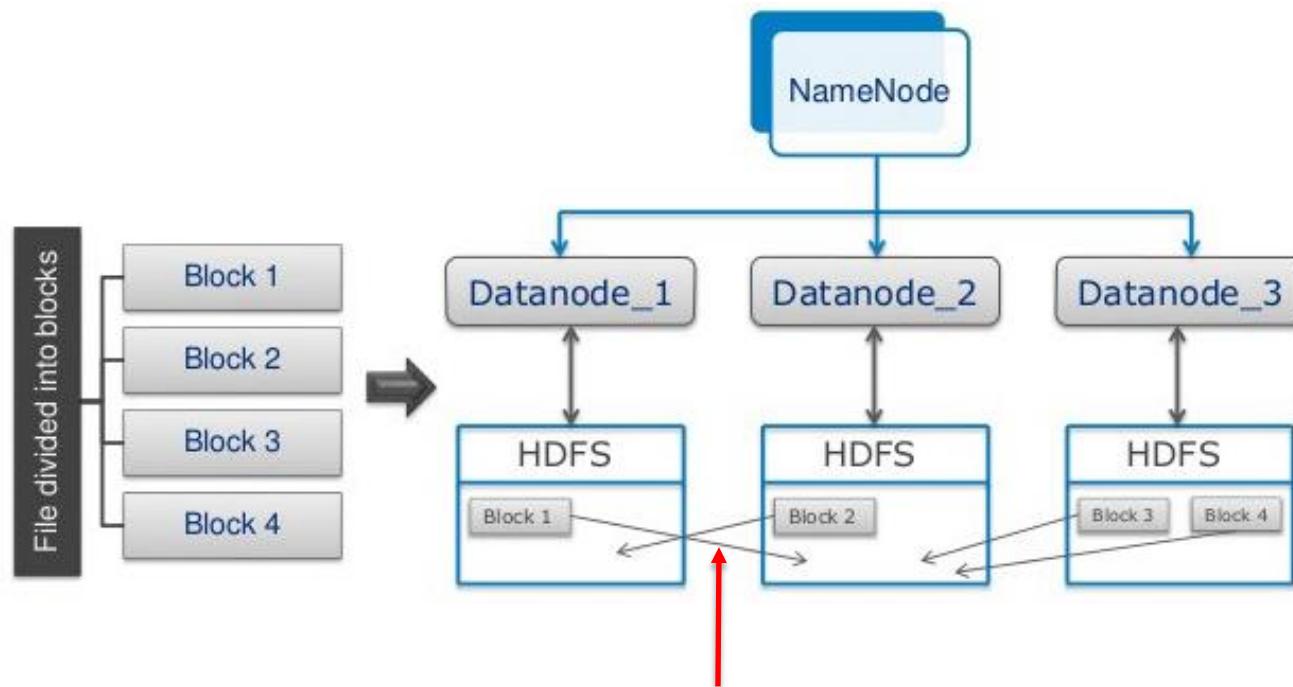


Por ejemplo, un fichero de 400 MiB de tamaño **almacenado en HDFS** estaría **dividido** en 4 bloques usando un tamaño de bloque de 128 MiB

Los bloques de datos del fichero se distribuyen entre los *DataNodes*. Estos bloques se almacenan en los discos de los nodos *worker* que hayan sido configurados para su uso con HDFS



- Los bloques de un fichero se **replican** en múltiples *DataNodes* para proporcionar **tolerancia a fallos en software**
  - El factor de replicación es configurable (fichero *hdfs-site.xml*)

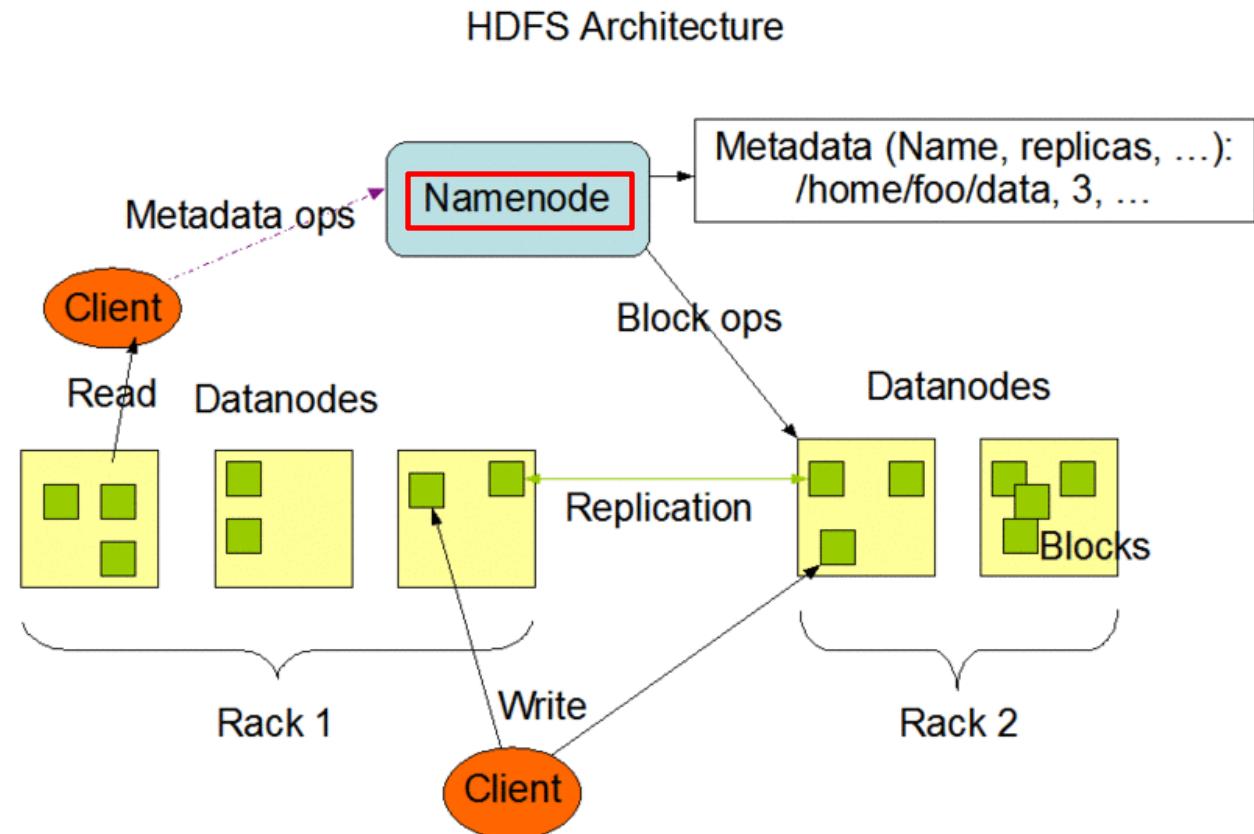


Las flechas de la figura indican en qué *DataNodes* se están replicando los bloques que componen el fichero. Por ejemplo, el primer bloque (*Block 1*) almacenado en *DataNode\_1* es replicado en *DataNode\_2*. Este ejemplo usa un factor de replicación de 2 (cada bloque tiene una única réplica). **Por defecto, HDFS usa factor de replicación 3**



## NameNode:

- Se encarga de mantener el espacio de nombres jerárquico del sistema de ficheros HDFS y gestiona sus **metadatos**.
- Recibe las operaciones a realizar (lectura/escritura) por parte de los clientes.
- Mapea el nombre de un fichero al conjunto de bloques de datos que lo componen.
- Mapea un bloque de datos de un fichero al conjunto de *DataNodes* que lo almacenan en local.
- Para alta disponibilidad, es posible ejecutar un **SecondaryNameNode** en clusters con más de un nodo *master*

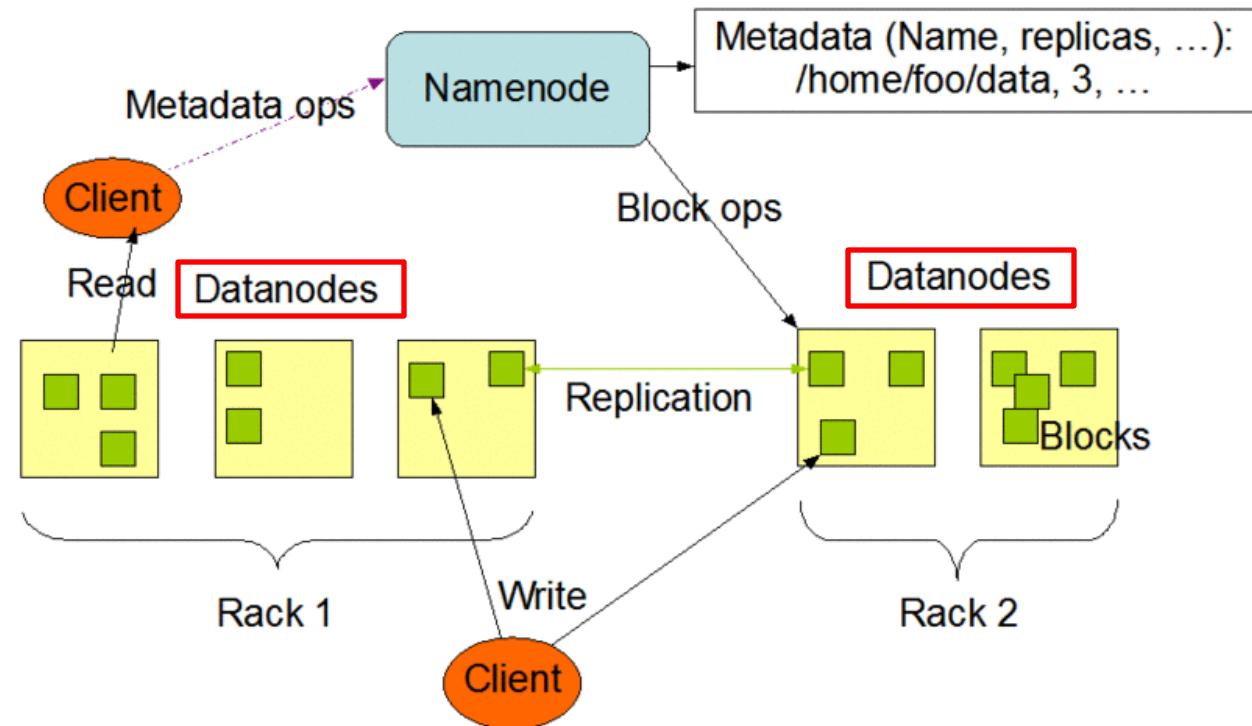




## DataNodes:

- Almacenan los bloques de datos en discos locales haciendo uso de sistema de ficheros existentes (p.e. EXT4, XFS).
- Cuando un cliente contacta con el *NameNode* para realizar una determinada operación, obtendrá la lista de *DataNodes* con los que debe contactar directamente para leer/escribir los datos.
- Se encargan también de replicar los bloques de datos de forma transparente para el cliente.

HDFS Architecture





# Contenidos

- **Objetivo**
- **Apache Hadoop**
- **Apache Spark**
- **Ejercicios propuestos**



# Apache Spark

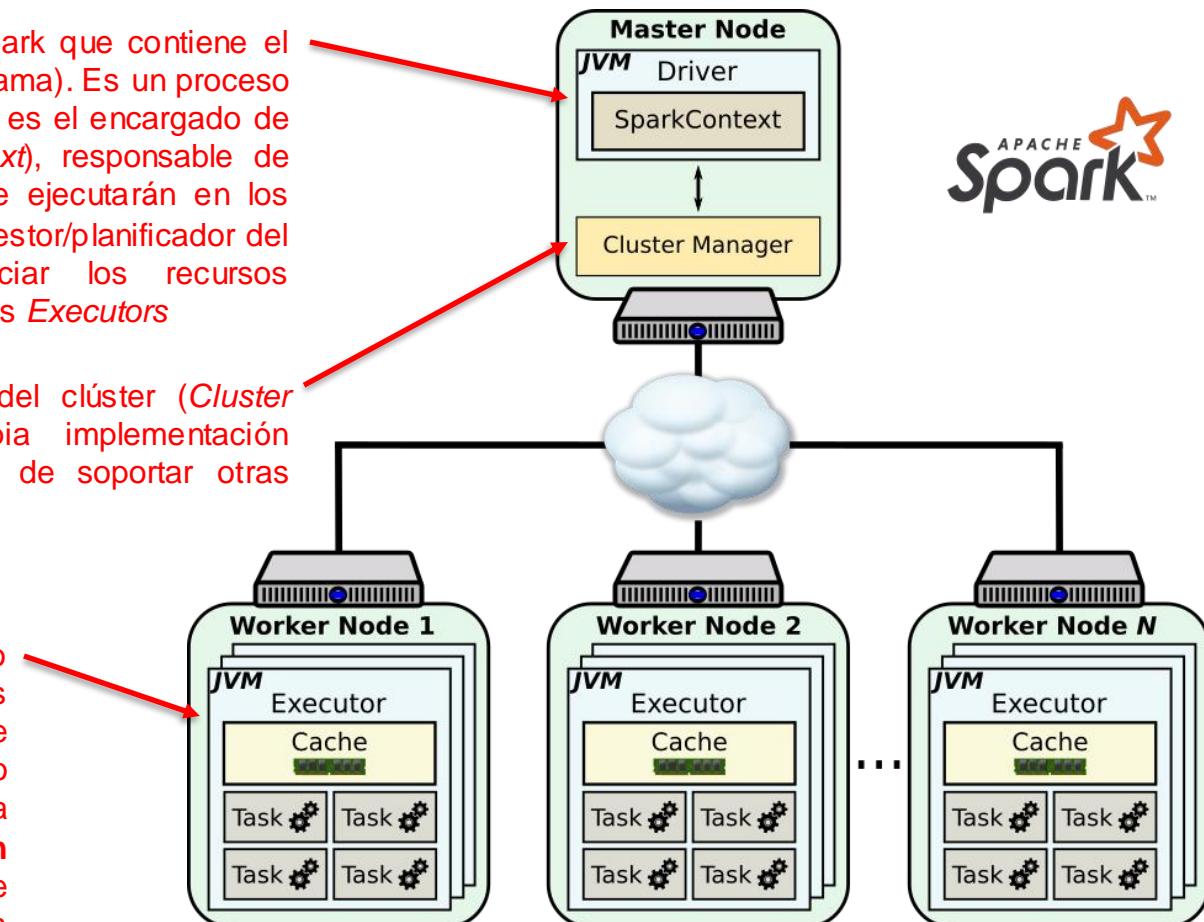
20

- Spark en modo clúster se basa en una arquitectura *master/worker*

El **driver** es el código de la aplicación Spark que contiene el método principal (punto de partida del programa). Es un proceso Java que suele residir en el nodo *master* y es el encargado de crear el **contexto de Spark** (`SparkContext`), responsable de crear las tareas de procesamiento que se ejecutarán en los **Executors**. El *driver* se comunica con un gestor/planificador del clúster (*Cluster Manager*) para negociar los recursos computacionales necesarios para ejecutar los *Executors*

Como **gestor de recursos/planificador** del clúster (*Cluster Manager*), Spark proporciona su propia implementación denominada **Spark Standalone**, además de soportar otras soluciones de terceros como **YARN**

Cada nodo *worker* del clúster ejecuta uno o más procesos Java llamados *Executors*, los cuales se encargan de ejecutar las tareas de procesamiento que reciben desde el nodo *master*. Pueden utilizar parte de la memoria RAM del nodo para **cachear datos en memoria** y acelerar así el procesamiento de los datos y el rendimiento de las aplicaciones

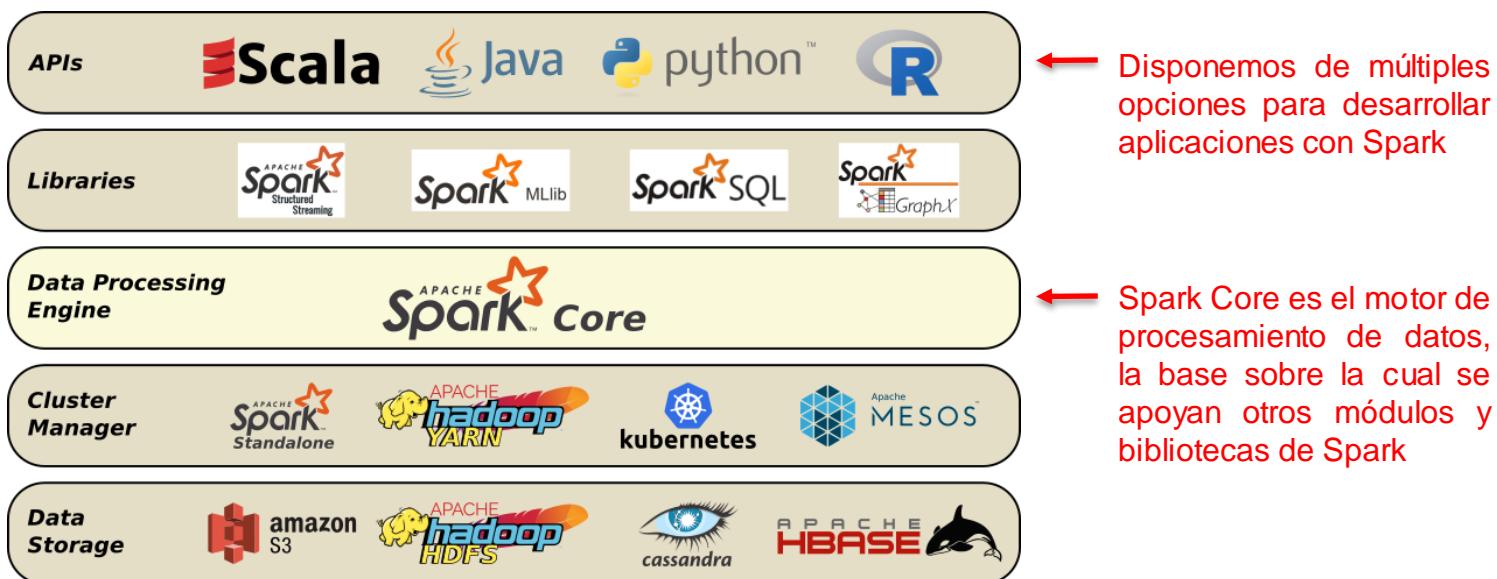




# Apache Spark

21

- Spark proporciona:
  - APIs de programación para diferentes lenguajes (Scala, Java, Python...)
  - Bibliotecas para *machine learning* (MLlib), procesamiento de datos estructurados mediante SQL, procesamiento de grafos (GraphX), procesamiento en modo *streaming*...
  - Soporte para diferentes *Cluster Managers* (Standalone, YARN, K8s...)
  - Integración con múltiples sistemas de almacenamiento (HDFS, HBase, S3...)

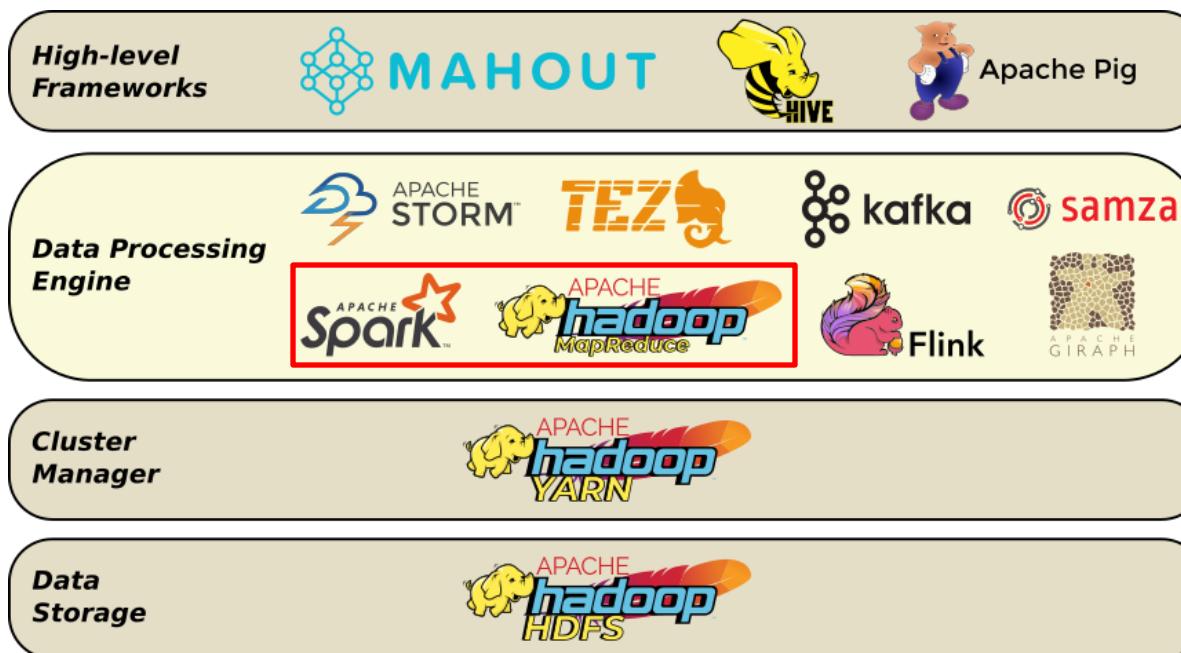




# Ecosistema Hadoop

22

- El ecosistema de proyectos que soportan YARN/HDFS como planificador de aplicaciones y almacenamiento de datos, respectivamente, es muy amplio
  - También existen bibliotecas de alto nivel implementadas sobre frameworks Big Data para proporcionar sus funcionalidades (p.e. Apache Hive)
  - En esta práctica, ejecutaremos la misma aplicación sobre YARN/HDFS simplemente cambiando el motor de procesamiento de datos: Hadoop MapReduce y Spark





# Contenidos

- **Objetivo**
- **Apache Hadoop**
- **Apache Spark**
- **Ejercicios propuestos**



# Ejercicio 1: Despliegue del clúster virtual

24

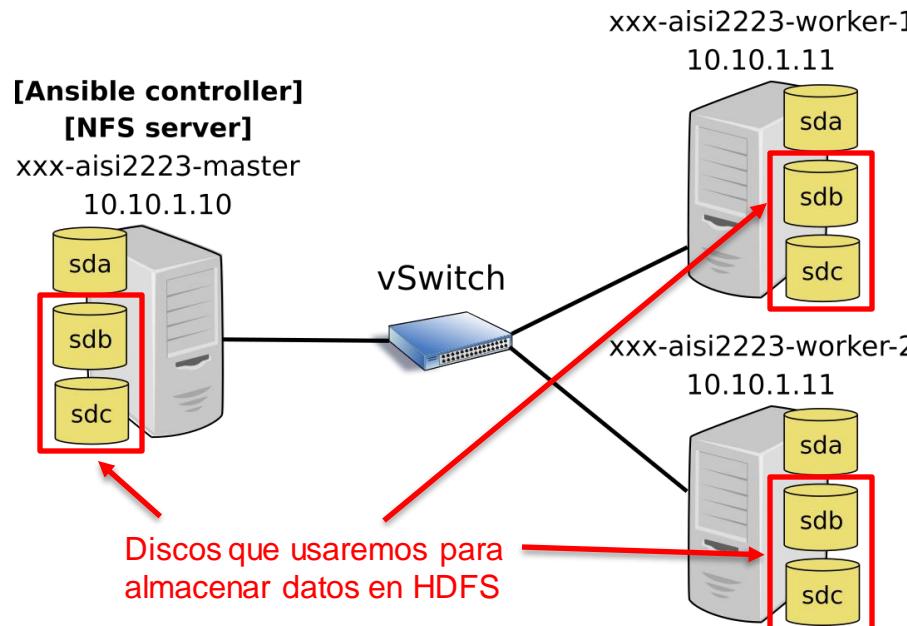
- Clona el [repositorio de la práctica 6](#) para obtener los ficheros necesarios para realizar los ejercicios propuestos
  - **Recuerda:** sin espacios, acentos, eñes o caracteres “raros” en la ruta 
- En cuanto a los recursos del clúster virtual, **no es necesario modificar** el `Vagrantfile` si tu *host* tiene al menos 8 GiB de memoria
  - Si dispones de menos recursos, ajusta la variable **NUM\_WORKERS** en el `Vagrantfile` (línea 23) para desplegar **un único nodo worker**
  - **NO cambies la memoria** de las VMs ya que eso implicaría modificar la configuración de Hadoop y Spark que se proporciona



# Ejercicio 1: Despliegue del clúster virtual

25

- El despliegue del clúster virtual está formado por:
  - Una VM funcionando como **nodo master, controlador Ansible y servidor NFS**
  - Dos VMs (o una) que harán las funciones de **nodos worker y clientes NFS**
- Todas las VMs disponen de 3 discos:
  - En el primer disco (sda) está instalado el SO (Debian Bullseye 11.6)
  - Los discos restantes (sdb y sdc) los usaremos para HDFS



Disponemos de conectividad entre las VMs a nivel de IP y *hostname*. La conectividad *ssh* desde el controlador Ansible hacia los nodos worker está configurado para hacerse sin introducir *password* (*ssh passwordless*)



# Ejercicio 1: Despliegue del clúster virtual

26

- **Modifica el Vagrantfile para cambiar el *hostname* de las VMs**
  - **Modifica las variables `MASTER_HOSTNAME` y `WORKER_HOSTNAME`**
  - **Debes sustituir "xxx" por tus iniciales correspondientes**
- **Edita también el fichero de inventario de Ansible (`ansible.inventory`) para sustituir "xxx" por tus iniciales correspondientes**
  - Solo si es necesario, ajusta el fichero en función de si tienes uno o dos nodos worker
- **Fíjate en los grupos que se definen en el inventario:**
  - El grupo ***masters*** incluye el nodo *master*
  - El grupo ***workers*** incluye los nodos *worker*
  - El grupo ***cluster*** está formado por los dos grupos anteriores

```
[masters]
rre-aisi2223-master
[workers]
rre-aisi2223-worker-1
rre-aisi2223-worker-2
# Group 'cluster' with all nodes
[cluster:children]
masters
workers
```





# Ejercicio 1: Despliegue del clúster virtual

27

- Despliega las VMs con Vagrant y al terminar conéctate al nodo **master**
  - `vagrant ssh` (o `vagrant ssh master`)
- Ejecuta los siguientes comandos en el **nodo master**:
  - `cat /etc/ansible/hosts`
  - `ansible cluster -m ping`
  - `ansible cluster -m shell -a "df -h | grep /data"`
- Con eso comprobaremos que:
  - El acceso ssh passwordless funciona y que todos los nodos del clúster son accesibles y responden a las peticiones desde el controlador de Ansible
  - Todos los nodos disponen de los discos sdb/sdc correctamente formateados y montados bajo la ruta `/data`
- Tanto la configuración ssh passwordless como la gestión de los discos se realizó en el aprovisionamiento global durante el despliegue de las VMs
  - Puedes echar un vistazo al script shell: `provisioning/bootstrap.sh`



# Ejercicio 1: Despliegue del clúster virtual

28



Fichero de inventario de Ansible en el controlador (nodo master)



```
vagrant@rre-aisi2223-master:~$ cat /etc/ansible/hosts
[masters]
rre-aisi2223-master
[workers]
rre-aisi2223-worker-1
rre-aisi2223-worker-2
# Group 'cluster' with all nodes
[cluster:children]
masters
workers
vagrant@rre-aisi2223-master:~$ ansible cluster -m ping
rre-aisi2223-worker-2 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
rre-aisi2223-worker-1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
rre-aisi2223-master | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
vagrant@rre-aisi2223-master:~$ ansible cluster -m shell -a "df -h | grep /data"
rre-aisi2223-worker-1 | CHANGED | rc=0 >>
/dev/sdb      9.8G  24K  9.3G  1% /data/disk0
/dev/sdc      9.8G  24K  9.3G  1% /data/disk1
rre-aisi2223-worker-2 | CHANGED | rc=0 >>
/dev/sdb      9.8G  24K  9.3G  1% /data/disk0
/dev/sdc      9.8G  24K  9.3G  1% /data/disk1
rre-aisi2223-master | CHANGED | rc=0 >>
/dev/sdb      9.8G  24K  9.3G  1% /data/disk0
/dev/sdc      9.8G  24K  9.3G  1% /data/disk1
vagrant@rre-aisi2223-master:~$
```

Discos formateados y montados en todos los nodos del clúster. Fíjate en su ocupación actual y el espacio libre disponible





## Ejercicio 2: Configuración de NFS

29

- La instalación de ambos *frameworks* Big Data (Hadoop y Spark) se realizará únicamente en el nodo *master* del clúster
  - El directorio del nodo *master* donde se instalen ambos *frameworks* será exportado al resto de nodos del clúster utilizando **NFS**
  - Esta aproximación es muy común en entornos clúster, de forma que es posible instalar *software* en un único nodo y “compartirlo” con el resto del clúster, simplificando así tanto su gestión como su configuración
- Por tanto, el nodo *master* actuará como **servidor NFS** mientras que los nodos *worker* actuarán como **clientes NFS**
  - Los clientes deberán montar el directorio exportado por el servidor en una ruta local para poder acceder al *software* (en este caso, Hadoop y Spark)
- La instalación y configuración del servidor y los clientes NFS se realizará usando **Ansible Playbooks**



# Ejercicio 2: Configuración de NFS

30

- En el directorio **ansible/nfs** dispones de:
  - **main.yml**: Playbook principal para configurar NFS que hace uso de otros tres playbooks **que deberás completar**:
    - **playbooks/nfs-common.yml**: tareas comunes a servidor y clientes
    - **playbooks/nfs-server.yml**: tareas específicas para configurar el servidor
    - **playbooks/nfs-client.yml**: tareas específicas para configurar los clientes
  - **playbooks/vars/main.yml**: fichero YAML que define las variables necesarias
    - Por ejemplo, la variable **nfs\_server** define el *hostname* del servidor NFS (nodo *master*), mientras que la variable **nfs\_dir** indica el directorio del servidor NFS que se va a exportar/compartir
  - **playbooks/templates/exports.j2**: plantilla Jinja2 del fichero */etc/exports* de configuración del servidor NFS, que especifica el directorio a exportar junto con las opciones necesarias (p.e. la red a la que se exporta)



# Ejercicio 2: Configuración de NFS

31

- **Playbook `nfs-common.yml`:**
  - **Este playbook debe ejecutarse en todos los nodos del clúster**
    - Usa el grupo **cluster** definido en el fichero de inventario
  - **Se compone de dos tareas (DEBES completar la SEGUNDA):**
    1. Usando el módulo **package**, instala el paquete **nfs-common**
    2. Usando el módulo **file**, crea un directorio usando como parámetro **path** la variable **nfs\_dir** definida en **vars/main.yml**
      - Este es el directorio que será exportado por el servidor NFS (nodo *master*) al resto de nodos del clúster, y donde instalaremos Hadoop y Spark
      - A su vez, los clientes NFS (nodos *worker*) usarán esta ruta para montar localmente el directorio exportado por el servidor (esto se hará en una tarea posterior)
      - El propietario y grupo del nuevo directorio debe ser **root**
      - Establece permisos **1777** (parámetro **mode**) al directorio creado



# Ejercicio 2: Configuración de NFS

32

- **Playbook `nfs-server.yml`:**
  - **Este playbook debe ejecutarse solo en el nodo master**
    - Usa el grupo **masters** definido en el fichero de inventario
  - Se compone de **tres tareas** (**DEBES completar la TERCERA**):
    1. Usando el módulo **package**, instala el paquete **nfs-kernel-server**
    2. Usando el módulo **service**, inicia el servicio NFS (**nfs-kernel-server**) y habilita su arranque automático en el inicio del sistema
    3. Usando el módulo **template**, copia la plantilla **templates/exports.j2** en la ruta **/etc(exports**
      - Como resultado de esta tarea se ha modificado la configuración del servicio NFS, por lo que esta tarea debe **notificar al handler** ya definido en el playbook



# Ejercicio 2: Configuración de NFS

33

- **Playbook *nfs-client.yml*:**
  - **Este playbook debe ejecutarse en todos los nodos *worker***
    - Usa el grupo **workers** definido en el fichero de inventario
  - Se compone de una **única tarea** que **DEBES completar**:
    1. Usando el módulo **[mount](#)**, monta el directorio exportado por el servidor NFS en la ruta local de los nodos *worker* definida por la variable ***nfs\_dir*** (parámetro ***path*** del módulo)
      - El tipo de sistema de ficheros a montar (parámetro ***fstype***) es ***nfs***
      - Como origen (parámetro ***src***) usa la variable ***nfs\_server*** para indicar el ***hostname*** del servidor NFS y la variable ***nfs\_dir*** para indicar el directorio
        - Sintaxis del parámetro ***src***: "*hostname:directory*"
        - Mira [ejemplos de uso](#) del módulo *mount* para el caso de NFS
      - En opciones de montaje (parámetro ***opts***) especifica "***rw, sync***"
      - Asegúrate que usas el valor adecuado del parámetro ***state*** para que el directorio se monte automáticamente durante el arranque del sistema
        - Es decir, la tarea debe encargarse de añadir la entrada correspondiente para su montaje automático en el fichero ***/etc/fstab***



# Ejercicio 2: Configuración de NFS



34

- Cuando completes los tres playbooks anteriores, ejecuta el playbook principal de configuración de NFS en el nodo *master*
  - *ansible-playbook /vagrant/ansible/nfs/main.yml*
- Antes de continuar, ejecuta los siguientes comandos en el nodo *master*:
  - *sudo exportfs*
  - *ansible workers -a "cat /proc/mounts" | grep "share/nfs"*
- Así comprobaremos que la configuración NFS es correcta: el servidor (nodo *master*) exporta el directorio y los clientes (nodos *worker*) lo montan en una ruta local

```
vagrant@rre-aisi2223-master:~$ sudo exportfs
/share/nfs      10.10.1.10/24
vagrant@rre-aisi2223-master:~$
vagrant@rre-aisi2223-master:~$ ansible workers -a "cat /proc/mounts" | grep "share/nfs"
rre-aisi2223-worker:/share/nfs /share/nfs nfs4 rw,sync,relatime,vers=4.2,rsize=262144,wsize=262144,namlen=255,hard,proto=tcp
1,local_lock=none,addr=10.10.1.10 0 0
rre-aisi2223-worker:/share/nfs /share/nfs nfs4 rw,sync,relatime,vers=4.2,rsize=262144,wsize=262144,namlen=255,hard,proto=tcp
2,local_lock=none,addr=10.10.1.10 0 0
vagrant@rre-aisi2223-master:~$
```



## Ejercicio 3: Instalación de Hadoop

35

- La instalación de Apache Hadoop se realizará en el nodo *master* usando el directorio compartido por NFS que hemos configurado previamente
- Además, como dependencia de Hadoop, debemos instalar la máquina virtual de Java (JVM) en todos los nodos del clúster
  - Hadoop está implementado en Java y ofrece su principal API para el desarrollo de aplicaciones MapReduce en dicho lenguaje
  - Spark está implementado en Scala, un lenguaje que se ejecuta en la JVM, pero ofrece APIs de desarrollo para otros lenguajes además de Scala (p.e. Java, Python, R...)
  - Por tanto, la instalación de la JVM en el clúster es una dependencia común a ambos *frameworks*
- De forma similar a NFS, usaremos **Ansible Playbooks** para automatizar la instalación de la JVM y de Hadoop



# Ejercicio 3: Instalación de Hadoop

36

- En el directorio **ansible/hadoop** dispones de:
  - **main.yml**: Playbook principal que instala Java y Hadoop usando tres playbooks adicionales **que deberás completar**:
    - **playbooks/java.yml**: tareas para instalar la JVM en todos los nodos
    - **playbooks/hadoop-user.yml**: tareas para crear y configurar un usuario para el uso de Hadoop en todos los nodos
    - **playbooks/hadoop.yml**: tareas para descargar, instalar y configurar Hadoop en el nodo *master*
    - **playbooks/spark.yml**: tareas para descargar, instalar y configurar Spark en el nodo *master*
      - Por ahora puedes ignorar este playbook, se usará en un ejercicio posterior



# Ejercicio 3: Instalación de Hadoop

37

- En el directorio **ansible/hadoop** dispones de (cont.):
  - **playbooks/vars/main.yml**: fichero YAML que define las variables necesarias
    - Por ejemplo, se definen variables con las URLs de descarga de Hadoop y Spark, así como variables con las versiones concretas de los frameworks que usaremos
    - También se definen variables relacionadas con los recursos de los nodos *worker* que son necesarias para la configuración de los frameworks
    - La variable **hadoop\_user\_passwd** define el password encriptado que se va a configurar para el usuario hadoop cuando sea creado por Ansible
      - El password es igual al nombre del usuario (es decir, hadoop)
  - **playbooks/templates**: directorio con las plantillas Jinja2 para la configuración de Hadoop y Spark
    - Entre otros, el fichero **hdfs-site.xml** especifica la configuración de HDFS (p.e. tamaño de bloque, factor de replicación, rutas locales que utilizan los nodos para almacenar los datos), mientras que el fichero **yarn-site.xml** especifica la configuración de YARN (p.e. recursos de los nodos)
    - El fichero **workers** lista el *hostname* de los nodos *worker* del clúster
    - El fichero **spark-defaults.conf** configura los parámetros más básicos Spark



# Ejercicio 3: Instalación de Hadoop

38

- **Playbook `java.yml` (**NO** es necesario modificarlo):**
  - **Este playbook se ejecuta en todos los nodos del clúster**
    - Usa el grupo `cluster` definido en el fichero de inventario
  - Se compone de una **Única tarea**:
    - Usando el módulo `package`, instala el paquete `openjdk-11-jdk`
- **Playbook `hadoop-user.yml`:**
  - **Este playbook debe ejecutarse en todos los nodos del clúster**
    - Usa el grupo `cluster` definido en el fichero de inventario
  - Se compone de **seis tareas** (**DEBES completar únicamente la SEGUNDA**):
    - Usando el módulo `user`, crea un usuario `hadoop` que pertenezca al grupo `hadoop`
      - Configura su directorio de usuario (parámetro `home`) en `/home/hadoop` y su `shell` a `/bin/bash`
      - Como `password` usa la variable `hadoop_user_passwd` definida en `vars/main.yml`
      - Configura la generación de claves ssh de tipo RSA de 2048 bits
        - Ver parámetros del módulo: `generate_ssh_key`, `ssh_key_type`, `ssh_key_bits`



# Ejercicio 3: Instalación de Hadoop

39

- **Playbook `hadoop.yml`:**

- **Este playbook debe ejecutarse solo en el nodo master**
  - Usa el grupo **masters** definido en el fichero de inventario
- **Se compone de cinco tareas (DEBES completar la 2<sup>a</sup>, 3<sup>a</sup> y 4<sup>a</sup>):**
  - Usando el módulo **get\_url**, descarga el *tarball* de la distribución de Hadoop
    - Usa la variable **hadoop\_url** como enlace de descarga (parámetro **url**)
    - Usa la variable **download\_dir** como destino de la misma (parámetro **dest**)
    - El nombre del fichero descargado debe ser *hadoop.tar.gz* (parámetro **dest**)
    - El propietario y grupo del fichero descargado debe ser *hadoop*
  - Usando el módulo **unarchive**, descomprime el *tarball* de Hadoop (*hadoop.tar.gz*) usando como origen (parámetro **src**) el parámetro **dest** de la tarea previa
    - Usa la variable **hadoop\_dir** como directorio destino de la descompresión (parámetro **dest**)
    - El propietario y grupo debe ser *hadoop*
  - Usando el módulo **template**, copia los ficheros de configuración de Hadoop usando la variable **hadoop\_conf\_dir** como destino (parámetro **dest**)
    - El propietario y grupo de los ficheros debe ser *hadoop*
    - Debes hacer uso del parámetro **loop** ya definido



# Ejercicio 3: Instalación de Hadoop

40

- Cuando completes los tres playbooks anteriores, ejecuta el playbook principal de Hadoop en el nodo *master*:
  - `ansible-playbook /vagrant/ansible/hadoop/main.yml`
- Antes de continuar, ejecuta los siguientes comandos en el nodo *master* para comprobar la correcta instalación de Java y Hadoop:
  - `ansible cluster -a "java -version"`
  - `ansible xxx-aisi2223-worker-1 -m shell -a "ls /share/nfs/hadoop/hadoop-*"`
- A continuación, ejecuta los siguientes comandos en el nodo *master* para comprobar la correcta configuración del usuario *hadoop*:
  - `su -l hadoop`
  - `echo $PATH && echo $HADOOP_HOME`
  - `tail -3 $HADOOP_HOME/etc/hadoop/hadoop-env.sh`
  - `hadoop version`
- **Salvo que se indique lo contrario, desde ahora debes usar el usuario hadoop**



# Ejercicio 3: Instalación de Hadoop

41

```
vagrant@rre-aisi2223-master:~$ ansible cluster -a "java -version"
rre-aisi2223-worker-1 | CHANGED | rc=0 >>
openjdk version "11.0.18" 2023-01-17
OpenJDK Runtime Environment (build 11.0.18+10-post-Debian-1deb11u1)
OpenJDK 64-Bit Server VM (build 11.0.18+10-post-Debian-1deb11u1, mixed mode, sharing)
rre-aisi2223-worker-2 | CHANGED | rc=0 >>
openjdk version "11.0.18" 2023-01-17
OpenJDK Runtime Environment (build 11.0.18+10-post-Debian-1deb11u1)
OpenJDK 64-Bit Server VM (build 11.0.18+10-post-Debian-1deb11u1, mixed mode, sharing)
rre-aisi2223-master | CHANGED | rc=0 >>
openjdk version "11.0.18" 2023-01-17
OpenJDK Runtime Environment (build 11.0.18+10-post-Debian-1deb11u1)
OpenJDK 64-Bit Server VM (build 11.0.18+10-post-Debian-1deb11u1, mixed mode, sharing)
vagrant@rre-aisi2223-master:~$ ansible rre-aisi2223-worker-1 -m shell -a "ls /share/nfs/hadoop/hadoop-*"
rre-aisi2223-worker-1 | CHANGED | rc=0 >>
LICENSE-binary
LICENSE.txt
NOTICE-binary
NOTICE.txt
README.txt
bin
etc
include
lib
libexec
licenses-binary
sbin
share
vagrant@rre-aisi2223-master:~$
```

Conjunto de fichero y directorios que componen Hadoop





# Ejercicio 3: Instalación de Hadoop

42



Estamos "logueados" como usuario hadoop

```
vagrant@rre-aisi2223-master:~$ su -l hadoop  
Password:  
hadoop@rre-aisi2223-master:~$ echo $PATH && echo $HADOOP_HOME  
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/share/nfs/hadoop/hadoop-3.3.4/bin:/share/nfs/  
/share/nfs/hadoop/hadoop-3.3.4  
hadoop@rre-aisi2223-master:~$ tail -3 $HADOOP_HOME/etc/hadoop/hadoop-env.sh  
# server jvm.  
# export HADOOP_REGISTRYDNS_SECURE_EXTRA_OPTS="-jvm server"  
export JAVA_HOME=/usr/lib/jvm/java-1.11.0-openjdk-amd64  
hadoop@rre-aisi2223-master:~$ hadoop version  
Hadoop 3.3.4  
Source code repository https://github.com/apache/hadoop.git -r a585a73c3e02ac62350c136643a5e7f6095a3dbb  
Compiled by stevel on 2022-07-29T12:32Z  
Compiled with protoc 3.7.1  
From source with checksum fb9dd8918a7b8a5b430d61af858f6ec  
This command was run using /share/nfs/hadoop/hadoop-3.3.4/share/hadoop/common/hadoop-common-3.3.4.jar  
hadoop@rre-aisi2223-master:~$
```

Recuerda que el *password* del usuario hadoop se ha configurado como hadoop



El PATH está correctamente configurado, podemos ejecutar un comando *hadoop* sin necesidad de especificar la ruta absoluta al binario



# Ejercicio 4: Despliegue de Hadoop

43

- Llegados a este punto, dispones de un clúster Hadoop totalmente configurado que debería ser plenamente funcional
- En este paso, iniciaremos los servicios necesarios para desplegar Hadoop y sus componentes en los nodos del clúster
- Antes debes comprobar que el usuario hadoop puede conectarse por ssh desde el *master* a los nodos *worker* sin necesidad de introducir password

```
hadoop@rre-aisi2223-master:~$ ssh rre-aisi2223-worker-1
The authenticity of host 'rre-aisi2223-worker-1 (10.10.1.11)' can't be established.
ECDSA key fingerprint is SHA256:Xw/YrlzR7bXuMtJLV4WDG7TQ50XR3s42X+mYNWTzgkk.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'rre-aisi2223-worker-1,10.10.1.11' (ECDSA) to the list of known hosts.
Linux rre-aisi2223-worker-1 5.10.0-20-amd64 #1 SMP Debian 5.10.158-2 (2022-12-13) x86_64
```

The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/\*/\*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.

```
hadoop@rre-aisi2223-worker-1:~$ exit
logout
Connection to rre-aisi2223-worker-1 closed.
hadoop@rre-aisi2223-master:~$
```

Conexión ssh *passwordless* desde el nodo master a uno de los nodos *worker*. Comprueba también la conectividad ssh con el segundo nodo *worker*



# Ejercicio 4: Despliegue de Hadoop

44

- La operativa básica de Hadoop, una vez instalado y configurado, se indica en esta parte concreta de la documentación:
  - [https://hadoop.apache.org/docs/r3.3.4/hadoop-project-dist/hadoop-common/ClusterSetup.html#Operating the Hadoop Cluster](https://hadoop.apache.org/docs/r3.3.4/hadoop-project-dist/hadoop-common/ClusterSetup.html#Operating_the_Hadoop_Cluster)
- Tal y como se indica en la documentación, **antes de desplegar HDFS por primera vez debemos darle un formato**
  - Ejecuta en el nodo *master*: *hdfs namenode -format*
- A continuación, **inicia los servicios HDFS y YARN**
  - Tal y como se recomienda en la documentación, es preferible que uses los *scripts* proporcionados por Hadoop para dicho propósito
    - *start-dfs.sh*
    - *start-yarn.sh*
  - Estos *scripts* leen el fichero *workers* que hemos configurado previamente para saber en qué nodos deben iniciarse los servicios correspondientes



# Ejercicio 4: Despliegue de Hadoop

45

- Parte final de la salida del comando de formateo de HDFS

```
2023-02-15 12:21:00,993 INFO namenode.FSImage: Allocated new BlockPoolId: BP-445278276-10.10.1.10-1676463660979
2023-02-15 12:21:01,036 INFO common.Storage: Storage directory /data/disk0/hadoop-data/namenode has been successfully formatted.
2023-02-15 12:21:01,390 INFO common.Storage: Storage directory /data/disk1/hadoop-data/namenode has been successfully formatted.
2023-02-15 12:21:01,462 INFO namenode.FSImageFormatProtobuf: Saving image file /data/disk0/hadoop-data/namenode/current/fsimage.
2023-02-15 12:21:01,463 INFO namenode.FSImageFormatProtobuf: Saving image file /data/disk1/hadoop-data/namenode/current/fsimage.
2023-02-15 12:21:01,603 INFO namenode.FSImageFormatProtobuf: Image file /data/disk1/hadoop-data/namenode/current/fsimage.ckpt_00
0 seconds .
2023-02-15 12:21:01,604 INFO namenode.FSImageFormatProtobuf: Image file /data/disk0/hadoop-data/namenode/current/fsimage.ckpt_00
0 seconds .
2023-02-15 12:21:01,625 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
2023-02-15 12:21:01,655 INFO namenode.FSNamesystem: Stopping services started for active state
2023-02-15 12:21:01,655 INFO namenode.FSNamesystem: Stopping services started for standby state
2023-02-15 12:21:01,662 INFO namenode.FSImage: FSImageSaver clean checkpoint: txid=0 when meet shutdown.
2023-02-15 12:21:01,662 INFO namenode.FSImage: FSImageSaver clean checkpoint: txid=0 when meet shutdown.
2023-02-15 12:21:01,663 INFO namenode.NameNode: SHUTDOWN_MSG:
*****
SHUTDOWN_MSG: Shutting down NameNode at rre-aisi2223-master/10.10.1.10
*****  
hadoop@rre-aisi2223-master:~$
```



## Asegúrate de que el formateo de HDFS se realiza sin errores.

Los mensajes resaltados en la figura superior indican que todo ha ido correctamente. De ser así, procede a ejecutar los *scripts* de Hadoop mencionados en la transparencia previa para iniciar los servicios HDFS y YARN



# Ejercicio 4: Despliegue de Hadoop

46



- Tras iniciar HDFS y YARN, ejecuta el siguiente comando en el nodo *master*
  - *ansible cluster -a "jps"*
  - Nota: el comando *jps* lista las JVMs en ejecución del usuario actual y sus PIDs
- Así comprobaremos que están ejecutándose los servicios correspondientes en cada nodo del clúster



Es importante que revises las transparencias 10 y 15 si no recuerdas qué son y qué hacen estos procesos Java y en qué nodo deben ejecutarse



Ten en cuenta que si reinicias o apagas/enciendes las VMs del clúster tendrás que volver a ejecutar estos *scripts* para iniciar los servicios. En esta práctica, suspender las VMs (*vagrant suspend*) con los servicios en ejecución en vez de detenerlas resulta más cómodo

```
hadoop@rre-aisi2223-master:~$ start-dfs.sh
Starting namenodes on [rre-aisi2223-master]
Starting datanodes
Starting secondary namenodes [rre-aisi2223-master]
hadoop@rre-aisi2223-master:~$
hadoop@rre-aisi2223-master:~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hadoop@rre-aisi2223-master:~$
hadoop@rre-aisi2223-master:~$ ansible cluster -a "jps"
rre-aisi2223-worker-2 | CHANGED | rc=0 >>
8555 Jps
8301 DataNode
8415 NodeManager
rre-aisi2223-worker-1 | CHANGED | rc=0 >>
8662 Jps
8408 DataNode
8522 NodeManager
rre-aisi2223-master | CHANGED | rc=0 >>
13776 Jps
13064 NameNode
13420 ResourceManager
13245 SecondaryNameNode
hadoop@rre-aisi2223-master:~$
```



# Ejercicio 5: Ejecución de Hadoop WordCount

47

- En este ejercicio ejecutaremos una aplicación MapReduce sencilla (*WordCount*) que viene incluida en la distribución de Hadoop
  - *WordCount* cuenta el número de ocurrencias de cada palabra en un conjunto de datos de entrada de tipo texto
- Además, conoceremos los comandos más básicos para **operar con HDFS**
  - La aplicación *WordCount* procesará como entrada un fichero de texto que almacenaremos en HDFS y escribirá su salida también en HDFS
- Para operar con HDFS, Hadoop proporciona el comando *hdfs*:
  - Sintaxis general: ***hdfs dfs -OPERACIÓN [OPCIONES] [RUTA]***
  - Comando sinónimo: ***hadoop fs -OPERACIÓN [OPCIONES] [RUTA]***
- Veamos un primer ejemplo de uso con la operación *df* que muestra información básica acerca del sistema de ficheros HDFS

```
hadoop@rre-aisi2223-master:~$ hdfs dfs -df -h
Filesystem          Size  Used Available  Use%
hdfs://rre-aisi2223-master:9000  39.0 G  128 K    36.9 G   0%
hadoop@rre-aisi2223-master:~$
```



# Ejercicio 5: Ejecución de Hadoop WordCount

48

- Descarga en el nodo **master** el conjunto de datos de entrada que procesará WordCount, que consiste en un único fichero en formato XML
  - `wget https://gac.udc.es/~rober/aisi/file.xml`
- Crea un **nuevo directorio** en **HDFS** con nombre: **/hadoop-input**
  - Consulta la ayuda de la operación de HDFS: [mkdir](#)
- Copia el fichero XML **file.xml** desde el sistema de ficheros local del nodo **master** (donde lo tenemos descargado) al directorio HDFS creado en el paso anterior
  - Consulta la ayuda de la operación de HDFS: [put](#)
- Después de copiar el fichero a HDFS, ejecuta los siguientes comandos:
  - `ls -lh $HOME`
  - `hdfs dfs -ls -R -h /`
  - `ansible workers -m shell -a "df -h | grep /data"`
  - `hdfs dfs -df -h`



Muchas de las operaciones de HDFS se "inspiran" en comandos de la *shell bash* (*df*, *ls*, *mkdir*, *mv*, *rm*...). Para saber más sobre todas las operaciones disponibles, puedes consultar este [enlace](#).



# Ejercicio 5: Ejecución de Hadoop WordCount



49

```
hadoop@rre-aisi2223-master:~$ ls -lh $HOME
total 453M
-rw-r--r-- 1 hadoop hadoop 453M Jan 13 2022 file.xml
hadoop@rre-aisi2223-master:~$ hdfs dfs -ls -R -h /
drwxr-xr-x - hadoop supergroup          0 2023-02-16 09:29 /hadoop-input
-rw-r--r-- 2 hadoop supergroup 452.5 M 2023-02-16 09:29 /hadoop-input/file.xml
hadoop@rre-aisi2223-master:~$
hadoop@rre-aisi2223-master:~$ ansible workers -m shell -a "df -h | grep /data"
rre-aisi2223-worker-2  CHANGED | rc=0 >>
/dev/sdb              9.8G 231M 9.1G 3% /data/disk0
/dev/sdc              9.8G 226M 9.1G 3% /data/disk1
rre-aisi2223-worker-1  CHANGED | rc=0 >>
/dev/sdc              9.8G 226M 9.1G 3% /data/disk1
/dev/sdb              9.8G 231M 9.1G 3% /data/disk0
hadoop@rre-aisi2223-master:~$ 
hadoop@rre-aisi2223-master:~$ hdfs dfs -df -h
Filesystem           Size  Used   Available  Use%
hdfs://rre-aisi2223-master:9000 39.0 G 912.3 M 36.0 G 2%
hadoop@rre-aisi2223-master:~$
```

- Relaciona el tamaño del fichero **file.xml** con:
- El espacio ocupado que observas en los dos discos de los nodos worker
- El espacio usado en HDFS
- El tamaño de bloque (**dfs.blocksize**) y el factor de replicación (**dfs.replication**) configurados para HDFS
  - Ruta del fichero de configuración de HDFS:
    - `/share/nfs/hadoop/hadoop-3.3.4/etc/hadoop/hdfs-site.xml`



# Ejercicio 5: Ejecución de Hadoop WordCount

50

- Examina en detalle el fichero **file.xml** para obtener información de sus bloques en HDFS ejecutando:
  - **hdfs fsck /hadoop-input/file.xml -files -blocks**

```
hadoop@rre-aisi2223-master:~$ hdfs fsck /hadoop-input/file.xml -files -blocks
Connecting to namenode via http://rre-aisi2223-master:9870/fsck?ugi=hadoop&files=1&blocks=1&path=%2Fhadoop-input%2Ffile.xml
FSCK started by hadoop (auth:SIMPLE) from /10.10.1.10 for path /hadoop-input/file.xml at Thu Feb 16 09:38:44 UTC 2023
```

```
/hadoop-input/file.xml 474510285 bytes, replicated: replication=2, 15 block(s):  OK
0. BP-445278276-10.10.1.10-1676463660979:blk_1073741825_1001 len=33554432 Live_repl=2
1. BP-445278276-10.10.1.10-1676463660979:blk_1073741826_1002 len=33554432 Live_repl=2
2. BP-445278276-10.10.1.10-1676463660979:blk_1073741827_1003 len=33554432 Live_repl=2
3. BP-445278276-10.10.1.10-1676463660979:blk_1073741828_1004 len=33554432 Live_repl=2
4. BP-445278276-10.10.1.10-1676463660979:blk_1073741829_1005 len=33554432 Live_repl=2
5. BP-445278276-10.10.1.10-1676463660979:blk_1073741830_1006 len=33554432 Live_repl=2
6. BP-445278276-10.10.1.10-1676463660979:blk_1073741831_1007 len=33554432 Live_repl=2
7. BP-445278276-10.10.1.10-1676463660979:blk_1073741832_1008 len=33554432 Live_repl=2
8. BP-445278276-10.10.1.10-1676463660979:blk_1073741833_1009 len=33554432 Live_repl=2
9. BP-445278276-10.10.1.10-1676463660979:blk_1073741834_1010 len=33554432 Live_repl=2
10. BP-445278276-10.10.1.10-1676463660979:blk_1073741835_1011 len=33554432 Live_repl=2
11. BP-445278276-10.10.1.10-1676463660979:blk_1073741836_1012 len=33554432 Live_repl=2
12. BP-445278276-10.10.1.10-1676463660979:blk_1073741837_1013 len=33554432 Live_repl=2
13. BP-445278276-10.10.1.10-1676463660979:blk_1073741838_1014 len=33554432 Live_repl=2
14. BP-445278276-10.10.1.10-1676463660979:blk_1073741839_1015 len=4748237 Live_repl=2
```

```
Status: HEALTHY
Number of data-nodes: 2
Number of racks: 1
Total dirs: 0
Total symlinks: 0
```

```
Replicated Blocks:
Total size: 474510285 B
Total files: 1
Total blocks (validated): 15 (avg. block size 31634019 B)
```

Relaciona el número de bloques que componen el fichero en HDFS con lo observado en la transparencia anterior (espacio ocupado en HDFS y espacio ocupado en los discos de los nodos *worker*)



¿Por qué el tamaño del último bloque es distinto al del resto?  
¿Cómo interpretas "Live\_repl=2"?



# Ejercicio 5: Ejecución de Hadoop WordCount

51

- Hadoop proporciona una interfaz web en el nodo que ejecuta el proceso *NameNode* (el nodo *master*)
  - Proporciona información acerca de HDFS y el estado de los *DataNodes*, acceso a los ficheros de *log*, permite navegar el sistema de ficheros, etc
- La interfaz web está disponible en la URL: <http://localhost:9870>
- Accede desde tu equipo y curiosea por las diferentes opciones del menú



## Overview 'rre-aisi2223-master:9000' (✓active)

Started:	Fri Feb 17 16:05:20 +0100 2023
Version:	3.3.4, ra585a73c3e02ac62350c136643a5e7f6095a3dbb
Compiled:	Fri Jul 29 14:32:00 +0200 2022 by stevel from branch-3.3.4
Cluster ID:	CID-4743b376-dc7a-4c23-9dc0-07575fe94d9e
Block Pool ID:	BP-1991826084-10.10.1.10-1676646308739

## Summary

Security is off.

Safemode is off.

1 files and directories, 0 blocks (0 replicated blocks, 0 erasure coded block groups) = 1 total filesystem object(s).

Heap Memory used 31.31 MB of 70.2 MB Heap Memory. Max Heap Memory is 494.94 MB.

Non Heap Memory used 49.03 MB of 52.59 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.



# Ejercicio 5: Ejecución de Hadoop WordCount

52

- Resumen en la pestaña Overview

## Summary

Security is off.

Safemode is off.

3 files and directories, 15 blocks (15 replicated blocks, 0 erasure coded block groups) = 18 total filesystem object(s).

Heap Memory used 47.52 MB of 70.2 MB Heap Memory. Max Heap Memory is 494.94 MB.

Non Heap Memory used 67.58 MB of 70.21 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	38.98 GB	← Capacidad disponible, espacio ocupado y espacio libre en HDFS
Configured Remote Capacity:	0 B	
DFS Used:	912.39 MB (2.29%)	←
Non DFS Used:	240 KB	
DFS Remaining:	36.03 GB (92.42%)	←
Block Pool Used:	912.39 MB (2.29%)	
DataNodes usages% (Min/Median/Max/stdDev):	2.29% / 2.29% / 2.29% / 0.00%	
Live Nodes	2 (Decommissioned: 0, In Maintenance: 0)	← DataNodes "vivos" (servicio en ejecución y funcionando)
Dead Nodes	0 (Decommissioned: 0, In Maintenance: 0)	← y "muertos" (el servicio no se está ejecutando o no responde con normalidad al NameNode)
Decommissioning Nodes	0	
Entering Maintenance Nodes	0	
Total Datanode Volume Failures	0 (0 B)	
Number of Under-Replicated Blocks	0	
Number of Blocks Pending Deletion (including replicas)	0	
Block Deletion Start Time	Fri Feb 17 16:05:20 +0100 2023	
Last Checkpoint Time	Fri Feb 17 16:05:08 +0100 2023	
Enabled Erasure Coding Policies	RS-6-3-1024k	

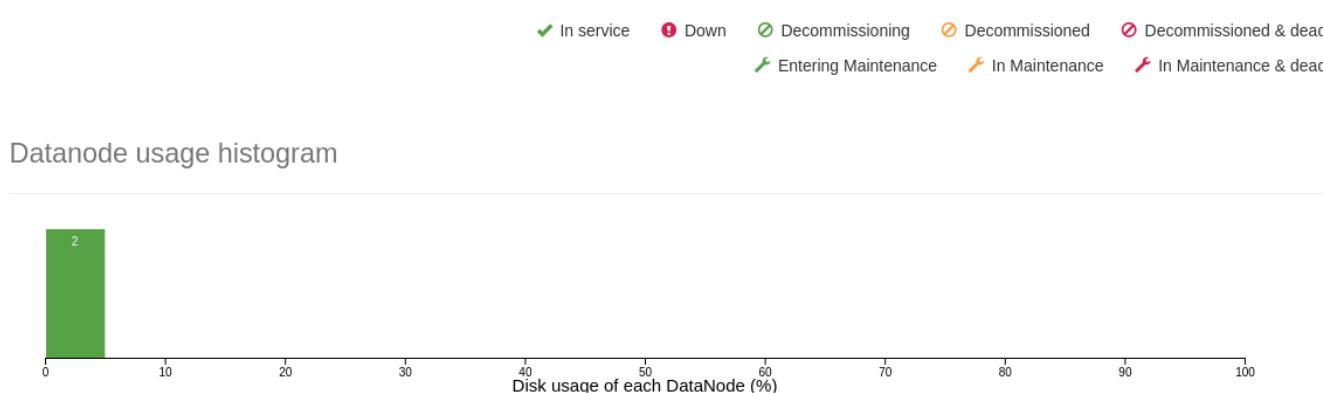


# Ejercicio 5: Ejecución de Hadoop WordCount

53

- Información en la pestaña *DataNodes*

## Datanode Information



## In operation

DataNode State	All	Show	25	entries	Search:				
Node	Http Address	Last contact	Last Block Report	Used	Non DFS Used	Capacity	Blocks	Block pool used	Version
✓/default-rack/rre-aisi2223-worker-1:9866 (10.10.1.11:9866)	<a href="http://rre-aisi2223-worker-1:9864">http://rre-aisi2223-worker-1:9864</a>	1s	45m	456.2 MB	120 KB	19.49 GB	15	456.2 MB (2.29%)	3.3.4
✓/default-rack/rre-aisi2223-worker-2:9866 (10.10.1.12:9866)	<a href="http://rre-aisi2223-worker-2:9864">http://rre-aisi2223-worker-2:9864</a>	0s	45m	456.2 MB	120 KB	19.49 GB	15	456.2 MB (2.29%)	3.3.4

Showing 1 to 2 of 2 entries

Previous **1** Next

Espacio ocupado en HDFS por *DataNode*:  
456 MiB de 19.49 GiB (~2%)



# Ejercicio 5: Ejecución de Hadoop WordCount

54

- Pestaña Utilities->Browse the file system
  - Entra en el directorio /hadoop-input

## Browse Directory

/hadoop-input

Show 25 entries

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hadoop	supergroup	452.53 MB	Feb 17 16:16	2	32 MB	file.xml

Showing 1 to 1 of 1 entries

Curiosea la información del fichero pinchando en su nombre

- Pestaña Utilities->Logs

## Directory: /logs/

Name ↑	Last Modified	Size
hadoop-hadoop-datanode-rre-aisi2223-worker-1.log	Feb 17, 2023, 3:16:47 PM	55,570 bytes
hadoop-hadoop-datanode-rre-aisi2223-worker-1.out	Feb 17, 2023, 3:05:22 PM	816 bytes
hadoop-hadoop-datanode-rre-aisi2223-worker-2.log	Feb 17, 2023, 3:17:12 PM	55,442 bytes
hadoop-hadoop-datanode-rre-aisi2223-worker-2.out	Feb 17, 2023, 3:05:22 PM	816 bytes
hadoop-hadoop-namenode-rre-aisi2223-master.log	Feb 17, 2023, 3:16:43 PM	54,706 bytes
hadoop-hadoop-namenode-rre-aisi2223-master.out	Feb 17, 2023, 3:15:58 PM	7,094 bytes
hadoop-hadoop-nodemanager-rre-aisi2223-worker-1.log	Feb 17, 2023, 3:45:51 PM	46,198 bytes
hadoop-hadoop-nodemanager-rre-aisi2223-worker-1.out	Feb 17, 2023, 3:05:48 PM	2,952 bytes
hadoop-hadoop-nodemanager-rre-aisi2223-worker-2.log	Feb 17, 2023, 3:46:16 PM	46,198 bytes
hadoop-hadoop-nodemanager-rre-aisi2223-worker-2.out	Feb 17, 2023, 3:06:18 PM	2,952 bytes
hadoop-hadoop-resourcemanager-rre-aisi2223-master.log	Feb 17, 2023, 3:15:49 PM	50,924 bytes
hadoop-hadoop-resourcemanager-rre-aisi2223-master.out	Feb 17, 2023, 3:05:49 PM	2,968 bytes
hadoop-hadoop-secondarynamenode-rre-aisi2223-master.log	Feb 17, 2023, 3:05:29 PM	36,977 bytes
hadoop-hadoop-secondarynamenode-rre-aisi2223-master.out	Feb 17, 2023, 3:05:27 PM	816 bytes
userlogs/	Feb 17, 2023, 3:05:45 PM	4,096 bytes

Curiosea el contenido de cualquier fichero de log pinchando en el enlace



Los ficheros de log también están accesibles en la ruta /share/hadoop-3.3.4/logs



# Ejercicio 5: Ejecución de Hadoop WordCount

55

- Para simplificar la ejecución de la aplicación WordCount incluida en Hadoop, vamos a definir la variable JAR con la ruta al fichero jar que la contiene
  - export JAR=\$HADOOP\_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.4.jar

```
hadoop@rre-aisi2223-master:~$ export JAR=$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.4.jar
hadoop@rre-aisi2223-master:~$ ls -lh $JAR
-rwxr-xr-x 1 hadoop hadoop 275K Jul 29 2022 /share/nfs/hadoop/hadoop-3.3.4/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.4.jar
hadoop@rre-aisi2223-master:~$
```

- Para ejecutar aplicaciones Hadoop con YARN, se utiliza el comando **yarn**:
  - Sintaxis general: **yarn jar JAR\_FILE [MAIN\_CLASS] [ARGS]**
  - Obtén el listado de aplicaciones incluidas en el fichero jar: **yarn jar \$JAR**

```
hadoop@rre-aisi2223-master:~$ yarn jar $JAR
An example program must be given as the first argument.
Valid program names are:
  aggregatewordcount: An Aggregate based map/reduce program that counts the words in the input files.
  aggregatewordhist: An Aggregate based map/reduce program that computes the histogram of the words in the input files.
  bbp: A map/reduce program that uses Bailey-Borwein-Plouffe to compute exact digits of Pi.
  dbcount: An example job that count the pageview counts from a database.
  distbbp: A map/reduce program that uses a BBP-type formula to compute exact bits of Pi.
  grep: A map/reduce program that counts the matches of a regex in the input.
  join: A job that effects a join over sorted, equally partitioned datasets
  multifilewc: A job that counts words from several files.
  pentomino: A map/reduce tile laying program to find solutions to pentomino problems.
  pi: A map/reduce program that estimates Pi using a quasi-Monte Carlo method.
  randomtextwriter: A map/reduce program that writes 10GB of random textual data per node.
  randomwriter: A map/reduce program that writes 10GB of random data per node.
  secondarysort: An example defining a secondary sort to the reduce.
  sort: A map/reduce program that sorts the data written by the random writer.
  sudoku: A sudoku solver.
  teragen: Generate data for the terasort
  terasort: Run the terasort
  teravalidate: Checking results of terasort
  wordcount: A map/reduce program that counts the words in the input files. wordcount: A map/reduce program that counts the words in the input files.
  wordmean: A map/reduce program that counts the average length of the words in the input files.
  wordmedian: A map/reduce program that counts the median length of the words in the input files.
  wordstandarddeviation: A map/reduce program that counts the standard deviation of the length of the words in the input files.
hadoop@rre-aisi2223-master:~$
```



# Ejercicio 5: Ejecución de Hadoop WordCount

56

- La aplicación *WordCount* requiere uno o más directorios de entrada en HDFS con los ficheros a procesar y un directorio de salida donde escribir los resultados

```
hadoop@rre-aisi2223-master:~$ yarn jar $JAR wordcount
Usage: wordcount <in> [<in>...] <out>
hadoop@rre-aisi2223-master:~$
```

- En nuestro ejemplo vamos a procesar un único fichero de texto que tenemos almacenado en el directorio de HDFS: */hadoop-input*
- Como directorio de salida en HDFS usaremos: */hadoop-output*
- Ejecutaremos *WordCount* usando un **número diferente de tareas Reduce**
  - Este parámetro tiene un impacto directo en el rendimiento de la fase de reducción de una aplicación Hadoop MapReduce
    - Su impacto depende de múltiples factores: la propia lógica de la aplicación, la configuración de Hadoop, los recursos computacionales del clúster...
  - Configurable mediante la definición de una *property* de Java:
    - *yarn jar \$JAR wordcount -Dmapreduce.job.reduces=X [in] [out]*



# Ejercicio 5: Ejecución de Hadoop WordCount

57

- Ejecuta la aplicación WordCount usando 1 y 4 tareas Reduce:
  - `time yarn jar $JAR wordcount -Dmapreduce.job.reduces=1 /hadoop-input /hadoop-output1`
  - `time yarn jar $JAR wordcount -Dmapreduce.job.reduces=4 /hadoop-input /hadoop-output4`



Anteponemos el comando `time`  
para medir el tiempo de ejecución

```
hadoop@rre-aisi2223-master:~$ time yarn jar $JAR wordcount -Dmapreduce.job.reduces=1 /hadoop-input /hadoop-output1
2023-02-16 11:05:53,715 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at rre-aisi2:
2023-02-16 11:05:54,285 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/sta...
2023-02-16 11:05:55,307 INFO input.FileInputFormat: Total input files to process : 1
2023-02-16 11:05:55,436 INFO mapreduce.JobSubmitter: number of splits:15
2023-02-16 11:05:55,920 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1676545478598_0001
2023-02-16 11:05:55,920 INFO mapreduce.JobSubmitter: Executing with tokens: []
2023-02-16 11:05:56,177 INFO conf.Configuration: resource-types.xml not found
2023-02-16 11:05:56,178 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2023-02-16 11:05:56,184 WARN mapred.YARNRunner: Configuration yarn.app.mapreduce.am.resource.memory-mb=256Mi is ove...
ration
2023-02-16 11:05:56,529 INFO impl.YarnClientImpl: Submitted application application_1676545478598_0001
2023-02-16 11:05:56,625 INFO mapreduce.Job: The url to track the job: http://rre-aisi2223-master:8088/proxy/applic...
2023-02-16 11:05:56,625 INFO mapreduce.Job: Running job: job_1676545478598_0001
2023-02-16 11:06:02,863 INFO mapreduce.Job: Job job_1676545478598_0001 running in uber mode : false
2023-02-16 11:06:02,863 INFO mapreduce.Job: map 0% reduce 0%
2023-02-16 11:06:17,995 INFO mapreduce.Job: map 13% reduce 0%
2023-02-16 11:06:25,037 INFO mapreduce.Job: map 33% reduce 0%
2023-02-16 11:06:31,072 INFO mapreduce.Job: map 40% reduce 0%
2023-02-16 11:06:32,080 INFO mapreduce.Job: map 47% reduce 0%
2023-02-16 11:06:42,135 INFO mapreduce.Job: map 53% reduce 0%
2023-02-16 11:06:43,139 INFO mapreduce.Job: map 62% reduce 0%
2023-02-16 11:06:44,143 INFO mapreduce.Job: map 64% reduce 0%
2023-02-16 11:06:45,147 INFO mapreduce.Job: map 73% reduce 0%
2023-02-16 11:06:47,159 INFO mapreduce.Job: map 73% reduce 24%
```

Extracto de la salida obtenida al ejecutar WordCount con una única tarea Reduce.  
Entre otras cosas, en esta parte de la salida vemos el progreso de la ejecución



# Ejercicio 5: Ejecución de Hadoop WordCount

58

```
2023-02-16 11:07:06,258 INFO mapreduce.Job: map 100% reduce 73%
2023-02-16 11:07:09,274 INFO mapreduce.Job: map 100% reduce 100%
2023-02-16 11:07:10,285 INFO mapreduce.Job: Job job_1676545478598_0001 completed successfully
2023-02-16 11:07:10,395 INFO mapreduce.Job: counters: 54
File System Counters
    FILE: Number of bytes read=443443966
    FILE: Number of bytes written=670864623
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=474569399
    HDFS: Number of bytes written=151944590
    HDFS: Number of read operations=50
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=2
    HDFS: Number of bytes read erasure-coded=0
Job Counters
    Launched map tasks=15
    Launched reduce tasks=1
    Data-local map tasks=15
    Total time spent by all maps in occupied slots (ms)=871724
    Total time spent by all reduces in occupied slots (ms)=147972
    Total time spent by all map tasks (ms)=217931
    Total time spent by all reduce tasks (ms)=36993
    Total vcore-milliseconds taken by all map tasks=217931
    Total vcore-milliseconds taken by all reduce tasks=36993
    Total megabyte-milliseconds taken by all map tasks=111580672
    Total megabyte-milliseconds taken by all reduce tasks=18940416
Map-Reduce Framework
    Map input records=7363638
    Map output records=44363323
    Map output bytes=632697182
    Map output materialized bytes=222998003
```

Tras finalizar el trabajo correctamente, Hadoop nos ofrece mucha información interesante acerca de la ejecución, como el número de tareas Map y Reduce que fueron ejecutadas. Cada tarea se ejecutó en un Container de YARN



¿Por qué se han ejecutado 15 tareas Map? ¿Cuántas tareas Map se ejecutan cuando usamos 4 tareas Reduce?



# Ejercicio 5: Ejecución de Hadoop WordCount

59



- Compara los tiempos de ejecución en función del número de Reducers
  - ¿Observas alguna diferencia?
    - En nuestro caso es probable que no, ya que estamos virtualizando un clúster pero en realidad todo se ejecuta en una única CPU y disco físicos
    - En un clúster real, el número de Reducers podría tener un impacto significativo
- Despues de las ejecuciones, lista de forma recursiva el contenido de los dos directorios de salida mediante la operación de HDFS: ls

```
hadoop@rre-aisi2223-master:~$ hdfs dfs -ls -R -h /hadoop-output1
-rw-r--r-- 2 hadoop supergroup          0 2023-02-17 11:37 /hadoop-output1/_SUCCESS ←
-rw-r--r-- 2 hadoop supergroup    144.9 M 2023-02-17 11:37 /hadoop-output1/part-r-00000
hadoop@rre-aisi2223-master:~$ 
hadoop@rre-aisi2223-master:~$ hdfs dfs -ls -R -h /hadoop-output4
-rw-r--r-- 2 hadoop supergroup          0 2023-02-17 11:42 /hadoop-output4/_SUCCESS
-rw-r--r-- 2 hadoop supergroup    36.0 M 2023-02-17 11:42 /hadoop-output4/part-r-00000
-rw-r--r-- 2 hadoop supergroup    36.0 M 2023-02-17 11:42 /hadoop-output4/part-r-00001
-rw-r--r-- 2 hadoop supergroup    36.3 M 2023-02-17 11:42 /hadoop-output4/part-r-00002
-rw-r--r-- 2 hadoop supergroup    36.6 M 2023-02-17 11:42 /hadoop-output4/part-r-00003
hadoop@rre-aisi2223-master:~$
```

Los ficheros vacíos "SUCCESS" simplemente indican que la aplicación ha terminado con éxito (no forman parte de la salida producida por la aplicación)



Fíjate en el número y en el tamaño de los ficheros de salida en función del número de Reducers. ¿Qué observas?



# Ejercicio 5: Ejecución de Hadoop WordCount

60

- Obtén más información del fichero de salida de la ejecución con **una única tarea Reduce** y analiza lo que observas:
  - `hdfs fsck /hadoop-output1/part-r-00000 -files -blocks`
- Copia ahora ese fichero de salida desde su ruta en HDFS a la ruta **/tmp** del sistema de ficheros local del nodo *master*
  - Consulta la ayuda de la operación de HDFS: [get](#)
  - Cópialo en un fichero con nombre **hadoopOutputR1**
- A continuación, ejecuta el siguiente comando en el nodo *master* para ver un extracto del fichero de salida y su formato:
  - `head -3 /tmp/hadoopOutputR1`

```
hadoop@rre-aisi2223-master:~$ hdfs dfs -get /hadoop-output1/part-r-00000 /tmp/hadoopOutputR1
hadoop@rre-aisi2223-master:~$ 
hadoop@rre-aisi2223-master:~$ ls -lh /tmp/hadoopOutputR1
-rw-r--r-- 1 hadoop hadoop 145M Feb 17 11:47 /tmp/hadoopOutputR1
hadoop@rre-aisi2223-master:~$ 
hadoop@rre-aisi2223-master:~$ head -3 /tmp/hadoopOutputR1
!
 60975
!!
 11168
!!!
 13
hadoop@rre-aisi2223-master:~$
```



*WordCount* nos proporciona como salida el número de ocurrencias para cada palabra presente en el fichero de entrada. Recuerda que el fichero de entrada en nuestro caso tiene formato XML



# Ejercicio 5: Ejecución de Hadoop WordCount

61



- "Copia y combina" todos los ficheros de salida de la ejecución con 4 **Reducers** desde HDFS en la ruta **/tmp** del nodo master
  - Consulta la ayuda de la operación de HDFS: [getmerge](#)
  - Cópialo en un fichero con nombre **hadoopOutputR4**
- Determina el número de ocurrencias que tiene en el fichero de entrada la etiqueta XML "**<page>**" en ambos ficheros de salida
  - `cat /tmp/hadoopOutputR1 | grep "<page>"`
  - `cat /tmp/hadoopOutputR4 | grep "<page>"`

```
hadoop@rre-aisi2223-master:~$ hdfs dfs -getmerge /hadoop-output4 /tmp/hadoopOutputR4
hadoop@rre-aisi2223-master:~$ 
hadoop@rre-aisi2223-master:~$ ls -lh /tmp/hadoopOutputR*
-rw-r--r-- 1 hadoop hadoop 145M Feb 17 11:47 /tmp/hadoopOutputR1
-rw-r--r-- 1 hadoop hadoop 145M Feb 17 11:51 /tmp/hadoopOutputR4
hadoop@rre-aisi2223-master:~$ 
hadoop@rre-aisi2223-master:~$ cat /tmp/hadoopOutputR1 | grep "<page>" 
<page> 134342
hadoop@rre-aisi2223-master:~$ 
hadoop@rre-aisi2223-master:~$ cat /tmp/hadoopOutputR4 | grep "<page>" 
<page> 134342
hadoop@rre-aisi2223-master:~$
```



¿Crees que el tamaño de los ficheros de salida debería ser siempre el mismo independientemente del número de Reducers?



# Ejercicio 6: Instalación de Spark

62

- La instalación de Apache Spark se realizará de forma análoga a la instalación de Hadoop:
  - Instalaremos Spark únicamente en el directorio del nodo *master* compartido por NFS (*/share/nfs*)
  - Se usará Ansible para automatizar su instalación/configuración
  - Se proporciona el playbook completo y listo para ejecutar en el controlador
- Ejecuta el playbook de instalación de Spark como **usuario vagrant**:
  - *ansible-playbook /vagrant/ansible/hadoop/playbooks/spark.yml*
- Antes de continuar, ejecuta los siguientes comandos en el nodo *master* para comprobar la instalación de Spark:
  - *ansible xxx-aisi2223-worker-1 -m shell -a "ls /share/nfs/spark/spark-\*"*
  - *su -l hadoop*
  - *echo \$PATH && echo \$SPARK\_HOME*
  - *spark-submit --version*



# Ejercicio 6: Instalación de Spark



63

```
vagrant@rre-aisi2223-master:~$ ansible rre-aisi2223-worker-1 -m shell -a "ls /share/nfs/spark/spark-*"
rre-aisi2223-worker-1 | CHANGED | rc=0 >>
LICENSE
NOTICE
R
README.md
RELEASE
bin
conf
data
examples
jars
kubernetes
licenses
python
sbin
yarn
vagrant@rre-aisi2223-master:~$ su -l hadoop
Password:
hadoop@rre-aisi2223-master:~$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/share/nfs/hadoop/hadoop-3.3.4/bin:/share/nfs,
/share/nfs/spark/spark-3.3.2-bin-hadoop3/sbin
hadoop@rre-aisi2223-master:~$ spark-submit --version
Welcome to

$$\begin{array}{c} \diagup \diagdown \diagup \diagdown \diagup \diagdown \\ \diagdown \diagup \diagdown \diagup \diagdown \diagup \\ \diagup \diagdown \diagup \diagdown \diagup \diagdown \\ \diagdown \diagup \diagdown \diagup \diagdown \diagup \\ \diagup \diagdown \diagup \diagdown \diagup \diagdown \\ \diagdown \diagup \diagdown \diagup \diagdown \diagup \\ \diagup \diagdown \diagup \diagdown \diagup \diagdown \end{array}$$
 version 3.3.2
```

Estamos logueados como usuario vagrant.  
Luego nos cambiamos a hadoop

PATH correctamente configurado

```
Using Scala version 2.12.15, OpenJDK 64-Bit Server VM, 11.0.18
Branch HEAD
Compiled by user liangchi on 2023-02-10T19:57:40Z
Revision 5103e00c4ce5fcc4264ca9c4df12295d42557af6
Url https://github.com/apache/spark
Type --help for more information.
hadoop@rre-aisi2223-master:~$
```



# Ejercicio 7: Ejecución de Spark WordCount

64

- En este ejercicio ejecutaremos *WordCount* implementado con Spark
  - Procesaremos como entrada el mismo fichero (*file.xml*) que con Hadoop MapReduce, el cual ya tenemos en HDFS (*/hadoop-input*)
- Para ejecutar una aplicación Spark se utiliza el comando *spark-submit*:
  - ***spark-submit --class [MAIN\_CLASS]--master [URL] JAR\_FILE [ARGS]***
- En este caso Spark no integra un fichero *jar* con la aplicación *WordCount*
  - Descarga en el nodo *master* el fichero *jar* que necesitamos
    - wget <https://gac.udc.es/~rober/aisi/sparkbench.jar>
  - Por tanto, el fichero *jar* a ejecutar sería: ***sparkbench.jar***
  - La clase (*--class*) para *WordCount* es: ***es.udc.gac.sparkbench.ScalaWordCount***
  - El valor del *parámetro --master* depende del *Cluster Manager*:
    - Recuerda que Spark proporciona su propio *Cluster Manager* (Spark Standalone), además de soportar YARN entre otros (ver transparencias 20-21)
    - En nuestro caso vamos a usar YARN, el cual ya lo tenemos desplegado en el clúster, por lo que debemos especificar: ***--master yarn***



# Ejercicio 7: Ejecución de Spark WordCount

65

- Los parámetros de la aplicación son los mismos que con Hadoop MapReduce:
  - Directorio de entrada en HDFS (`/hadoop-input`) con los ficheros a procesar
  - Directorio de salida en HDFS donde Spark escribirá los resultados (`/spark-output`)
- Esta aplicación para Spark soporta varios formatos del fichero de entrada
  - Como nuestro fichero de entrada es de tipo texto, deberemos indicarlo en el último parámetro de la aplicación mediante la opción **Text**
- Por último, el parámetro equivalente de Spark para especificar el número de tareas Reduce de Hadoop es: **--conf spark.default.parallelism=X**
- El comando final de ejecución, incluyendo la medición de tiempo, quedaría:
  - `time spark-submit --class es.udc.gac.sparkbench.ScalaWordCount --master yarn --conf spark.default.parallelism=X sparkbench.jar /hadoop-input /spark-outputX Text`
- Ejecuta la aplicación WordCount usando **1 y 4 tareas Reduce**:
  - `time spark-submit --class es.udc.gac.sparkbench.ScalaWordCount --master yarn --conf spark.default.parallelism=1 sparkbench.jar /hadoop-input /spark-output1 Text`
  - `time spark-submit --class es.udc.gac.sparkbench.ScalaWordCount --master yarn --conf spark.default.parallelism=4 sparkbench.jar /hadoop-input /spark-output4 Text`



# Ejercicio 7: Ejecución de Spark WordCount

66

```
23/02/17 12:18:20 INFO YarnScheduler: Adding task set 0.0 with 15 tasks resource profile 0
23/02/17 12:18:21 INFO TaskSetManager: Starting task 0.0 in stage 0.0 (TID 0) (rre-aisi2223-worker-2, executor 2, partition 0, NODE_LOCAL, 4564 bytes)
23/02/17 12:18:21 INFO TaskSetManager: Starting task 1.0 in stage 0.0 (TID 1) (rre-aisi2223-worker-1, executor 1, partition 1, NODE_LOCAL, 4564 bytes)
23/02/17 12:18:21 INFO TaskSetManager: Starting task 2.0 in stage 0.0 (TID 2) (rre-aisi2223-worker-2, executor 2, partition 2, NODE_LOCAL, 4564 bytes)
23/02/17 12:18:21 INFO TaskSetManager: Starting task 3.0 in stage 0.0 (TID 3) (rre-aisi2223-worker-1, executor 1, partition 3, NODE_LOCAL, 4564 bytes)
23/02/17 12:18:21 INFO BlockManagerInfo: Added broadcast_1_piece0 in memory on rre-aisi2223-worker-2:35699 (size: 3.8 KiB, free: 148.4 MiB)
23/02/17 12:18:21 INFO BlockManagerInfo: Added broadcast_1_piece0 in memory on rre-aisi2223-worker-1:43049 (size: 3.8 KiB, free: 148.4 MiB)
23/02/17 12:18:21 INFO BlockManagerInfo: Added broadcast_0_piece0 in memory on rre-aisi2223-worker-2:35699 (size: 33.3 KiB, free: 148.4 MiB)
23/02/17 12:18:21 INFO BlockManagerInfo: Added broadcast_0_piece0 in memory on rre-aisi2223-worker-1:43049 (size: 33.3 KiB, free: 148.4 MiB)
23/02/17 12:18:29 INFO TaskSetManager: Starting task 4.0 in stage 0.0 (TID 4) (rre-aisi2223-worker-1, executor 1, partition 4, NODE_LOCAL, 4564 bytes)
23/02/17 12:18:29 INFO TaskSetManager: Finished task 1.0 in stage 0.0 (TID 1) in 7989 ms on rre-aisi2223-worker-1 (executor 1) (1/15)
23/02/17 12:18:29 INFO TaskSetManager: Starting task 5.0 in stage 0.0 (TID 5) (rre-aisi2223-worker-1, executor 1, partition 5, NODE_LOCAL, 4564 bytes)
23/02/17 12:18:29 INFO TaskSetManager: Finished task 3.0 in stage 0.0 (TID 3) in 8030 ms on rre-aisi2223-worker-1 (executor 1) (2/15)
23/02/17 12:18:29 INFO TaskSetManager: Starting task 6.0 in stage 0.0 (TID 6) (rre-aisi2223-worker-2, executor 2, partition 6, NODE_LOCAL, 4564 bytes)
23/02/17 12:18:29 INFO TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 8706 ms on rre-aisi2223-worker-2 (executor 2) (3/15)
23/02/17 12:18:29 INFO TaskSetManager: Starting task 7.0 in stage 0.0 (TID 7) (rre-aisi2223-worker-2, executor 2, partition 7, NODE_LOCAL, 4564 bytes)
23/02/17 12:18:29 INFO TaskSetManager: Finished task 2.0 in stage 0.0 (TID 2) in 8748 ms on rre-aisi2223-worker-2 (executor 2) (4/15)
23/02/17 12:18:32 INFO TaskSetManager: Starting task 8.0 in stage 0.0 (TID 8) (rre-aisi2223-worker-1, executor 1, partition 8, NODE_LOCAL, 4564 bytes)
23/02/17 12:18:32 INFO TaskSetManager: Finished task 5.0 in stage 0.0 (TID 5) in 3738 ms on rre-aisi2223-worker-1 (executor 1) (5/15)
23/02/17 12:18:33 INFO TaskSetManager: Starting task 9.0 in stage 0.0 (TID 9) (rre-aisi2223-worker-1, executor 1, partition 9, NODE_LOCAL, 4564 bytes)
23/02/17 12:18:33 INFO TaskSetManager: Finished task 4.0 in stage 0.0 (TID 4) in 4127 ms on rre-aisi2223-worker-1 (executor 1) (6/15)
23/02/17 12:18:34 INFO TaskSetManager: Starting task 10.0 in stage 0.0 (TID 10) (rre-aisi2223-worker-2, executor 2, partition 10, NODE_LOCAL, 4564 bytes)
23/02/17 12:18:34 INFO TaskSetManager: Finished task 6.0 in stage 0.0 (TID 6) in 4451 ms on rre-aisi2223-worker-2 (executor 2) (7/15)
23/02/17 12:18:34 INFO TaskSetManager: Starting task 11.0 in stage 0.0 (TID 11) (rre-aisi2223-worker-2, executor 2, partition 11, NODE_LOCAL, 4564 bytes)
23/02/17 12:18:34 INFO TaskSetManager: Finished task 7.0 in stage 0.0 (TID 7) in 4875 ms on rre-aisi2223-worker-2 (executor 2) (8/15)
23/02/17 12:18:36 INFO TaskSetManager: Starting task 12.0 in stage 0.0 (TID 12) (rre-aisi2223-worker-1, executor 1, partition 12, NODE_LOCAL, 4564 bytes)
23/02/17 12:18:36 INFO TaskSetManager: Finished task 8.0 in stage 0.0 (TID 8) in 3585 ms on rre-aisi2223-worker-1 (executor 1) (9/15)
23/02/17 12:18:37 INFO TaskSetManager: Starting task 13.0 in stage 0.0 (TID 13) (rre-aisi2223-worker-1, executor 1, partition 13, NODE_LOCAL, 4564 bytes)
23/02/17 12:18:37 INFO TaskSetManager: Finished task 9.0 in stage 0.0 (TID 9) in 4708 ms on rre-aisi2223-worker-1 (executor 1) (10/15)
23/02/17 12:18:38 INFO TaskSetManager: Starting task 14.0 in stage 0.0 (TID 14) (rre-aisi2223-worker-2, executor 2, partition 14, NODE_LOCAL, 4564 bytes)
23/02/17 12:18:38 INFO TaskSetManager: Finished task 10.0 in stage 0.0 (TID 10) in 4374 ms on rre-aisi2223-worker-2 (executor 2) (11/15)
23/02/17 12:18:39 INFO TaskSetManager: Finished task 11.0 in stage 0.0 (TID 11) in 4446 ms on rre-aisi2223-worker-2 (executor 2) (12/15)
23/02/17 12:18:39 INFO TaskSetManager: Finished task 14.0 in stage 0.0 (TID 14) in 772 ms on rre-aisi2223-worker-2 (executor 2) (13/15)
23/02/17 12:18:40 INFO TaskSetManager: Finished task 12.0 in stage 0.0 (TID 12) in 3898 ms on rre-aisi2223-worker-1 (executor 1) (14/15)
23/02/17 12:18:45 INFO TaskSetManager: Finished task 13.0 in stage 0.0 (TID 13) in 7179 ms on rre-aisi2223-worker-1 (executor 1) (15/15)
23/02/17 12:18:45 INFO YarnScheduler: Removed TaskSet 0.0, whose tasks have all completed, from pool
23/02/17 12:18:45 INFO DAGScheduler: ShuffleMapStage 0 (map at ScalaWordCount.scala:30) finished in 24.233 s
23/02/17 12:18:45 INFO DAGScheduler: looking for newly runnable stages
```



La salida de la aplicación Spark es **muy extensa**. Esta captura muestra un extracto de la ejecución con **4 Reducers**, en concreto la parte de la ejecución correspondiente a la **fase Map**. Si todo ha ido bien, deberías ver algo similar, con mensajes sobre los **Executors** de Spark ejecutando tareas de procesamiento



Tenemos **dos Executors** ejecutando tareas. ¿por qué dos? Qué entre ambos ejecuten en total **15 tareas**, ¿es casualidad? ¿podría tener que ver con algo visto previamente? Repasa la transparencia 50



# Ejercicio 7: Ejecución de Spark WordCount

67



Esta captura continua la salida de la ejecución previa, mostrando la parte correspondiente a la **fase Reduce** y el final de la ejecución de la aplicación

```
23/02/17 12:18:45 INFO YarnScheduler: Adding task set 1_0 with 4 tasks resource profile _0
23/02/17 12:18:45 INFO TaskSetManager: Starting task 0.0 in stage 1.0 (TID 15) (rre-aisi2223-worker-1, executor 1, partition 0, NODE_LOCAL, 4282 bytes)
23/02/17 12:18:45 INFO TaskSetManager: Starting task 1.0 in stage 1.0 (TID 16) (rre-aisi2223-worker-2, executor 2, partition 1, NODE_LOCAL, 4282 bytes)
23/02/17 12:18:45 INFO TaskSetManager: Starting task 2.0 in stage 1.0 (TID 17) (rre-aisi2223-worker-1, executor 1, partition 2, NODE_LOCAL, 4282 bytes)
23/02/17 12:18:45 INFO TaskSetManager: Starting task 3.0 in stage 1.0 (TID 18) (rre-aisi2223-worker-2, executor 2, partition 3, NODE_LOCAL, 4282 bytes)
23/02/17 12:18:45 INFO BlockManagerInfo: Added broadcast_2_piece0 in memory on rre-aisi2223-worker-1:43049 (size: 38.2 KiB, free: 148.3 MiB)
23/02/17 12:18:45 INFO BlockManagerInfo: Added broadcast_2_piece0 in memory on rre-aisi2223-worker-2:35699 (size: 38.2 KiB, free: 148.3 MiB)
23/02/17 12:18:45 INFO MapOutputTrackerMasterEndpoint: Asked to send map output locations for shuffle 0 to 10.10.1.11:60768
23/02/17 12:18:45 INFO MapOutputTrackerMasterEndpoint: Asked to send map output locations for shuffle 0 to 10.10.1.12:35790
23/02/17 12:18:50 INFO TaskSetManager: Finished task 3.0 in stage 1.0 (TID 18) in 4920 ms on rre-aisi2223-worker-2 (executor 2) (1/4)
23/02/17 12:18:50 INFO TaskSetManager: Finished task 1.0 in stage 1.0 (TID 16) in 5109 ms on rre-aisi2223-worker-2 (executor 2) (2/4)
23/02/17 12:18:55 INFO TaskSetManager: Finished task 2.0 in stage 1.0 (TID 17) in 10070 ms on rre-aisi2223-worker-1 (executor 1) (3/4)
23/02/17 12:18:55 INFO TaskSetManager: Finished task 0.0 in stage 1.0 (TID 15) in 10226 ms on rre-aisi2223-worker-1 (executor 1) (4/4)
23/02/17 12:18:55 INFO YarnScheduler: Removed TaskSet 1.0, whose tasks have all completed, from pool
23/02/17 12:18:55 INFO DAGScheduler: ResultStage 1 (runJob at SparkHadoopWriter.scala:83) finished in 10.268 s
23/02/17 12:18:55 INFO DAGScheduler: Job 0 is finished. Cancelling potential speculative or zombie tasks for this job
23/02/17 12:18:55 INFO YarnScheduler: Killing all running tasks in stage 1. Stage finished
23/02/17 12:18:55 INFO DAGScheduler: Job 0 finished: runJob at SparkHadoopWriter.scala:83, took 35.123795 s
23/02/17 12:18:55 INFO SparkHadoopWriter: Start to commit write Job job_202302171218204517397322599718821_0005
23/02/17 12:18:55 INFO BlockManagerInfo: Removed broadcast_2_piece0 on rre-aisi2223-worker-1:43049 in memory (size: 38.2 KiB, free: 148.4 MiB)
23/02/17 12:18:55 INFO SparkHadoopWriter: Write Job job_202302171218204517397322599718821_0005 committed. Elapsed time: 142 ms.
23/02/17 12:18:55 INFO BlockManagerInfo: Removed broadcast_2_piece0 on rre-aisi2223-master:42705 in memory (size: 38.2 KiB, free: 482.9 MiB)
23/02/17 12:18:55 INFO BlockManagerInfo: Removed broadcast_2_piece0 on rre-aisi2223-worker-2:35699 in memory (size: 38.2 KiB, free: 148.4 MiB)
23/02/17 12:18:55 INFO SparkContext: Invoking stop() from shutdown hook
23/02/17 12:18:55 INFO BlockManagerInfo: Removed broadcast_1_piece0 on rre-aisi2223-worker-1:43049 in memory (size: 3.8 KiB, free: 148.4 MiB)
23/02/17 12:18:55 INFO BlockManagerInfo: Removed broadcast_1_piece0 on rre-aisi2223-worker-2:35699 in memory (size: 3.8 KiB, free: 148.4 MiB)
23/02/17 12:18:55 INFO BlockManagerInfo: Removed broadcast_1_piece0 on rre-aisi2223-master:42705 in memory (size: 3.8 KiB, free: 482.9 MiB)
23/02/17 12:18:55 INFO SparkUI: Stopped Spark web UI at http://rre-aisi2223-master:4040
23/02/17 12:18:55 INFO YarnClientSchedulerBackend: Interrupting monitor thread
23/02/17 12:18:55 INFO YarnClientSchedulerBackend: Shutting down all executors
23/02/17 12:18:55 INFO YarnSchedulerBackend$YarnDriverEndpoint: Asking each executor to shut down
23/02/17 12:18:55 INFO YarnClientSchedulerBackend: YARN client scheduler backend Stopped
23/02/17 12:18:55 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
23/02/17 12:18:55 INFO MemoryStore: MemoryStore cleared
23/02/17 12:18:55 INFO BlockManager: BlockManager stopped
23/02/17 12:18:55 INFO BlockManagerMaster: BlockManagerMaster stopped
23/02/17 12:18:55 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: outputCommitCoordinator stopped!
23/02/17 12:18:55 INFO SparkContext: Successfully stopped SparkContext
23/02/17 12:18:55 INFO ShutdownHookManager: Shutdown hook called
23/02/17 12:18:55 INFO ShutdownHookManager: Deleting directory /tmp/spark-27f06903-dd17-45ed-9a60-8617777b8ceb
23/02/17 12:18:55 INFO ShutdownHookManager: Deleting directory /tmp/spark-c229f9b7-4a02-494f-be22-c91285baac46
```

```
real    1m0.127s
user    0m12.558s
sys     0m2.539s
```

Los Executors ejecutan 4 tareas Reduce en total

Tras la fase Reduce, el trabajo termina y se detiene el *SparkContext* para finalizar la aplicación. El tiempo de total de la ejecución se refleja en el valor *real* del comando *time*



# Ejercicio 7: Ejecución de Spark WordCount

68



- Compara los tiempos de ejecución en función del número de Reducers
  - ¿Observas alguna diferencia?
- Compara el mejor tiempo de ejecución con Spark con el mejor tiempo de ejecución con Hadoop MapReduce
  - ¿Observas alguna diferencia significativa?
  - ¿Cuánto más rápido ejecuta Spark la aplicación WordCount?
- Después de las ejecuciones, lista de **forma recursiva** el contenido de los **dos directorios de salida** en HDFS de Spark

```
hadoop@rre-aisi2223-master:~$ hdfs dfs -ls -R -h /spark-output1
-rw-r--r--  2 hadoop supergroup      0 2023-02-17 12:45 /spark-output1/_SUCCESS
-rw-r--r--  2 hadoop supergroup  144.9 M 2023-02-17 12:45 /spark-output1/part-r-00000
hadoop@rre-aisi2223-master:~$ 
hadoop@rre-aisi2223-master:~$ hdfs dfs -ls -R -h /spark-output4
-rw-r--r--  2 hadoop supergroup      0 2023-02-17 12:18 /spark-output4/_SUCCESS
-rw-r--r--  2 hadoop supergroup  36.4 M 2023-02-17 12:18 /spark-output4/part-r-00000
-rw-r--r--  2 hadoop supergroup  36.0 M 2023-02-17 12:18 /spark-output4/part-r-00001
-rw-r--r--  2 hadoop supergroup  36.2 M 2023-02-17 12:18 /spark-output4/part-r-00002
-rw-r--r--  2 hadoop supergroup  36.4 M 2023-02-17 12:18 /spark-output4/part-r-00003
hadoop@rre-aisi2223-master:~$
```



Fíjate en el número y en el tamaño de los ficheros de salida que genera Spark. ¿Tiene alguna relación con lo que hemos visto previamente?



# Ejercicio 7: Ejecución de Spark WordCount

69



- "Copia y combina" todos los ficheros de salida de la ejecución con **4 Reducers** desde HDFS en la ruta **/tmp** del nodo master
  - Cójalo en un fichero con nombre **sparkOutputR4**
- Determina el número de ocurrencias que tiene en el fichero de entrada la etiqueta XML "**<page>**" en el fichero de salida de Spark
  - `cat /tmp/sparkOutputR4 | grep "<page>"`



Mira la transparencia 61 ¿Debería coincidir siempre el resultado obtenido con Spark con el obtenido con Hadoop MapReduce o ha podido ser simple casualidad?

```
hadoop@rre-aisi2223-master:~$ hdfs dfs -getmerge /spark-output4 /tmp/sparkOutputR4
hadoop@rre-aisi2223-master:~$ ls -lh /tmp/sparkOutputR4
-rw-r--r-- 1 hadoop hadoop 145M Feb 17 12:50 /tmp/sparkOutputR4
hadoop@rre-aisi2223-master:~$ cat /tmp/sparkOutputR4 | grep "<page>"
<page> 134342
hadoop@rre-aisi2223-master:~$
```



# Ejercicio 8: Limpieza

70



- Elimina el directorio temporal **/tmp** en HDFS junto con todos sus ficheros de forma recursiva
  - Consulta la ayuda de la operación de HDFS: [rm](#)
- A continuación, muestra el **estado actual de HDFS**:
  - Lista de forma recursiva todo el contenido del directorio raíz (/) de HDFS
  - Obtén información del sistema de ficheros

```
hadoop@rre-aisi2223-master:~$ hdfs dfs -rm -R /tmp
Deleted /tmp
hadoop@rre-aisi2223-master:~$ hdfs dfs -ls -R -h /
drwxr-xr-x  - hadoop supergroup          0 2023-02-16 09:29 /hadoop-input
-rw-r--r--  2 hadoop supergroup    452.5 M 2023-02-16 09:29 /hadoop-input/file.xml
drwxr-xr-x  - hadoop supergroup          0 2023-02-17 11:37 /hadoop-output1
-rw-r--r--  2 hadoop supergroup          0 2023-02-17 11:37 /hadoop-output1/_SUCCESS
-rw-r--r--  2 hadoop supergroup   144.9 M 2023-02-17 11:37 /hadoop-output1/part-r-00000
drwxr-xr-x  - hadoop supergroup          0 2023-02-17 11:42 /hadoop-output4
-rw-r--r--  2 hadoop supergroup          0 2023-02-17 11:42 /hadoop-output4/_SUCCESS
-rw-r--r--  2 hadoop supergroup          0 2023-02-17 11:42 /hadoop-output4/part-r-00000
-rw-r--r--  2 hadoop supergroup          36.0 M 2023-02-17 11:42 /hadoop-output4/part-r-00001
-rw-r--r--  2 hadoop supergroup          36.0 M 2023-02-17 11:42 /hadoop-output4/part-r-00002
-rw-r--r--  2 hadoop supergroup          36.3 M 2023-02-17 11:42 /hadoop-output4/part-r-00003
-rw-r--r--  2 hadoop supergroup          36.6 M 2023-02-17 11:42 /hadoop-output4/part-r-00004
drwxr-xr-x  - hadoop supergroup          0 2023-02-17 12:45 /spark-output1
-rw-r--r--  2 hadoop supergroup          0 2023-02-17 12:45 /spark-output1/_SUCCESS
-rw-r--r--  2 hadoop supergroup          0 2023-02-17 12:45 /spark-output1/part-r-00000
drwxr-xr-x  - hadoop supergroup          0 2023-02-17 12:18 /spark-output4
-rw-r--r--  2 hadoop supergroup          0 2023-02-17 12:18 /spark-output4/_SUCCESS
-rw-r--r--  2 hadoop supergroup          36.4 M 2023-02-17 12:18 /spark-output4/part-r-00000
-rw-r--r--  2 hadoop supergroup          36.0 M 2023-02-17 12:18 /spark-output4/part-r-00001
-rw-r--r--  2 hadoop supergroup          36.2 M 2023-02-17 12:18 /spark-output4/part-r-00002
-rw-r--r--  2 hadoop supergroup          36.4 M 2023-02-17 12:18 /spark-output4/part-r-00003
drwxr-xr-x  - hadoop supergroup          0 2023-02-16 16:00 /user
drwxr-xr-x  - hadoop supergroup          0 2023-02-16 16:00 /user/hadoop
drwxr-xr-x  - hadoop supergroup          0 2023-02-17 12:45 /user/hadoop/.sparkStaging
hadoop@rre-aisi2223-master:~$ 
hadoop@rre-aisi2223-master:~$ 
hadoop@rre-aisi2223-master:~$ hdfs dfs -df -h
Filesystem           Size   Used  Available  Use%
hdfs://rre-aisi2223-master:9000  39.0 G  2.0 G   30.9 G   5%
```



# Ejercicio 8: Limpieza

71

- **Para terminar la práctica, detén los servicios HDFS y YARN**
  - De nuevo es preferible usar los scripts *stop-dfs.sh* y *stop-yarn.sh*

```
hadoop@rre-aisi2223-master:~$ stop-yarn.sh
Stopping nodemanagers
Stopping resourcemanager
hadoop@rre-aisi2223-master:~$ stop-dfs.sh
Stopping namenodes on [rre-aisi2223-master]
Stopping datanodes
Stopping secondary namenodes [rre-aisi2223-master]
hadoop@rre-aisi2223-master:~$
```