

---

# **BIG DATA**

# **YARN MAPREDUCE**

**EDUARD LARA**

# 1. INTRODUCCIÓN

---

- Hasta ahora hemos visto la parte de gestión de datos de Hadoop: HDFS, creación y subida de ficheros, etc.
  - Ahora veremos la parte de los procesos de Hadoop
  - Esta compuesto de dos versiones distintas:
    - MAPREDUCE V1, la versión 1 de hadoop. No es la más recomendable a dia de hoy.
    - MAPREDUCE V2, mas conocida como YARN. Es la versión que se normalmente se utiliza ya
  - NOTA: MapReduce V1 sigue utilizándose con YARN.
  - Veremos que MapReduce V2 tiene la V1 de MapReduce como un componente más.
-

# 1. INTRODUCCION

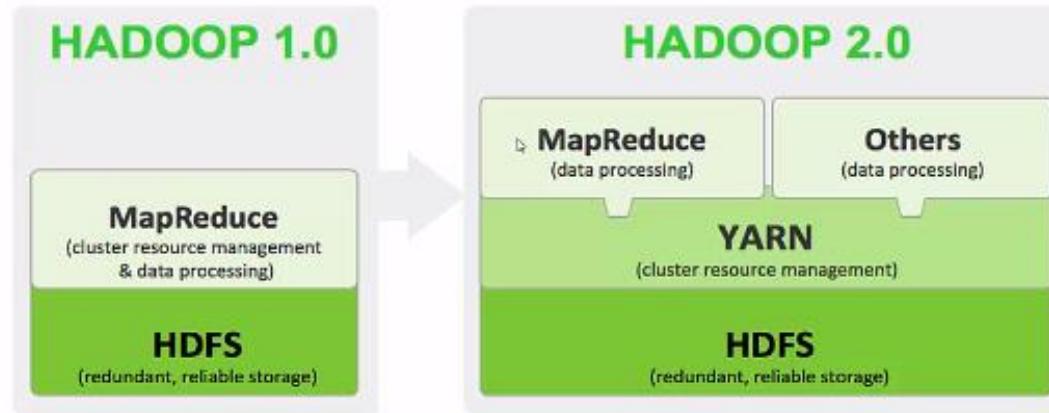
---

- MapReduce V1
  - Estaba pensada para procesos Batch.
  - Se encarga tanto de la parte de procesamiento de los datos como de la parte de la gestión del cluster
- MapReduce V2 - Yarn
  - Admite otra tipo de productos y procesos que no sean sólo batch
  - Tiene procesos distintos para el proceso de datos y para la gestión del clúster.
  - Yarn ha separado la parte de la gestión de la parte del proceso consiguiendo un mejor rendimiento.

# 1. INTRODUCCIÓN

---

- Diferencias entre V1 y V2.



- En Hadoop V1 MapReduce se encargaba de toda la gestión del cluster y del procesamiento de los datos
- En Hadoop V2, Yarn es el gestor del cluster. MapReduce ha pasado de ser el core central a ser un componente mas de Hadoop. En la parte superior podemos tener procesos MapReduce (de la V1) o procesos de otro tipo.

# 1. INTRODUCCIÓN

---

## MapReduce V1

- Tiene problemas de escalabilidad a partir de los 5000 nodos. Hay muchas empresas grandes que tienen muchos más de 5000 nodos de Cluster hadoop
- Tiene graves problemas de rendimiento sobretodo en aplicaciones que no han sido hechas propiamente para ser batch
- No es capaz de gestionar correctamente los recursos.

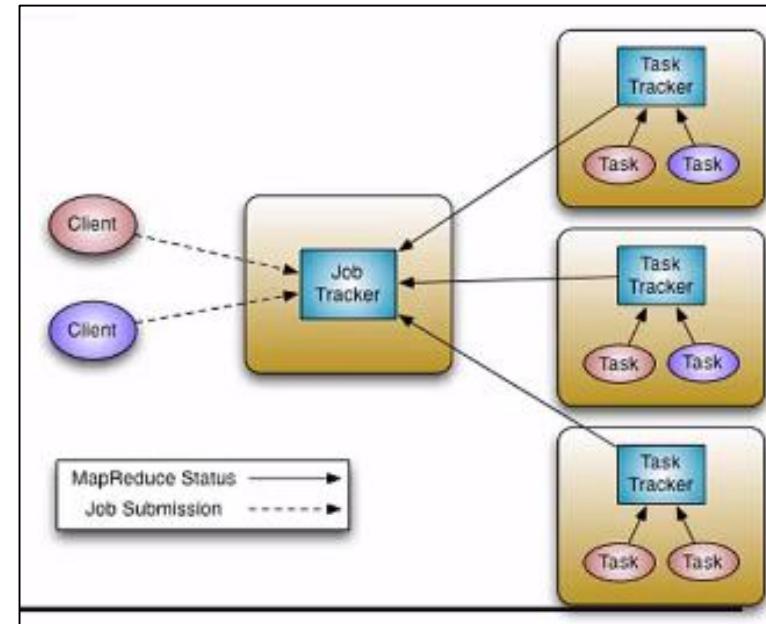
# 1. INTRODUCCIÓN

---

## MapReduce V1

Tenía 2 procesos denominados:

- Job Tracker
  - En el maestro
  - Gestionaba los recursos del cluster y la planificación de cada uno de los trabajos.
- TaskTracker
  - Agente en los esclavos
  - Gestionaba las tareas y recursos en los nodos esclavos



# 1. INTRODUCCIÓN

---

## Yarn

- Ha cambiado completamente la arquitectura de procesos



- Ahora MapReduce se reduce a una pequeña parte, pero podemos poner otros componentes que no sean Batch.
- Tez, Hbase, Storm, Giraph, Spark que son determinados productos que se han ido subiendo al carro de Big Data en los últimos tiempos y que no son procesos batch

# 1. INTRODUCCIÓN

---

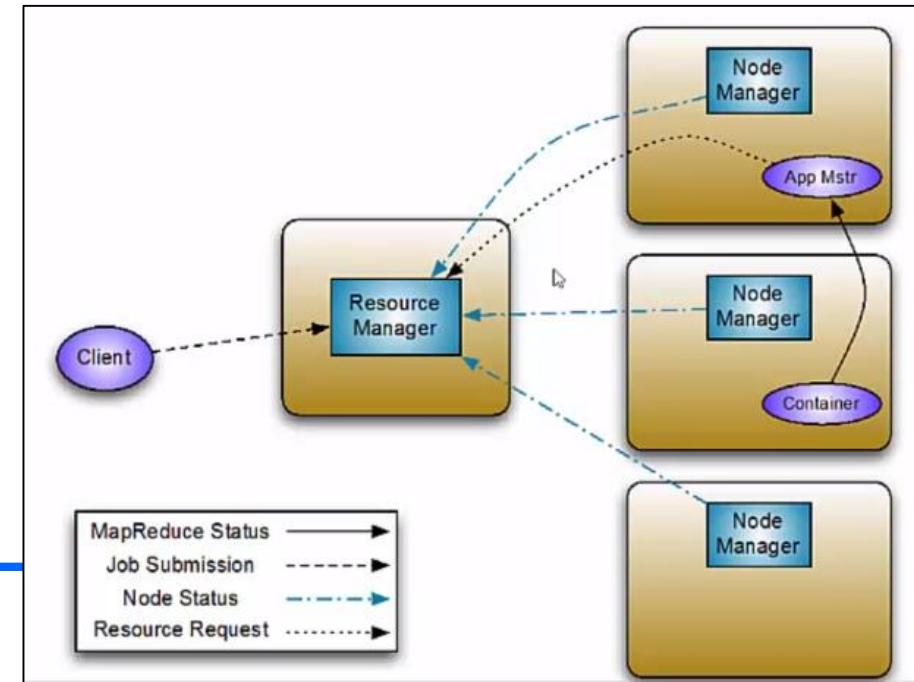
## Yarn

- La capa de Yarn que sería la capa del gestor del cluster
- HDFS versión 2 que ha tenido mejoras con respecto a versión 1
- Ya hemos trabajado anteriormente con HDFS 2
- Por ejemplo los snapshots corresponden esta versión
- La gestión que se hace a nivel de Yarn es mucho más efectiva que la que se hace a nivel de MapReduce

## 2. FUNCIONAMIENTO YARN

Los principales procesos de Yarn son:

- 1) **ResourceManager (RM)**: En el maestro. Gestiona los recursos del cluster: que nodos hay, quién necesita más.
- 2) **Nodemanager (NM)**: En cada uno de los esclavos. Se encarga de gestionar los recursos de cada nodo
  - Ya no hay un solo proceso para hacer eso como en la v1. En cada nodo hay un gestor que se encarga de gestionar estos recursos



## 2. FUNCIONAMIENTO YARN

---

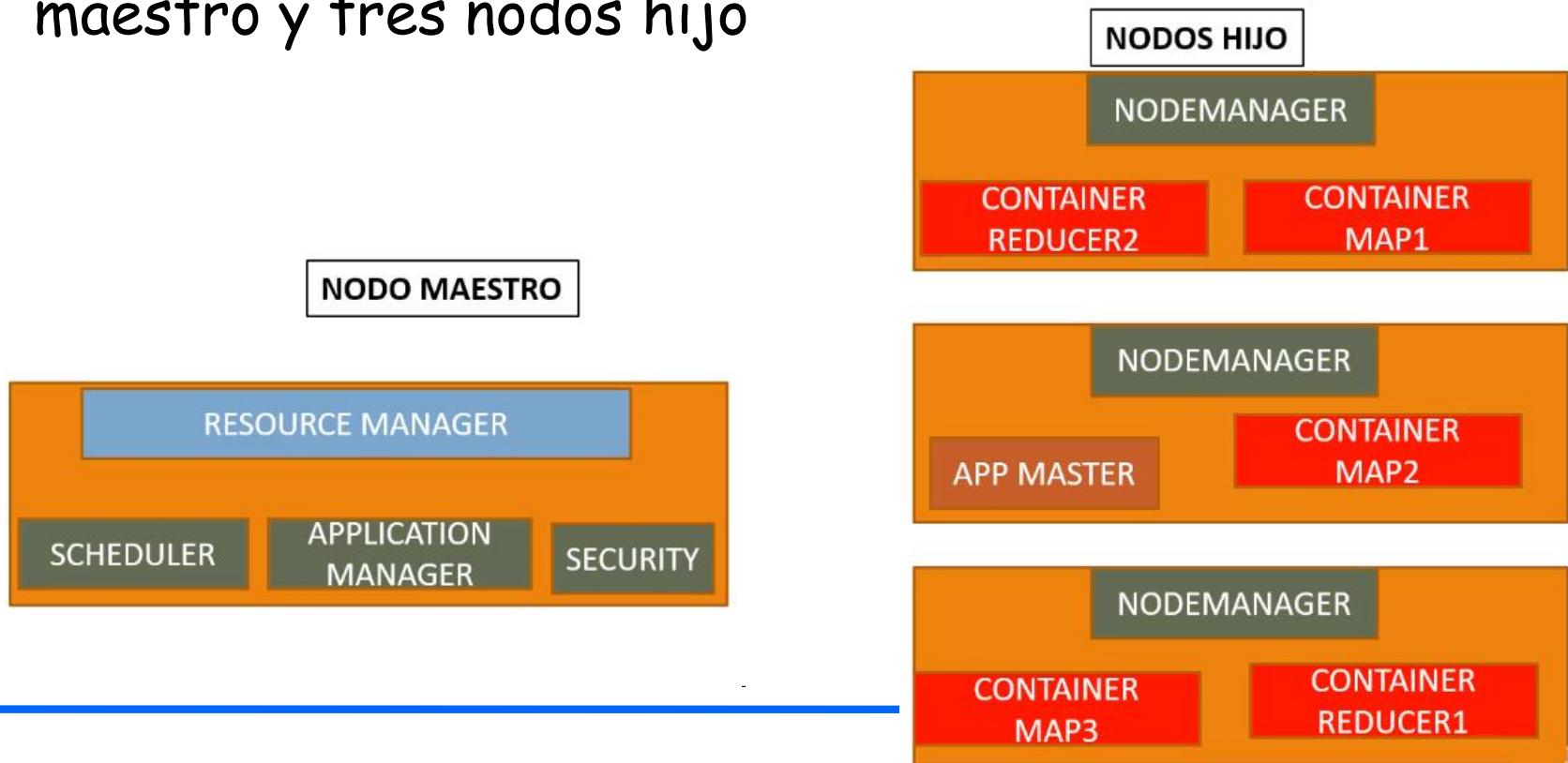
3) **ApplicationMaster (AM)**. Vamos a tener uno por aplicación.

- Se encarga de gestionar los recursos que necesita la aplicación su ciclo de vida y la planificación de la aplicación
- Va a arrancar determinados contenedores que van a permitir que se gestione mejor todos los recursos del cluster.
- Ya no sólo son procesos batch, sino que estamos utilizando otro tipo de recursos.

## 2. FUNCIONAMIENTO YARN

---

- Funcionamiento de los procesos Yarn y cómo se integran unos con otros.
- Suponemos una arquitectura de 4 servidores uno de maestro y tres nodos hijo



## 2. FUNCIONAMIENTO YARN

---

- En el **Nodo Maestro** se ejecuta el **Resource Manager**: Gestor de todo Yarn. Tiene 3 subcomponentes:
  - **Scheduler**: Determina la planificación de trabajos. Determina los recursos en un nodo que el App Master va a utilizar. Hay dos schedulers. El más habitual es el capacity Scheduler → el planificador de capacidad.
  - **Application Manager**: Cuando alguien solicita ejecutar una aplicación, arranca un App Master, en los nodos para que se encargue de esa aplicación
  - **Security**: Implementa medidas de seguridad

NODO MAESTRO



## 2. FUNCIONAMIENTO YARN

---

- En cada nodo hijo tenemos el NodeManager.
- Cuando se lanza una aplicación, primero se solicita al Application Manager que abra un App Master.
- Application Manager tendrá en cuenta el Scheduler para decidir en qué nodo va a arrancar el App Master
- App Master será el coordinador de la aplicación lanzada, si tenemos 100 aplicaciones lanzadas dentro de un clúster, tendremos 100 procesos App Master.
- Este es uno de los motivos por el que en los Clusters de tipo Yarn pueden escalar tanto

## 2. FUNCIONAMIENTO YARN

---

- El **App Master** junto con la información suministrada por el Scheduler, determinará en qué nodos va a ir arrancando los containers. Un container es un recurso de memoria y CPU donde se ejecuta algo
- En el ejemplo el APP Master ha arrancado 3 containers en los 3 nodos hijo, donde se ejecuta un mapper (app de tipo MapReduce). Y 2 contenedores MapReduce
- Cada proceso tiene que acceder a su bloque de datos en local. El bloque de datos de cada mapper está en el nodo donde se han lanzado
- Yarn es un gestor completo de recursos que no hace nada de trabajo de aplicación sino que sencillamente se encarga de gestionar los recursos del cluster.

# 3. CONFIGURAR YARN EN UN CLUSTER

---

Paso 1. Montaremos nuestro primer cluster con Yarn. Hasta ahora hemos arrancado solo la parte de los datos. Después de arrancar el sistema de ficheros HDFS con **start-dfs.sh**, mediante el comando **jps** vemos que tenemos tres procesos:

- datanode
- secondary namenode
- Namenode

```
hadoop@nodo1:~$ start-dfs.sh
Starting namenodes on [nodo1]
nodo1: starting namenode, logging to /opt/hadoop/logs/hadoop-hadoop-namenode-nodo1.out
localhost: starting datanode, logging to /opt/hadoop/logs/hadoop-hadoop-datanode-nodo1.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /opt/hadoop/logs/hadoop-hadoop-secondarynamenode-nodo1.out
hadoop@nodo1:~$ jps
9730 SecondaryNameNode
18163 Jps
9194 NameNode
9388 DataNode
hadoop@nodo1:~$
```

### 3. CONFIGURAR YARN EN UN CLUSTER

---

Paso 2 . Ahora montaremos la parte de los procesos. Para ello no usaremos la versión 1 de MapReduce puesto que es obsoleta.  
Montaremos la versión 2 denominada Yarn.

Con **stop-dfs.sh** pararemos nuestro cluster. Se ve como estamos parando:

- el namenode
- el Datanode
- y por último el secondarynamenode.

```
hadoop@nodo1:~$ stop-dfs.sh
Stopping namenodes on [nodo1]
nodo1: stopping namenode
localhost: stopping datanode
Stopping secondary namenodes [0.0.0.0]
0.0.0.0: stopping secondarynamenode
hadoop@nodo1:~$
```

# 3. CONFIGURAR YARN EN UN CLUSTER

---

## Ficheros de configuración

- Los ficheros de configuración estan en /opt/hadoop/etc/hadoop
- Para arrancar la parte de HDFS hemos tocado el **core-site.xml** y el **hdfs-site.xml**
- Para montar la parte de procesos Yarn tendremos que tocar otros dos:
  - **mapred-site.xml**
  - **yarn-site.xml**

```
hadoop@nodo1:/opt/hadoop/etc/hadoop$ ls
capacity-scheduler.xml      httpfs-env.sh          mapred-env.sh
configuration.xsl            httpfs-log4j.properties  mapred-queues.xml.template
container-executor.cfg       httpfs-signature.secret mapred-site.xml.template
core-site.xml                httpfs-site.xml        slaves
hadoop-env.cmd               kms-acls.xml         ssl-client.xml.example
hadoop-env.sh                kms-env.sh           ssl-server.xml.example
hadoop-metrics2.properties   kms-log4j.properties yarn-env.cmd
hadoop-metrics.properties    kms-site.xml         yarn-env.sh
hadoop-policy.xml            log4j.properties     yarn-site.xml
hdfs-site.xml                mapred-env.cmd
hadoop@nodo1:/opt/hadoop/etc/hadoop$
```

# 3. CONFIGURAR YARN EN UN CLUSTER

---

Paso 3. Modificaremos el primer fichero de configuración:

- Primero debemos hacer una copia de mapred-site.xml.template:

```
hadoop@nodo1:/opt/hadoop/etc/hadoop$ cp mapred-site.xml.template mapred-site.xml  
hadoop@nodo1:/opt/hadoop/etc/hadoop$ █
```

- Editaremos el fichero mapred-site.xml con el siguiente contenido:

```
GNU nano 6.2                               mapred-site.xml

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

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

- Indicamos que el motor que va a gestionar hadoop va a ser de tipo Yarn, mediante la propiedad mapreduce.framework.name, es decir el Framework que se va a utilizar va a ser Yarn

# 3. CONFIGURAR YARN EN UN CLUSTER

---

Paso 4. El segundo fichero que tenemos que tocar es el yarn-site.xml

- Indicaremos 3 propiedades

```
GNU nano 6.2                                                 yarn-site.xml
limitations under the License. See accompanying LICENSE file.
-->
<configuration>

<!-- Site specific YARN configuration properties -->
<property>
    <name>yarn.resourcemanager.hostname</name>
    <value>localhost</value>
</property>
<property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
</property>
<property>
    <name>yarn.nodemanager.aux-services.mapreduce_shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
</configuration>
```

### 3. CONFIGURAR YARN EN UN CLUSTER

---

- La 1º propiedad especifica el nombre de la máquina donde se ejecuta el proceso resourcemanager, el gestor del clúster Yarn. Se indica el maestro que va a gestionar los procesos Yarn
- Las otras dos propiedades indican la clase que gestiona el Yarn
- Una indica que el gestor de servicios auxiliares que va a gestionar el MapReduce vale mapreduce\_shuffle
- En la otra propiedad indicamos que la clase mapreduce\_shuffle que se va a usar para hacer esa gestión, es una clase hadoop:  
**org.apache.hadoop.mapred.ShuffleHandler**
- Este es el valor por defecto que tiene la arquitectura de Yarn

## 4. INICIAR EL CLUSTER YARN

---

**Paso 1.** Una vez configurados los ficheros, para iniciar el cluster primero arrancamos siempre HDFS (los datos) con **start-dfs.sh**: se inicia el namenode, el datanode y el secondary namenode

```
hadoop@nodo1:/opt/hadoop/etc/hadoop$ start-dfs.sh
Starting namenodes on [nodo1]
nodo1: starting namenode, logging to /opt/hadoop/logs/hadoop-hadoop-namenode-nodo1.out
localhost: starting datanode, logging to /opt/hadoop/logs/hadoop-hadoop-datanode-nodo1.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /opt/hadoop/logs/hadoop-hadoop-secondarynamenode-nodo1.out
hadoop@nodo1:/opt/hadoop/etc/hadoop$
```

**Paso 2.** Arrancamos la segunda parte del cluster que es el Yarn, mediante el comando **start-yarn.sh**. Si no nos hemos equivocado en los fichero de configuración nos va a arrancar el resource manager y el nodemanager. El applicationmanager no aparece hasta que no arrancamos una aplicación

```
hadoop@nodo1:/opt/hadoop/etc/hadoop$ start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /opt/hadoop/logs/yarn-hadoop-resourcemanager-nodo1.out
localhost: starting nodemanager, logging to /opt/hadoop/logs/yarn-hadoop-nodemanager-nodo1.out
hadoop@nodo1:/opt/hadoop/etc/hadoop$
```

## 4. INICIAR EL CLUSTER YARN

---

Paso 3. Mediante el comando jps podemos ver que tenemos todos los procesos en el mismo nodo:

- los de datos: DataNode, NameNode y SecundaryNameNode
- los de procesos: ResourceManager, NodeManager

```
hadoop@nodo1:/opt/hadoop/etc/hadoop$ jps
687130 NodeManager
683090 NameNode
688550 Jps
686935 ResourceManager
683668 SecondaryNameNode
683301 DataNode
hadoop@nodo1:/opt/hadoop/etc/hadoop$
```

NOTA: Hemos visto cómo configurar y arrancar de una manera muy rápida nuestro primer cluster Yarn completo, es decir con HFS y con Yarn, dentro de un solo nodo.

---

# 5. WEB ADMINISTRACION YARN

Paso 1. De la misma manera que tenemos una página web donde podemos ver la configuración HDFS, también disponemos de una página parecida para la configuración de YARN

- nodo1:50070 o localhost:50070 → para HDFS
- nodo1:8088 o localhost:8088 → para YARN

The screenshot shows the HDFS Health Overview page. At the top, there's a navigation bar with tabs: Hadoop (selected), Overview, Datanodes, Datanode Volume Failures, Snapshot, and Start. Below the navigation bar, the title is "Overview 'nodo1:9000' (active)". A table provides detailed information about the cluster:

Started:	Sat Feb 11 23:45:48 +0000 2023
Version:	2.10.2, r965fd380006fa78b2315668fbc7eb432e1d8200
Compiled:	Tue May 24 22:35:00 +0000 2022 by ubuntu from branch-2.10
Cluster ID:	CID-18867706-b47d-445c-9ff2-53ed060664b6
Block Pool ID:	BP-703191218-127.0.1.1-1673434148125

The screenshot shows the YARN Cluster Overview page. At the top, there's a navigation bar with tabs: Overview (selected), Applications, Node Labels, Applications, and Scheduler. Below the navigation bar, the title is "Cluster Metrics". On the left, there's a sidebar with links: Cluster (About, Nodes, Node Labels, Applications, NEW, NEW\_SAVING, SUBMITTED, ACCEPTED, RUNNING, FINISHED, FAILED, KILLED), Applications, and Scheduler.

**Cluster Metrics**

Apps Submitted	Apps Pending	Apps Running	App
0	0	0	0

**Cluster Nodes Metrics**

Active Nodes	Decommissioning Nod
1	0

**Scheduler Metrics**

Scheduler Type	Capacity Scheduler [name=memory-mb default-unit=
Capacity Scheduler	[name=memory-mb default-unit=

**Show 20 entries**

ID	User	Name	Application Type	Queue	Application Priority	Star

# 5. WEB ADMINISTRACION YARN

Paso 2. En la ventana principal **Cluster** aparecen un montón de opciones a la izquierda. Es un poco distinta a la de HDFS:

The screenshot shows the YARN Cluster web interface. On the left, there is a sidebar with the following navigation options:

- Cluster
- About
- Nodes
- Node Labels
- Applications** (highlighted)
- NEW
- NEW SAVING
- SUBMITTED
- ACCEPTED
- RUNNING
- FINISHED
- FAILED
- KILLED
- Scheduler
- Tools

The main content area is titled "All Applications". It displays various cluster metrics and application details.

**Cluster Metrics:**

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Used Resources	Total Resources	Reserved Resources	Physical Mem Used %	Physical Vcores Used %
0	0	0	0	0	<memory:0 B, vCores:0>	<memory:8 GB, vCores:8>	<memory:0 B, vCores:0>	79	25

**Cluster Nodes Metrics:**

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes
1	0	0	0	0	0	0

**Scheduler Metrics:**

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority
Capacity Scheduler	[<name=memory-mb default-unit=Mi type=COUNTABLE>, <name=vcores default-unit= type=COUNTABLE>]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0

**Applications Table:**

ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU Vcores	Allocated Memory MB	Allocated GPUs	Reserved CPU Vcores	Reserved Memory MB	Reserved GPUs	% of Queue	% of Cluster	Progress	Tracking UI	Blacklisted Nodes
No data available in table																						

At the bottom of the table, it says "Showing 0 to 0 of 0 entries". There are also links for "First", "Previous", "Next", and "Last".

Primero vemos todas las métricas del cluster junto con la planificación que hay reservada para aplicaciones (ahora no tenemos ninguna). Podemos ver: memoria física utilizada, cores usados, etc

# 5. WEB ADMINISTRACION YARN

Paso 3. En la opción de **Cluster/About** nos permite ver información de todo nuestro cluster:

- cuando ha arrancado
- si está activo o no el resourcemanager
- la memoria que se está utilizando (por defecto asume que cada nodo debería tener un mínimo 1GB y un máximo 8GB), etc

About the Cluster																
Cluster Metrics																
Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Used Resources	Total Resources	Reserved Resources	Physical Mem Used %	Physical VCores Used %							
0	0	0	0	0	<memory:0 B, vCores:0>	<memory:8 GB, vCores:8>	<memory:0 B, vCores:0>	81	0							
Cluster Nodes Metrics																
Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes										
1	0	0	0	0	0	0										
Scheduler Metrics																
Scheduler Type	Scheduling Resource Type				Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority									
Capacity Scheduler	<name=memory-mb default-unit=Mi type=COUNTABLE>, <name=vcores default-unit= type=COUNTABLE>				<memory:1024, vCores:1>	<memory:8192, vCores:4>	0									
Cluster overview																
<b>Cluster ID:</b> 1676206880351 <b>ResourceManager state:</b> STARTED <b>ResourceManager HA state:</b> active <b>ResourceManager HA zookeeper connection state:</b> Could not find leader elector. Verify both HA and automatic failover are enabled. <b>ResourceManager RMStateStore:</b> org.apache.hadoop.yarn.server.resourcemanager.recovery.NullRMStateStore <b>ResourceManager started on:</b> dom feb 12 13:01:20 +0000 2023 <b>ResourceManager version:</b> 2.10.2 from 965fd380006fa78b2315668fbc7eb432e1d8200f by ubuntu source checksum 332d49624f6d2fd6fb142861335c939 on 2022-05-24T22:40Z <b>Hadoop version:</b> 2.10.2 from 965fd380006fa78b2315668fbc7eb432e1d8200f by ubuntu source checksum d3ab737f7788f05d467784f0a86573fe on 2022-05-24T22:35Z																

# 5. WEB ADMINISTRACION YARN

Paso 4. La opción **Cluster/Nodes** nos permita ver los nodos que tenemos activos dentro del cluster. En este momento solo tenemos uno, con el que nos podemos comunicar por el puerto 8042.

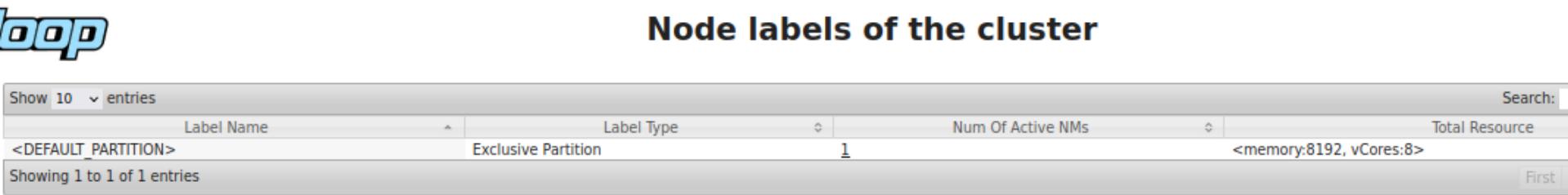
Nodes of the cluster																
Cluster Metrics																
Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Used Resources	Total Resources	Reserved Resources	Physical Mem Used %	Physical Vcores Used %							
0	0	0	0	0	<memory:0 B, vCores:0>	<memory:8 GB, vCores:8>	<memory:0 B, vCores:0>	81	0							
Cluster Nodes Metrics																
Active Nodes	Decommissioning Nodes			Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes								
1	0	0		0	0	0	0	0	0							
Scheduler Metrics																
Scheduler Type	Scheduling Resource Type							Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority						
Capacity Scheduler	<name=memory-mb default-unit=Mi type=COUNTABLE>, <name=vcores default-unit= type=COUNTABLE>]							<memory:1024, vCores:1>	<memory:8192, vCores:4>	0						
Show 20	▼ entries	Search:														
Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Mem Used	Mem Avail	Phys Mem Used %	Vcores Used	Vcores Avail	Phys Vcores Used %	GPUs Used	GPUs Avail	Version
/default-rack	RUNNING	nodo1:43807	<a href="#">nodo1:8042</a>		dom feb 12 13:27:15 +0000 2023		0	0 B	8 GB	81	0	8	6	0	0	2.10.2

Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update
/default-rack	RUNNING	nodo1:43807	<a href="#">nodo1:8042</a>		dom feb 12 13:27:15 +0000 2023

# 5. WEB ADMINISTRACION YARN

---

Paso 5. En la opción **Cluster/Node Labels**, se muestran las etiquetas que podemos poner a cada nodo



The screenshot shows a table titled "Node labels of the cluster" in the YARN Web UI. The table has columns for Label Name, Label Type, Num Of Active NMs, and Total Resource. There is one entry: <DEFAULT\_PARTITION> which is an Exclusive Partition with 1 active NM and a total resource of <memory:8192, vCores:8>. The table includes standard UI elements like a search bar, a "Show 10 entries" dropdown, and a "First" link.

Node labels of the cluster				
Label Name	Label Type	Num Of Active NMs	Total Resource	Search:
<DEFAULT_PARTITION>	Exclusive Partition	1	<memory:8192, vCores:8>	
Showing 1 to 1 of 1 entries				

# 5. WEB ADMINISTRACION YARN

Paso 6. En la pestaña **Clusters/Applications**, vemos las aplicaciones que tenemos en este momento funcionando. Se puede filtrar por estado: nuevas, lanzadas, aceptadas, ejecutándose, terminadas, que han fallado, etc.

The screenshot shows the Apache YARN web interface under the 'All Applications' tab. The top navigation bar includes 'Cluster Metrics', 'Cluster Nodes Metrics', 'Scheduler Metrics', and a search bar. The main content area displays various metrics and a table of applications.

**Cluster Metrics:**

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Used Resources	Total Resources	Reserved Resources	Physical Mem Used %	Physical VCores Used %
0	0	0	0	0	<memory:0 B, vCores:0>	<memory:8 GB, vCores:8>	<memory:0 B, vCores:0>	81	0

**Cluster Nodes Metrics:**

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes
1	0	0	0	0	0	0

**Scheduler Metrics:**

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority
Capacity Scheduler	[<name=memory-mb default-unit=Mi type=COUNTABLE>, <name=vcores default-unit= type=COUNTABLE>]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0

**Show 20 entries** **Search:**

ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Allocated GPUs	Reserved CPU VCores	Reserved Memory MB	Reserved GPUs	% of Queue	% of Cluster	Progress	Tracking UI	Blacklisted Nodes	
No data available in table																							

**Applications:**

- NEW**
- NEW\_SAVING**
- SUBMITTED**
- ACCEPTED**
- RUNNING**
- FINISHED**
- FAILED**
- KILLED**

Evidentemente en este momento no tenemos nada, porque todavía no hemos lanzado ninguna aplicación

# 5. WEB ADMINISTRACION YARN

Paso 7. En el pestaña **Cluster/Scheduler** podemos ver todas las métricas del planificador: que capacidad tenemos. cuánto se ha utilizado, cuánto se ha utilizado sobre la capacidad actual o permitida. Esta vacío puesto que no hemos lanzado ninguna aplicación. Una vez que se haya lanzado algún aplicativo permite ver lo que ha sucedido y los recursos que se han estado utilizando a nivel del cluster.

The screenshot shows the YARN Cluster/Scheduler web interface. On the left, there is a sidebar with navigation links: Cluster, About, Nodes, Node Labels, Applications, and Scheduler. The Applications link is currently selected and highlighted in blue. The main content area has a title "NEW, NEW\_SAVING, SUBMITTED, ACCEPTED, RUNNING Applications". Below the title, there are several tables and sections:

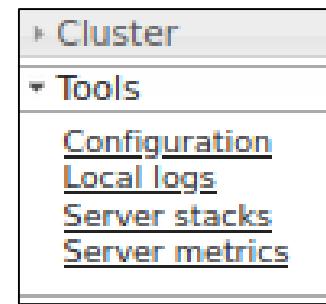
- Metrics:** Shows counters for Apps Pending (0), Apps Running (0), Apps Completed (0), Containers Running (0), Used Resources (<memory:0 B, vCores:0>), Total Resources (<memory:8 GB, vCores:8>), Reserved Resources (<memory:0 B, vCores:0>), Physical Mem Used % (81), and Physical VCores Used % (0).
- Nodes Metrics:** Shows counts for Active Nodes (0), Decommissioning Nodes (0), Decommissioned Nodes (0), Lost Nodes (0), Unhealthy Nodes (0), Rebooted Nodes (0), and Shutdown Nodes (0).
- Scheduler Metrics:** Shows Scheduling Resource Type (<name=memory-mb default-unit=Mi type=COUNTABLE>, <name=vcores default-unit= type=COUNTABLE>), Minimum Allocation (<memory:1024, vCores:1>), Maximum Allocation (<memory:8192, vCores:4>), and Maximum Cluster Application Priority (0).
- Scheduler logs:** A dropdown menu set to 1 min, showing log entries for root and default queues.
- Queues:** A table showing queue usage: Queue: root (0.0% used) and Queue: default (0.0% used).
- entries:** A table header for tracking application entries, with columns: Name, Application Type, Queue, Application Priority, StartTime, LaunchTime, FinishTime, State, FinalStatus, Running Containers, Allocated CPU VCores, Allocated Memory MB, Allocated GPUs, Reserved CPU VCores, Reserved Memory MB, Reserved GPUs, % of Queue, % of Cluster, Progress, Tracking UI, and Blacklisted Nodes. The message "No data available in table" is displayed.
- Showing 0 to 0 of 0 entries**
- Aggregate scheduler counts:** Total Container Allocations(count), Total Container Releases(count), Total Fulfilled Reservations(count), and Total Container Preemptions(count).

# 5. WEB ADMINISTRACION YARN

Paso 8. En la pestaña Tools/Configuration podemos ver la configuración de este nodo del cluster.

A screenshot of a web browser window displaying the configuration XML for a YARN node. The URL is `localhost:8088/conf`. The page shows a large block of XML code with syntax highlighting for tags like `<configuration>`, `<property>`, `<name>`, `<value>`, and `<final>`. The XML content includes properties for opportunistic container allocation, RPC class, mapreduce job max task failures, and speculative retries.

```
<configuration>
<property>
<name>yarn.resourcemanager.opportunistic-container-allocation.enabled</name>
<value>false</value>
<final>false</final>
<source>yarn-default.xml</source>
</property>
<property>
<name>yarn.ipc.rpc.class</name>
<value>org.apache.hadoop.yarn.ipc.HadoopYarnProtoRPC</value>
<final>false</final>
<source>yarn-default.xml</source>
</property>
<property>
<name>mapreduce.job.maxtaskfailures.per.tracker</name>
<value>3</value>
<final>false</final>
<source>mapred-default.xml</source>
</property>
<property>
<name>mapreduce.job.speculative.retry-after-speculate</name>
<value>15000</value>
</property>
```

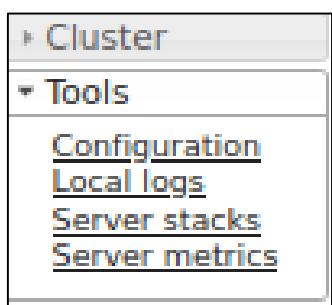


Hadoop tiene muchas propiedades de las cuales sólo hemos tocado 2, las que nos han interesado. Las que no ponemos se ponen por defecto

# 5. WEB ADMINISTRACION YARN

---

Paso 9. En Tools/Local Logs podemos ver los log locales. En realidad tenemos los log de todos los procesos del cluster: datanode, namenode, Secondarynamenode, ResourceManager, etc



Directory: /logs/			
<a href="#">SecurityAuth-hadoop.audit</a>	0 bytes	14-ene-2023	20:03:59
<a href="#">hadoop-hadoop-datanode-nodo1.log</a>	146573 bytes	12-feb-2023	12:57:49
<a href="#">hadoop-hadoop-datanode-nodo1.out</a>	844 bytes	12-feb-2023	12:57:34
<a href="#">hadoop-hadoop-datanode-nodo1.out.1</a>	844 bytes	11-feb-2023	23:45:48
<a href="#">hadoop-hadoop-datanode-nodo1.out.2</a>	844 bytes	11-feb-2023	17:02:02
<a href="#">hadoop-hadoop-datanode-nodo1.out.3</a>	844 bytes	31-ene-2023	11:12:51
<a href="#">hadoop-hadoop-datanode-nodo1.out.4</a>	844 bytes	14-ene-2023	20:04:03
<a href="#">hadoop-hadoop-namenode-nodo1.log</a>	289964 bytes	12-feb-2023	13:58:58
<a href="#">hadoop-hadoop-namenode-nodo1.out</a>	844 bytes	12-feb-2023	12:57:28
<a href="#">hadoop-hadoop-namenode-nodo1.out.1</a>	844 bytes	11-feb-2023	23:45:43
<a href="#">hadoop-hadoop-namenode-nodo1.out.2</a>	844 bytes	11-feb-2023	17:01:57
<a href="#">hadoop-hadoop-namenode-nodo1.out.3</a>	6161 bytes	01-feb-2023	9:20:30
<a href="#">hadoop-hadoop-namenode-nodo1.out.4</a>	6161 bytes	29-ene-2023	20:38:25
<a href="#">hadoop-hadoop-secondarynamenode-nodo1.log</a>	280197 bytes	12-feb-2023	13:58:58
<a href="#">hadoop-hadoop-secondarynamenode-nodo1.out</a>	844 bytes	12-feb-2023	12:57:44

# 5. WEB ADMINISTRACION YARN

Paso 10. También podemos ver el Server stacks (en cluster/Server stacks) y las métricas del server (en cluster/Server metrics): qué cosas se han lanzado, que tengo libre, estadísticas de uso, etc.

Al profundizar más en los clusters hadoop, estas propiedades o estos logs pueden ser de utilidad.

Process Thread Dump:

198 active threads

Thread 211 (498771968@qtp-350059321-2):

- State: RUNNABLE
- Blocked count: 39
- Waited count: 127
- Stack:

```
sun.management.ThreadImpl.getThreadInfo(Native Method)
sun.management.ThreadImpl.getThreadInfo(ThreadImpl.java:185)
sun.management.ThreadImpl.getThreadInfo(ThreadImpl.java:144)
org.apache.hadoop.util.ReflectionUtils.printThreadInfo(ReflectionUtils.java:185)
org.apache.hadoop.http.HttpServer2$StackServlet doGet(HttpServer2.java:707)
javax.servlet.http.HttpServlet.service(HttpServlet.java:828)
org.mortbay.jetty.servlet.ServletHolder.handle(ServletHolder.java:512)
org.mortbay.jetty.servlet.ServletHandler$CachedChain.doFilter(ServletHandler.java:1075)
com.google.inject.servlet.FilterChainInvocation.doFilter(FilterChainInvocation.java:103)
com.sun.jersey.spi.container.servlet.ServletContainer.doFilter(ServletContainer.java:470)
com.sun.jersey.spi.container.servlet.ServletContainer.doFilter(ServletContainer.java:470)
org.apache.hadoop.yarn.server.resourcemanager.webapp.RMWebAppFilter.doFilter(RMWebAppFilter.java:100)
com.sun.jersey.spi.container.servlet.ServletContainer.doFilter(ServletContainer.java:470)
com.google.inject.servlet.FilterDefinition.doFilter(FilterDefinition.java:103)
com.google.inject.servlet.FilterChainInvocation.doFilter(FilterChainInvocation.java:103)
com.google.inject.servlet.ManagedFilterPipeline.dispatch(ManagedFilterPipeline.java:103)
com.google.inject.servlet.GuiceFilter.doFilter(GuiceFilter.java:113)
org.mortbay.jetty.servlet.ServletHandler$CachedChain.doFilter(ServletHandler.java:1075)
org.apache.hadoop.security.http.XFrameOptionsFilter.doFilter(XFrameOptionsFilter.java:100)
```

Thread 210 (DestroyJavaVM):

- State: RUNNABLE
- Blocked count: 0
- Waited count: 0
- Stack: 

```
sun.management.ThreadImpl.getThreadInfo(Native Method)
sun.management.ThreadImpl.getThreadInfo(ThreadImpl.java:185)
sun.management.ThreadImpl.getThreadInfo(ThreadImpl.java:144)
org.apache.hadoop.util.ReflectionUtils.printThreadInfo(ReflectionUtils.java:185)
org.apache.hadoop.http.HttpServer2$StackServlet doGet(HttpServer2.java:707)
javax.servlet.http.HttpServlet.service(HttpServlet.java:828)
org.mortbay.jetty.servlet.ServletHolder.handle(ServletHolder.java:512)
org.mortbay.jetty.servlet.ServletHandler$CachedChain.doFilter(ServletHandler.java:1075)
com.google.inject.servlet.FilterChainInvocation.doFilter(FilterChainInvocation.java:103)
com.sun.jersey.spi.container.servlet.ServletContainer.doFilter(ServletContainer.java:470)
com.sun.jersey.spi.container.servlet.ServletContainer.doFilter(ServletContainer.java:470)
org.apache.hadoop.yarn.server.resourcemanager.webapp.RMWebAppFilter.doFilter(RMWebAppFilter.java:100)
com.sun.jersey.spi.container.servlet.ServletContainer.doFilter(ServletContainer.java:470)
com.google.inject.servlet.FilterDefinition.doFilter(FilterDefinition.java:103)
com.google.inject.servlet.FilterChainInvocation.doFilter(FilterChainInvocation.java:103)
com.google.inject.servlet.ManagedFilterPipeline.dispatch(ManagedFilterPipeline.java:103)
com.google.inject.servlet.GuiceFilter.doFilter(GuiceFilter.java:113)
org.mortbay.jetty.servlet.ServletHandler$CachedChain.doFilter(ServletHandler.java:1075)
org.apache.hadoop.security.http.XFrameOptionsFilter.doFilter(XFrameOptionsFilter.java:100)
```

localhost:8088/stacks

localhost:8088/jmx?qry=Hadoop:\*

JSON Datos sin procesar Cabeceras

Guardar Copiar Contraer todo Expandir todo Filtrar JSON

beans:

- 0:
  - name: "Hadoop:service=ResourceManager,name=RMNMInfo"
  - modelerType: "org.apache.hadoop.yarn.server.resourcemanager.RMNMInfo"
  - LiveNodeManagers:[{"HostName": "nodo1", "Rack": "/default-rack", "State": "RUNNING", "UsedMemoryMB": 0, "AvailableMemoryMB": 8192}]
- 1:
  - name: "Hadoop:service=ResourceManager,name=RpcActivityForPort8033"
  - modelerType: "RpcActivityForPort8033"
  - tag.port: "8033"
  - tag.Context: "rpc"
  - tag.NumOpenConnectionsPerUser: "{}"
  - tag.Hostname: "nodo1"
  - ReceivedBytes: 0
  - SentBytes: 0
  - RpcQueueTimeNumOps: 0
  - RpcQueueTimeAvgTime: 0
  - RpcClockWaitTimeNumOps: 0

# 5. WEB ADMINISTRACION YARN

Paso 11. Por último si nos vamos a **cluster/Nodes** podemos entrar a ver la información general de cada nodo concreto haciendo click en su enlace. Ya no estamos en el puerto 8088 sino en el 8042.

Node Labels	Rack	Node State	Node Address	Node HTTP Address
/default-rack		RUNNING	nod01:43807	<a href="#">nod01:8042</a>

La información del NodeManager que aparece es virtual (memory, process) y no se corresponde con la máquina real que tenemos por debajo.

The screenshot shows a web browser window displaying the NodeManager information for a specific node. The URL in the address bar is `nod01:8042/node`. The page features a logo for "hadoop" and a sidebar with navigation links like "ResourceManager", "NodeManager", "Node Information", "List of Applications", "List of Containers", and "Tools". The main content area is titled "NodeManager information" and contains several data tables with system statistics:

Total Vmem allocated for Containers	16.80 GB
Vmem enforcement enabled	true
Total Pmem allocated for Container	8 GB
Pmem enforcement enabled	true
Total VCores allocated for Containers	8
NodeHealthyStatus	true
LastNodeHealthTime	Sun Feb 12 15:03:15 UTC 2023
NodeHealthReport	
NodeManager started on	Sun Feb 12 13:01:09 UTC 2023
NodeManager Version:	2.10.2 from 965fd380006fa78b2315668fbc7eb432e1d8200f by ubuntu source checksum 332d49624f
Hadoop Version:	2.10.2 from 965fd380006fa78b2315668fbc7eb432e1d8200f by ubuntu source checksum d3ab737f77

# 5. WEB ADMINISTRACION YARN

Paso 12. En la opción **List of Applications** podemos ver las aplicaciones que están lanzándose aquí o que le han tocado. En la opción **List of Containers** podemos ver los containers. Por cada aplicación se abren los contenedores que son los recursos reales que se utilizan para hacer los procesos. Como vemos no hay nada

The image displays two screenshots of the Hadoop YARN NodeManager web interface, showing the 'List of Applications' and 'List of Containers' pages.

**Screenshot 1: List of Applications**

- URL:** nodo1:8042/node/allApplications
- Title:** Applications running on this node
- Content:** A table with columns ApplicationId and ApplicationState. The message "No data available in table" is displayed.
- Left Sidebar:** ResourceManager, NodeManager (selected), Node Information, List of Applications, List of Containers, Tools.

**Screenshot 2: List of Containers**

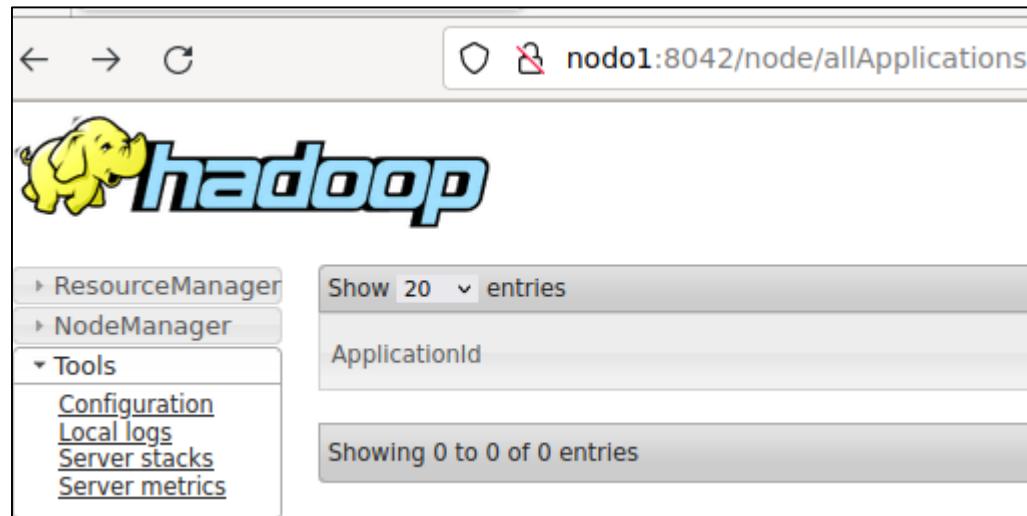
- URL:** nodo1:8042/node/allContainers
- Title:** All containers running on this node
- Content:** A table with columns ContainerId and ContainerState. The message "No data available in table" is displayed.
- Left Sidebar:** ResourceManager, RM Home (selected), NodeManager, Tools.

# 5. WEB ADMINISTRACION YARN

---

Paso 13. En Tools también podemos ver lo mismo

- el fichero de configuración,
- los logs locales
- las estadísticas del servidor.



# 5. WEB ADMINISTRACION YARN

Paso 14. Si queremos volver de nuevo a la ventana principal donde estaba trabajando con el clúster, hacemos click en el ResourceManager, y después en RM Home, volvemos de nuevo al cluster nodo1:8088 o localhost:8088

The screenshot shows the Hadoop ResourceManager web interface. At the top, there are two browser tabs: one for 'nodo1:8042/node/allApplications' and another for 'localhost:8088/cluster'. The main content area displays the 'All Applications' page. On the left, a sidebar menu includes 'ResourceManager' (with 'RM Home' selected), 'NodeManager', and 'Tools'. Below the sidebar is a message 'Showing 0 to 0 of 0 entries'. The main content area features a large 'hadoop' logo at the top. It includes sections for 'Cluster Metrics' (showing 0 for all metrics like Apps Submitted, Apps Pending, etc.) and 'Cluster Nodes Metrics' (showing 1 active node). A 'Scheduler Metrics' section indicates the use of 'Capacity Scheduler'. The bottom part of the page shows a table header for 'All Applications' with columns for ID, User, Name, Application Type, Queue, Application Priority, Start Time, Launch Time, Finish Time, State, Final Status, Running Containers, Allocated CPU, and Allocated Memory. A note at the bottom states 'No data available in table'. The URL 'localhost:8088/cluster' is also visible in the browser's address bar.

## 5. WEB ADMINISTRACION YARN

---

Paso 15. En la versión 3 de Hadoop, a veces puede generar un error al realizar alguno de los ejemplos. Para solucionarlo, es necesario añadir algunas librerías en yarn-site.xml:

```
<property>
<name>yarn.application.classpath</name>
  <value>
    /opt/hadoop3/hadoop/etc/hadoop,
    /opt/hadoop3/share/hadoop/common/*,
    /opt/hadoop3/share/hadoop/common/lib/*,
    /opt/hadoop3/share/hadoop/hdfs/*,
    /opt/hadoop3/share/hadoop/hdfs/lib/*,
    /opt/hadoop3/share/hadoop/mapreduce/*,
    /opt/hadoop3/share/hadoop/mapreduce/lib/*,
    /opt/hadoop3/share/hadoop/yarn/*,
    /opt/hadoop3/share/hadoop/yarn/lib/*
  </value>
</property>
```

---

## 5. WEB ADMINISTRACION YARN

---

Paso 16. Para apagar hadoop se hace en orden inverso, primero se apaga yarn y después hdfs.

```
hadoop@nodo1:/opt/hadoop/etc/hadoop$ stop-yarn.sh
stopping yarn daemons
stopping resourcemanager
localhost: no nodemanager to stop
no proxyserver to stop
hadoop@nodo1:/opt/hadoop/etc/hadoop$ stop-dfs.sh
Stopping namenodes on [nodo1]
nodo1: stopping namenode
localhost: stopping datanode
Stopping secondary namenodes [0.0.0.0]
0.0.0.0: stopping secondarynamenode
hadoop@nodo1:/opt/hadoop/etc/hadoop$
```

# 6. FUNCIONAMIENTO DE MAPREDUCE

---

- Haremos una pequeña descripción del funcionamiento MapReduce.
  - Aunque Yarn le ha cogido terreno en algunas cosas de la gestión del clúster, sigue siendo uno de los procesos más habituales de BigData
  - La mayor parte de las herramientas de consulta están diseñadas para realizar consultas simples que deban ejecutarse rápidamente.
  - El dato suele estar indexado y por tanto solo pequeñas porciones de datos se examinan durante la búsqueda.
  - Esta solución en cambio no es útil para datos que no pueden ser indexados de tipo semi estructurado (textos) o sin estructurar (multimedia)
  - Para responder una query en esta solución es necesario examinar todos los datos
  - Hadoop utiliza MapReduce para realizar un análisis exhaustivo que permita hacer estos procesos de búsqueda en cargas masivas de manera rápida.
-

# 6. FUNCIONAMIENTO DE MAPREDUCE

---

- MapReduce es un algoritmo de datos de tipo batch que va implementando procesos en paralelo.
- La filosofía es divide y vencerás: en vez de ejecutar un proceso de manera secuencial, se trocea y se manda a cada nodo y eso hace que el rendimiento sea mucho mayor, al ejecutarse en paralelo
- MapReduce de forma simple distribuye las tareas a través de los nodos de un cluster ejecutando una función "map". Esta función estudia el problema, divide cada proceso en múltiples trozos y los manda a diferentes maquinas a ejecutarse en paralelo contra cada uno de los bloques de 128MB de datos.
- Si tenemos 50 trozos va a ejecutarse 50 veces el mismo proceso en paralelo pero con datos distintos.
- Los resultados de este proceso paralelo se recogen y se distribuyen a través de distintos servidores que ejecutan una función "reduce", que toma los resultados de los trozos y los recombina para obtener una respuesta simple

# 6. FUNCIONAMIENTO DE MAPREDUCE

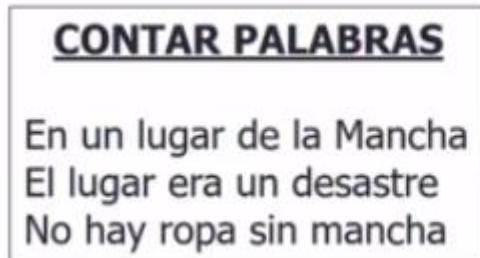
---

Vamos a tener en realidad tres pasos/componentes:

- **Procesos Mapper:**
  - Los programas Map Reduce se dividen en Mappers, tareas que se ejecutan en los nodos contra los bloques de datos
  - Cada tarea MAP ataca a un solo bloque de datos HDFS
  - Siempre se ejecutan en el nodo donde reside el bloque
- **Shuffle y Sort**
  - Ordena y consolida los datos intermedios (temporales) que han generado los mapper
  - Se lanzan después de que todos los mappers hayan terminado y antes de que se lancen los procesos Reducer
- **Reducer**
  - Operan sobre los datos intermedios que se han generado en el paso anterior. Son los que generan la salida final .

# 6. FUNCIONAMIENTO DE MAPREDUCE

- Imaginamos que queremos contar palabras. Puede parecer sencillo pero si tenemos millones de datos ya no es tan sencillo
- Esto sería un proceso MapReduce. Cogemos un fichero que tiene unas cuantas frases y cuento las palabras para ver cuántas veces aparece cada palabra.



Resultado final

## RESULTADOS

De	1
El	1
En	1
Era	1
Hay	1
La	1
Lugar	2
Mancha	2
No	1
Ropa	1
Sin	1
Un	2

# 6. FUNCIONAMIENTO DE MAPREDUCE

## a) Procesos Mapper

- Vamos a suponer que cada una de estas frases es un bloque. Imaginamos que en vez de ser 3 líneas son 3 bloques de 128 megas. Hay 3 Mappers donde cada mapper coge un trozo de ese fichero y empieza a contar las ocurrencias que van apareciendo y las va etiquetando

**CONTAR PALABRAS**

En un lugar de la Mancha  
El lugar era un desastre  
No hay ropa sin mancha

The diagram illustrates the Map phase of a MapReduce process. On the left, a box contains the input text: "En un lugar de la Mancha", "El lugar era un desastre", and "No hay ropa sin mancha". Three arrows labeled "MAP" point from this input to three separate tables on the right, representing the output of three different mappers.

En	1
Un	1
Lugar	1
De	1
La	1
mancha	1

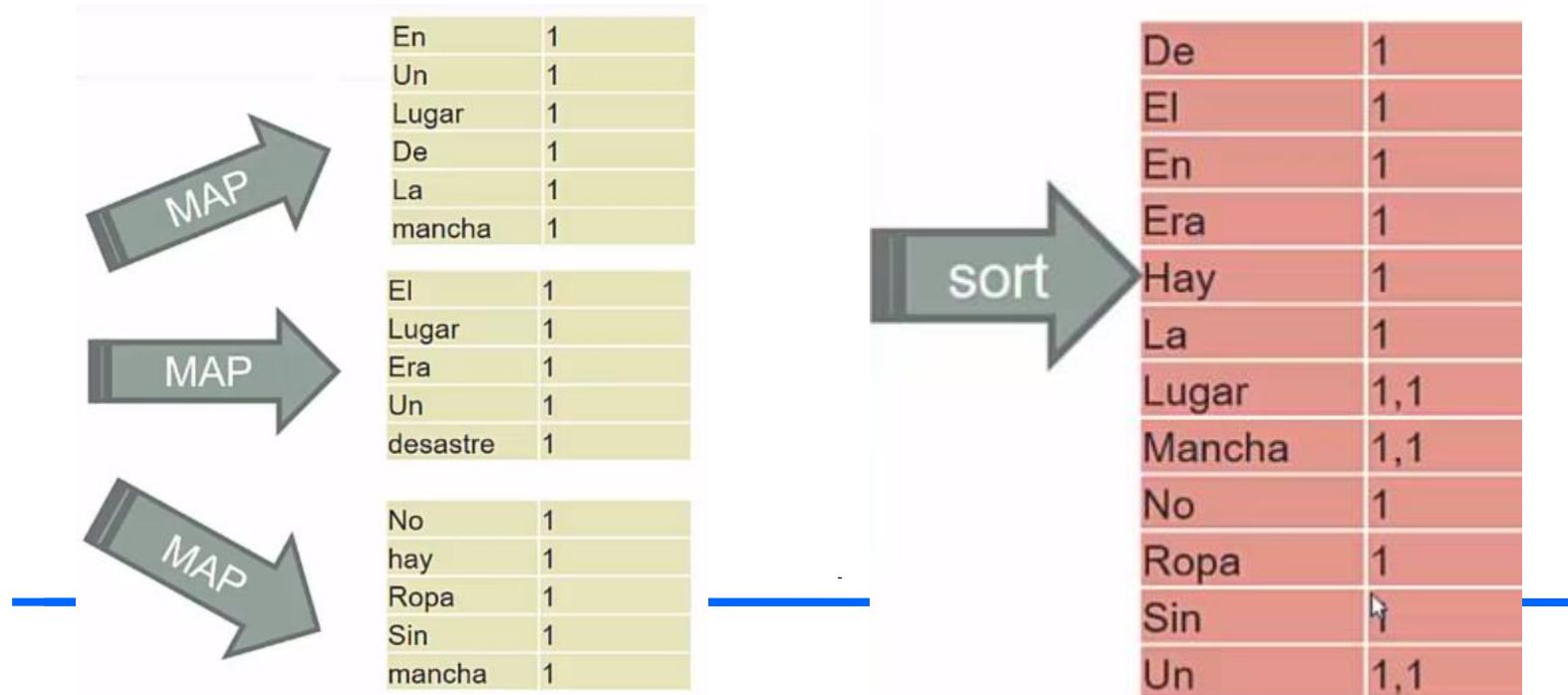
El	1
Lugar	1
Era	1
Un	1
desastre	1

No	1
hay	1
Ropa	1
Sin	1
mancha	1

# 6. FUNCIONAMIENTO DE MAPREDUCE

## b) Shuffle y sort

- En el siguiente paso Shuffle trabaja con clave-valor: Confecciona una lista donde la clave es cada palabra y los valores son el número de veces que aparece.
- En el proceso intermedio generamos una serie de claves con posibles valores que no tienen porque ser sólo 1

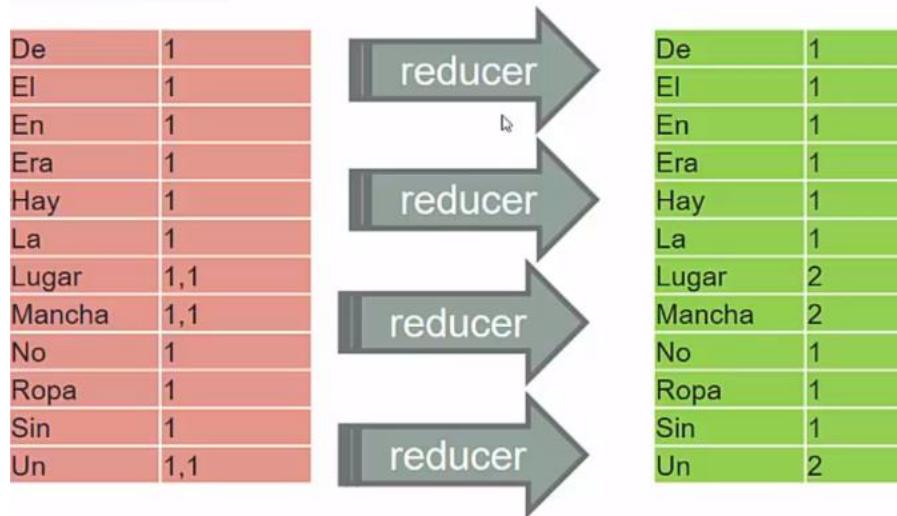


# 6. FUNCIONAMIENTO DE MAPREDUCE

---

## c) Procesos Reducers

En el proceso final los Reducers van a coger esto básicamente lo que van a hacer como veis es contar



Un proceso grande, lo hemos dividido en múltiples procesos paralelos. Hemos generado con sort y Shuffle una serie de datos intermedios y por último otros procesos de tipo Reducers que normalmente son menos que los Mappers nos han hecho el resultado final.

---

# 7. EJEMPLO CON MAPREDUCE

Paso 1. Contaremos las palabras que más aparecen en el libro **El Quijote** mediante un proceso Hadoop MapReduce. Descargaremos un fichero muy pequeño, el cual no generará muchos Mappers ni Reducers. Desde la pagina Proyecto Gutenberg <https://www.gutenberg.org> podemos descargar libremente El Quijote, como simple texto.

The screenshot shows a web browser displaying the Project Gutenberg website at [www.gutenberg.org/ebooks/2000](https://www.gutenberg.org/ebooks/2000). The page title is "Don Quijote by Miguel de Cervantes Saavedra". On the left, there is a thumbnail image of the book cover, which is teal with a purple circular logo and the text "Project Gutenberg". Below the cover are social sharing icons for Twitter, Facebook, and Print. To the right, there is a table titled "Download This eBook" with columns for "Format", "Size", and download links. The formats listed are: HTML5 (2.2 MB), HTML (as submitted) (2.2 MB), EPUB3 (E-readers incl. Send-to-Kindle) (877 kB), EPUB (no images) (917 kB), Kindle (1.5 MB), older Kindles (1.4 MB), Plain Text UTF-8 (2.1 MB), and More Files... (represented by a folder icon).

Format	Size			
<a href="#">Read this book online: HTML5</a>	2.2 MB			
<a href="#">Read this book online: HTML (as submitted)</a>	2.2 MB			
<a href="#">EPUB3 (E-readers incl. Send-to-Kindle)</a>	877 kB			
<a href="#">EPUB (no images)</a>	917 kB			
<a href="#">Kindle</a>	1.5 MB			
<a href="#">older Kindles</a>	1.4 MB			
<a href="#">Plain Text UTF-8</a>	2.1 MB			
<a href="#">More Files...</a>				

# 7. EJEMPLO CON MAPREDUCE

---

Paso 2. Movemos el fichero `quijote.txt` al directorio `/tmp`. Creamos el directorio `/libros` dentro de HDFS y subimos el fichero `quijote.txt` a este directorio

```
hadoop@nodo1:~/Downloads$ ls
hadoop-2.10.2.tar.gz  hadoop-3.3.4.tar.gz  quijote.txt
hadoop@nodo1:~/Downloads$ mv quijote.txt /tmp
hadoop@nodo1:~/Downloads$ hdfs dfs -mkdir /libros
hadoop@nodo1:~/Downloads$ hdfs dfs -put /tmp/quijote.txt /libros
hadoop@nodo1:~/Downloads$ hdfs dfs -ls /libros
Found 1 items
-rw-r--r--  1 hadoop supergroup  2226045 2023-02-18 21:24 /libros/quijote.txt
hadoop@nodo1:~/Downloads$ █
```

**OBJETIVO:** Haremos que nos genere en otro fichero el número de palabras que tiene y cuantas veces aparecen cada palabra.

# 7. EJEMPLO CON MAPREDUCE

---

Paso 3. Recordamos que en `/opt/hadoop/share/hadoop/mapreduce` tenemos una serie de librerías entre la que destaca `hadoop-mapreduce-examples`

```
hadoop@nodo1:~$ pwd
/home/hadoop
hadoop@nodo1:~$ cd /opt/hadoop/share/hadoop/mapreduce/
hadoop@nodo1:/opt/hadoop/share/hadoop/mapreduce$ ls
hadoop-mapreduce-client-app-2.10.2.jar
hadoop-mapreduce-client-common-2.10.2.jar
hadoop-mapreduce-client-core-2.10.2.jar
hadoop-mapreduce-client-hs-2.10.2.jar
hadoop-mapreduce-client-hs-plugins-2.10.2.jar
hadoop-mapreduce-client-jobclient-2.10.2.jar
hadoop-mapreduce-client-jobclient-2.10.2-tests.jar
hadoop-mapreduce-client-shuffle-2.10.2.jar
hadoop-mapreduce-examples-2.10.2.jar
jdiff
lib
lib-examples
sources
hadoop@nodo1:/opt/hadoop/share/hadoop/mapreduce$ █
```

# 7. EJEMPLO CON MAPREDUCE

---

Paso 4. Con el comando jar, podemos ver una lista de determinados comandos o opciones que hay dentro de este paquete. Utilizaremos el comando wordcount que permite contar palabras dentro de los ficheros.

```
hadoop@nodo1:/opt/hadoop/share/hadoop/mapreduce$ jar tf hadoop-mapreduce-examples-2.10.2.jar  
META-INF/  
org/apache/  
org/apache/hadoop/  
org/apache/hadoop/examples/  
org/apache/hadoop/examples/terasort/  
org/apache/hadoop/examples/dancing/  
org/apache/hadoop/examples/pi/  
org/apache/hadoop/examples/pi/math/  
org/apache/hadoop/examples/BaileyBorweinPlouffe$BbpInputFormat$1.class
```

```
org/apache/hadoop/examples/DBCountPageView.class  
org/apache/hadoop/examples/DBCountPageView$PageviewMapper.class  
org/apache/hadoop/examples/WordCount.class  
org/apache/hadoop/examples/AggregateWordCount$WordCountPlugInClass.class  
org/apache/hadoop/examples/SecondarySort.class  
META-INF/maven/  
META-INF/maven/org.apache.hadoop/  
META-INF/maven/org.apache.hadoop/hadoop-mapreduce-examples/  
META-INF/maven/org.apache.hadoop/hadoop-mapreduce-examples/pom.xml  
META-INF/maven/org.apache.hadoop/hadoop-mapreduce-examples/pom.properties  
hadoop@nodo1:/opt/hadoop/share/hadoop/mapreduce$
```

# 7. EJEMPLO CON MAPREDUCE

---

Paso 5. Ejecutaremos el programa hadoop-mapreduce **wordcount** que cogerá el contenido del directorio /libros y dejará en /salida\_libros el resultado de las palabras contadas.

- **hadoop jar hadoop-mapreduce-examples-2.10.2.jar wordcount /libros /salida\_libros** (salida\_libros no es necesario crearlo).

```
hadoop@nodo1:/opt/hadoop/share/hadoop/mapreduce$ hadoop jar hadoop-mapreduce-examples-2.10.2.jar wordcount
/libros /salida_libros
23/02/18 22:07:21 INFO client.RMProxy: Connecting to ResourceManager at localhost
23/02/18 22:07:50 INFO mapreduce.Job: map 0% reduce 0%
23/02/18 22:08:08 INFO mapreduce.Job: map 100% reduce 0%
23/02/18 22:08:20 INFO mapreduce.Job: map 100% reduce 100%
23/02/18 22:08:21 INFO mapreduce.Job: Job job_1676758030441_0001 completed successfully
23/02/18 22:08:21 INFO mapreduce.Job: Counters: 49
      File System Counters
          FILE: Number of bytes read=602460
          FILE: Number of bytes written=1625401
          FILE: Number of read operations=0
          FILE: Number of large read operations=0
          FILE: Number of write operations=0
          HDFS: Number of bytes read=2226146
          HDFS: Number of bytes written=446927
          HDFS: Number of read operations=6
          HDFS: Number of large read operations=0
          HDFS: Number of write operations=2
      Job Counters
          Launched map tasks=1
```

# 7. EJEMPLO CON MAPREDUCE

---

```
Map output bytes=3740951
Map output materialized bytes=602460
Input split bytes=101
Combine input records=389719
Combine output records=39791
Reduce input groups=39791
Reduce shuffle bytes=602460
Reduce input records=39791
Reduce output records=39791
Spilled Records=79582
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=452
CPU time spent (ms)=12080
Physical memory (bytes) snapshot=509894656
Virtual memory (bytes) snapshot=3772424192
Total committed heap usage (bytes)=304087040
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
    Bytes Read=2226045
File Output Format Counters
    Bytes Written=446927
hadoop@nodo1:/opt/hadoop/share/hadoop/mapreduce$
```

# 7. EJEMPLO CON MAPREDUCE

---

Paso 5. Observamos el resultado obtenido

- **hdfs dfs -ls /salida\_libros :**

```
hadoop@nodo1:/opt/hadoop/share/hadoop/mapreduce$ hdfs dfs -ls /salida_libros
Found 2 items
-rw-r--r-- 1 hadoop supergroup          0 2023-02-18 22:08 /salida_libros/_SUCCESS
-rw-r--r-- 1 hadoop supergroup  446927 2023-02-18 22:08 /salida_libros/part-r-00000
hadoop@nodo1:/opt/hadoop/share/hadoop/mapreduce$
```

- Se han generado dos ficheros:
  - \_SUCCESS
  - part-r-00000 → Debe de contener las palabras contadas.

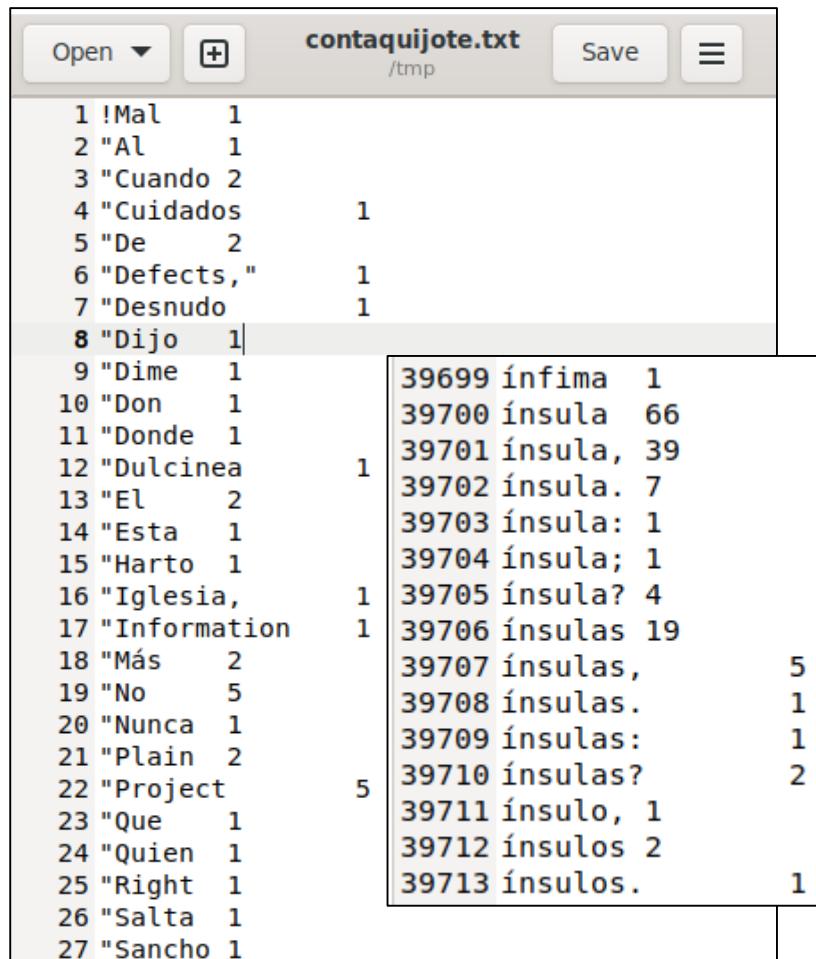
Paso 6. Descargamos el resultado a nuestra maquina local

- **hdfs dfs -get /salida\_libros/part-r-00000 /tmp/contaquiote.txt**

```
hadoop@nodo1:/opt/hadoop/share/hadoop/mapreduce$ hdfs dfs -get /salida_libros/part-r-00000
/tmp/contaquiote.txt
hadoop@nodo1:/opt/hadoop/share/hadoop/mapreduce$
```

# 7. EJEMPLO CON MAPREDUCE

Paso 7. Abrimos el fichero contaquijote.txt con gedit:  
**gedit/tmp/contaquijote.txt**



The screenshot shows a gedit window with the title "contaquijote.txt" and the path "/tmp". The text area displays the following wordcount output:

Palabra	Conteo
1 !Mal	1
2 "Al	1
3 "Cuando	2
4 "Cuidados	1
5 "De	2
6 "Defects,"	1
7 "Desnudo	1
8 "Dijo	1
9 "Dime	1
10 "Don	1
11 "Donde	1
12 "Dulcinea	
13 "El	2
14 "Esta	1
15 "Harto	1
16 "Iglesia,	
17 "Information	
18 "Más	2
19 "No	5
20 "Nunca	1
21 "Plain	2
22 "Project	
23 "Que	1
24 "Quien	1
25 "Right	1
26 "Salta	1
27 "Sancho	1

A secondary window is overlaid on the bottom right, showing a detailed wordcount for the word "ínsula".

Palabra	Conteo
39699 ínfima	1
39700 ínsula	66
39701 ínsula,	39
39702 ínsula.	7
39703 ínsula:	1
39704 ínsula;	1
39705 ínsula?	4
39706 ínsulas	19
39707 ínsulas,	5
39708 ínsulas.	1
39709 ínsulas:	1
39710 ínsulas?	2
39711 ínsulo,	1
39712 ínsulos	2
39713 ínsulos.	1

- Vemos que el programa wordcount no es muy inteligente: cuenta palabras, pero entiende que algo que empiece por doble comilla o finalice con punto o coma son palabras diferentes.
- El fichero tiene 40000 líneas, que en teoría son el nº de palabras diferentes, pero en realidad hay muchas palabras repetidas. Por ejemplo:
  - ínsula → aparece 66
  - ínsula, → aparece 39
- Wordcount aunque puede ser útil para algunos casos, para contar palabras dentro de un texto no es el más satisfactorio.

# 7. EJEMPLO CON MAPREDUCE

Paso 8. Si ejecutamos otra vez el programa wordcount:

- **hadoop jar hadoop-mapreduce-examples-2.10.2.jar wordcount /libros /salida\_libros → Encontramos un error**

```
hadoop@nodo1:/opt/hadoop/share/hadoop/mapreduce$ hadoop jar hadoop-mapreduce-examples-2.10.2.jar  
wordcount /libros /salida_libros  
23/02/19 11:13:55 INFO client.RMProxy: Connecting to ResourceManager at localhost/127.0.0.1:8032  
org.apache.hadoop.mapred.FileAlreadyExistsException: Output directory hdfs://nodo1:9000/salida_l  
ibros already exists  
    at org.apache.hadoop.mapreduce.lib.output.FileOutputFormat.checkOutputSpecs(FileOutputFo  
rmat.java:146)  
    at org.apache.hadoop.mapreduce.JobSubmitter.checkSpecs(JobSubmitter.java:279)  
    at org.apache.hadoop.mapreduce.JobSubmitter.submitJobInternal(JobSubmitter.java:145)
```

Debemos borrar el directorio de salida para ejecutarla otra vez:

```
255 hadoop@nodo1:/opt/hadoop/share/hadoop/mapreduce$ hdfs dfs -rm -R /salida_libros  
Deleted /salida_libros  
hadoop@nodo1:/opt/hadoop/share/hadoop/mapreduce$ hdfs dfs -ls /  
Found 5 items  
drwxr-xr-x  - hadoop supergroup          0 2023-02-05 20:49 /datos  
drwxr-xr-x  - hadoop supergroup          0 2023-02-18 21:24 /libros  
drwxr-xr-x  - hadoop supergroup          0 2023-02-05 14:59 /temporal  
drwxr-xr-x  - hadoop supergroup          0 2023-02-05 14:53 /temporal1  
drwx----- - hadoop supergroup          0 2023-02-18 22:03 /tmp  
hadoop@nodo1:/opt/hadoop/share/hadoop/mapreduce$ hadoop jar hadoop-mapreduce-examples-2.10.2.jar  
wordcount /libros /salida_libros  
23/02/19 11:43:11 INFO client.RMProxy: Connecting to ResourceManager at localhost/127.0.0.1:8032
```

# 7. EJEMPLO CON MAPREDUCE

Paso 9. Vamos a la pagina web de gestión nodo1:8088 o localhost:8088. Podemos ver todas las aplicaciones que se han lanzado (con su Application Ids), las veces que hemos lanzado el proceso (en nuestro caso 3 veces), que aplicaciones han terminado, cuáles han fallado, cuáles se están ejecutando, etc

The screenshot shows the Hadoop Cluster Management UI at the URL `localhost:8088/cluster`. The interface includes a sidebar with navigation links like 'About', 'Nodes', 'Labels', 'Applications' (with sub-options: NEW, NEW SAVING, SUBMITTED, ACCEPTED, RUNNING, FINISHED, FAILED, KILLED), and 'Scheduler'. The main content area is titled 'All Applications' and displays cluster metrics, node metrics, and scheduler metrics. Below these, a table lists three completed MapReduce applications. Each application row contains detailed information such as ID, User, Name, Application Type, Queue, Application Priority, StartTime, LaunchTime, FinishTime, State, FinalStatus, Running Containers, Allocated CPU, Allocated Memory, and Allocated GPUs.

ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU	Allocated Memory	Allocated GPUs
<a href="#">application_1676758030441_0003</a>	hadoop	word count	MAPREDUCE	default	0	Sun Feb 19 11:43:14 +0000 2023	Sun Feb 19 11:43:15 +0000 2023	Sun Feb 19 11:44:01 +0000 2023	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A
<a href="#">application_1676758030441_0002</a>	hadoop	word count	MAPREDUCE	default	0	Sun Feb 19 11:41:13 +0000 2023	Sun Feb 19 11:41:13 +0000 2023	Sun Feb 19 11:41:53 +0000 2023	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A
<a href="#">application_1676758030441_0001</a>	hadoop	word count	MAPREDUCE	default	0	Sat Feb 18 22:07:29 +0000 2023	Sat Feb 18 22:07:31 +0000 2023	Sat Feb 18 22:08:19 +0000 2023	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A

# 7. EJEMPLO CON MAPREDUCE

Paso 10. Si seleccionamos en Application\_ID vemos lo que ha hecho

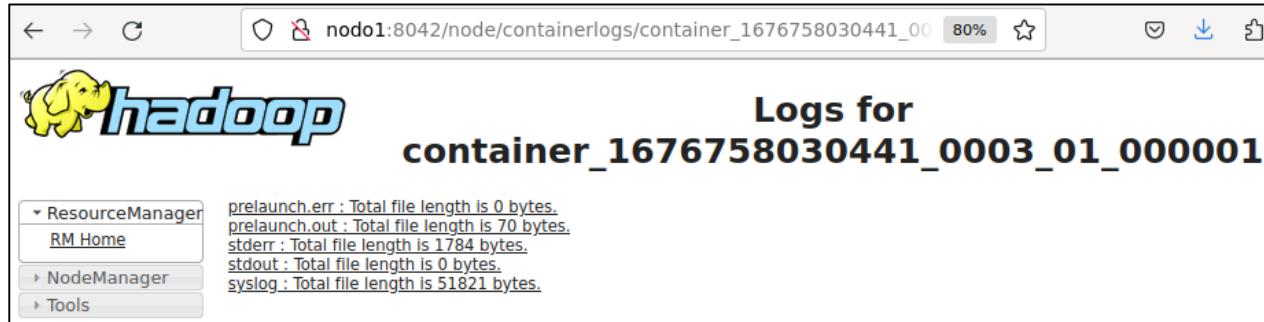
Application	
application_1676758030441_0003	
<a href="#">Application Overview</a>	
User:	hadoop
Name:	word count
Application Type:	MAPREDUCE
Application Tags:	
Application Priority:	0 (Higher Integer value indicates higher priority)
YarnApplicationState:	FINISHED
Queue:	default
FinalStatus Reported by AM:	SUCCEEDED
Started:	dom feb 19 11:43:14 +0000 2023
Launched:	dom feb 19 11:43:15 +0000 2023
Finished:	dom feb 19 11:44:01 +0000 2023
Elapsed:	46sec
Tracking URL:	History
Log Aggregation Status:	DISABLED
Application Timeout (Remaining Time):	Unlimited
Diagnostics:	
Unmanaged Application:	false
Application Node Label expression:	<Not set>
AM container Node Label expression:	<DEFAULT_PARTITION>

- Indica el programa que se ha lanzado: Word count
- Tipo Aplicación: MapReduce.
- Cuánto ha tardado: 46 segundos.
- Cuál ha sido el intento de generar la aplicación.
- El nodo dónde se ha lanzado.
- Los logs

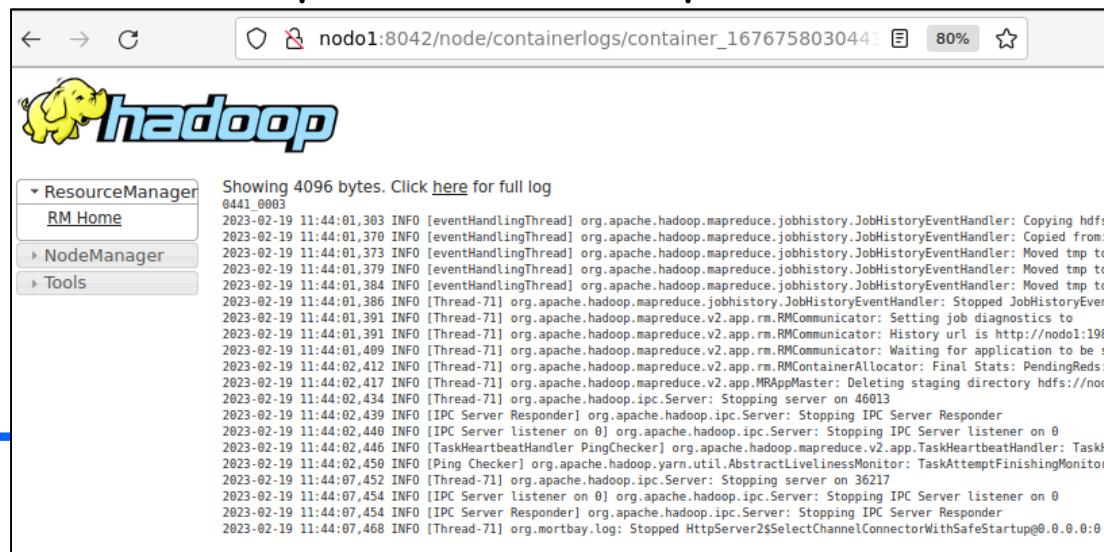
Show	20	entries
Attempt ID	Started	Node Logs
appattempt_1676758030441_0003_000001	Sun Feb 19 11:43:14 +0000 2023	<a href="http://nodo1:8042">http://nodo1:8042</a> <a href="#">Logs</a> 0
Showing 1 to 1 of 1 entries		

# 7. EJEMPLO CON MAPREDUCE

Paso 11. Si vamos a ver los logs para ver un poco cuál ha sido el proceso



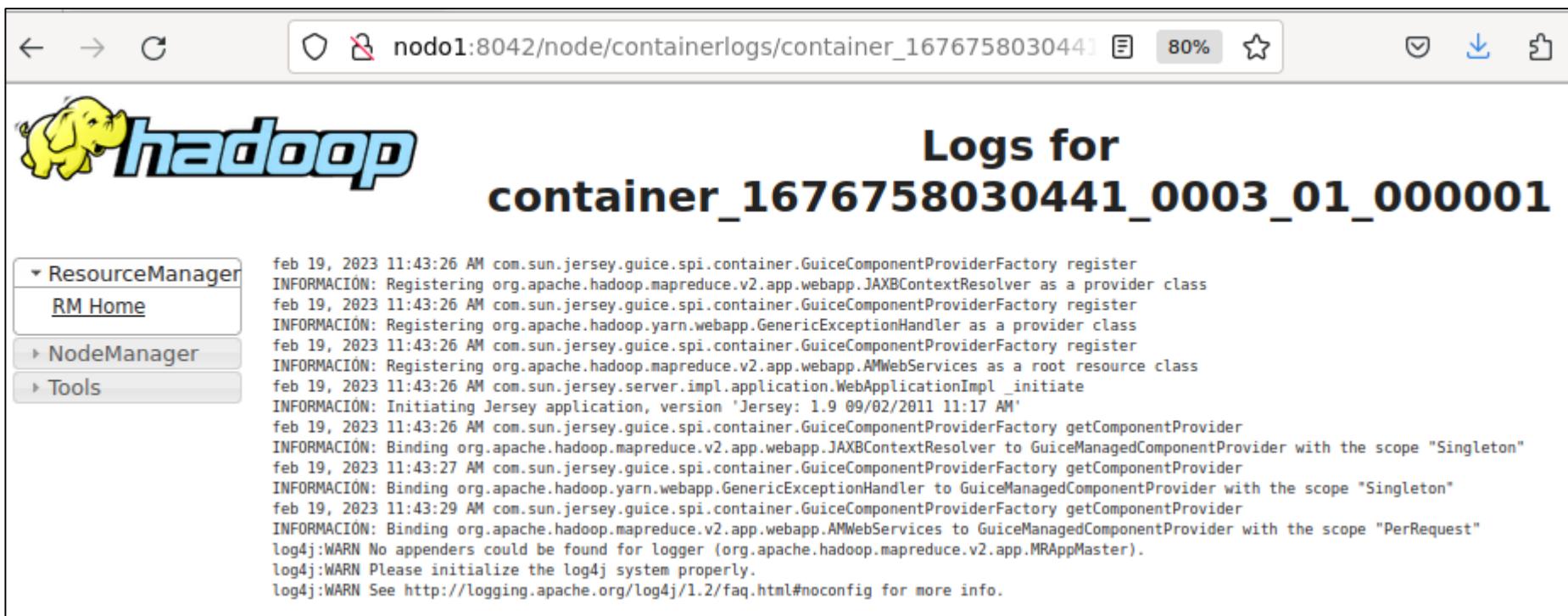
- Si vamos al syslog nos dice lo que ha ido haciendo. En este proceso ha estado haciendo procesos de MapReduce



# 7. EJEMPLO CON MAPREDUCE

---

Paso 12. Dentro de los logs de stderr. Si hubiéramos tenido algún error pues también nos lo hubiera dicho



The screenshot shows a web browser window displaying the Hadoop container logs. The URL in the address bar is `nodo1:8042/node/containerlogs/container_1676758030441`. The page title is "Logs for container\_1676758030441\_0003\_01\_000001". On the left, there is a sidebar with navigation links: "ResourceManager" (selected), "RM Home", "NodeManager", and "Tools". The main content area displays log entries from February 19, 2023, at 11:43:26 AM. The log entries are as follows:

```
feb 19, 2023 11:43:26 AM com.sun.jersey.spi.container.GuiceComponentProviderFactory register
INFORMACION: Registering org.apache.hadoop.mapreduce.v2.app.webapp.JAXBContextResolver as a provider class
feb 19, 2023 11:43:26 AM com.sun.jersey.spi.container.GuiceComponentProviderFactory register
INFORMACION: Registering org.apache.hadoop.yarn.webapp.GenericExceptionHandler as a provider class
feb 19, 2023 11:43:26 AM com.sun.jersey.spi.container.GuiceComponentProviderFactory register
INFORMACION: Registering org.apache.hadoop.mapreduce.v2.app.webapp.AMWebServices as a root resource class
feb 19, 2023 11:43:26 AM com.sun.jersey.server.impl.application.WebApplicationImpl _initiate
INFORMACION: Initiating Jersey application, version 'Jersey: 1.9 09/02/2011 11:17 AM'
feb 19, 2023 11:43:26 AM com.sun.jersey.spi.container.GuiceComponentProviderFactory getComponentProvider
INFORMACION: Binding org.apache.hadoop.mapreduce.v2.app.webapp.JAXBContextResolver to GuiceManagedComponentProvider with the scope "Singleton"
feb 19, 2023 11:43:27 AM com.sun.jersey.spi.container.GuiceComponentProviderFactory getComponentProvider
INFORMACION: Binding org.apache.hadoop.yarn.webapp.GenericExceptionHandler to GuiceManagedComponentProvider with the scope "Singleton"
feb 19, 2023 11:43:29 AM com.sun.jersey.spi.container.GuiceComponentProviderFactory getComponentProvider
INFORMACION: Binding org.apache.hadoop.mapreduce.v2.app.webapp.AMWebServices to GuiceManagedComponentProvider with the scope "PerRequest"
log4j:WARN No appenders could be found for logger (org.apache.hadoop.mapreduce.v2.app.MRAppMaster).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
```

# 7. EJEMPLO CON MAPREDUCE

Paso 13. En el intento, vemos el número de peticiones que se han hecho, el número de containers que se han lanzado

The screenshot shows a web browser window with the URL `localhost:8088/cluster/appattempt/appattempt_1676758030441_0003_000001`. The page title is "Application Attempt appattempt\_1676758030441\_0003\_000001". On the left, there's a sidebar with a yellow elephant logo and the word "hadoop". The sidebar menu includes "Cluster" (About, Nodes, Node Labels, Applications, NEW, NEW\_SAVING, SUBMITTED, ACCEPTED, RUNNING, FINISHED, FAILED, KILLED), "Scheduler", and "Tools". The main content area displays the "Application Attempt Overview" with the following details:

Application Attempt Overview	
Application Attempt State:	FINISHED
Started:	dom feb 19 11:43:14 +0000 2023
Elapsed:	46sec
AM Container:	container_1676758030441_0003_01_000001
Node:	nodo1:36217
Tracking URL:	<a href="#">History</a>
Diagnostics Info:	-
Nodes blacklisted by the application:	-
Nodes blacklisted by the system:	-

Total Allocated Containers: 3

Each table cell represents the number of NodeLocal/RackLocal/OffSwitch containers satisfied by NodeLocal/RackLocal/OffSwitch resource requests.

	Node Local Request	Rack Local Request	Off Switch Request
Num Node Local Containers (satisfied by)	1		
Num Rack Local Containers (satisfied by)	0	0	
Num Off Switch Containers (satisfied by)	0	0	2

Show 20 entries Search:

Container ID	Node	Container Exit Status	Logs
No data available in table			

Showing 0 to 0 of 0 entries First Previous Next Last

## 7. EJEMPLO CON MAPREDUCE

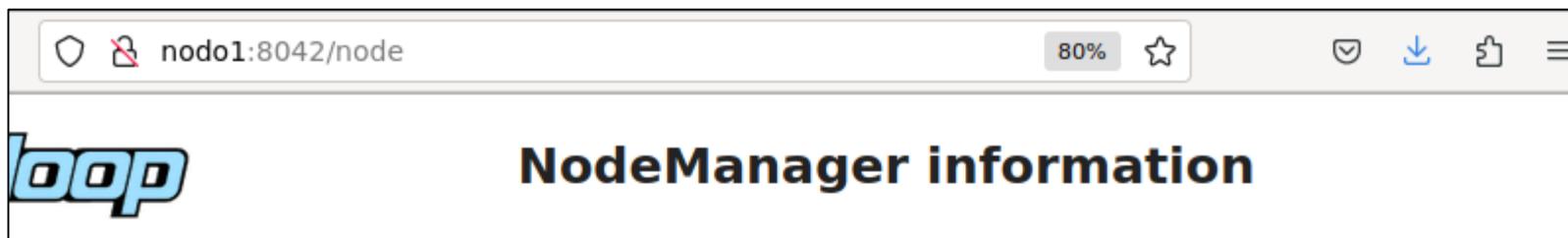
---

**Paso 14.** Si hacemos click en History nos da un error porque no tenemos el histórico arrancado. Nos permite ver comandos que se han lanzado hace tiempo..



# 7. EJEMPLO CON MAPREDUCE

Paso 15. Si hacemos click en <http://nodo1:8042>, podemos ver que se ha lanzado en este nodo. En **List of Applications** vemos las aplicaciones que se han lanzado en este nodo y cómo han terminado. Como ya han terminado no tenemos en este momento ningún otro tipo de información.



The screenshot shows the NodeManager information page for node1:8042/node. It displays various system statistics:

NodeManager information	
Total Vmem allocated for Containers	16.80 GB
Vmem enforcement enabled	true
Total Pmem allocated for Container	8 GB
Pmem enforcement enabled	true
Total VCores allocated for Containers	8
NodeHealthyStatus	true
LastNodeHealthTime	Sun Feb 19 19:41:03 UTC 2023
NodeHealthReport	
NodeManager started on	Sat Feb 18 22:06:58 UTC 2023
NodeManager Version:	2.10.2 from 965fd380006fa78b2332d49624f6d2fd6fb142861335
Hadoop Version:	2.10.2 from 965fd380006fa78bd3ab737f7788f05d467784f0a86


The screenshot shows the Applications running on this node page for node1:8042/node/allApplications. It lists two Hadoop applications:

ApplicationId	ApplicationState
application_1676758030441_0002	FINISHED
application_1676758030441_0003	FINISHED

Showing 1 to 2 of 2 entries

## 7. EJEMPLO CON MAPREDUCE

---

Paso 16. En **List of Containers** podemos ver la lista de contenedores. Vemos que ya no hay ninguno pero que en su momento los debió haber cuando se lanzó la aplicación.

The screenshot shows a web browser window with the following details:

- Address Bar:** nodo1:8042/node/allContainers
- Title Bar:** All containers running on this node
- Content Area:**
  - Loop Logo:** On the left side.
  - Table Headers:** ContainerId, ContainerState
  - Message:** No data available in table
  - Table Footer:** Showing 0 to 0 of 0 entries

# 8. MAPREDUCE DESDE JAVA

---

Paso 1. Compilaremos el código Java de un programa

ContarPalabras.java, lanzaremos su proceso contra el fichero access\_log, que tiene un tamaño elevado. Y por ultimo en la pagina de administración de Yarn veremos como ha ido lanzando los mapers y los reducers.

Primero descargamos el programa y lo copiamos en el directorio practicas, dentro de /home/hadoop

```
1 hadoop@nodo1:~$ cd Downloads/
hadoop@nodo1:~/Downloads$ pwd
/home/hadoop/Downloads
hadoop@nodo1:~/Downloads$ ls
ContarPalabras.java  hadoop-2.10.2.tar.gz  hadoop-3.3.4.tar.gz
hadoop@nodo1:~/Downloads$ cd
hadoop@nodo1:~$ mkdir practicas
hadoop@nodo1:~$ cp Do
Documents/ Downloads/
hadoop@nodo1:~$ cp Downloads/ContarPalabras.java practicas/
hadoop@nodo1:~$ cd practicas
hadoop@nodo1:~/practicas$ ls
ContarPalabras.java
hadoop@nodo1:~/practicas$
```

# 8. MAPREDUCE DESDE JAVA

---

Paso 2. En ContarPalabras.java tenemos :

- Un objeto Mapper que extiende la clase `org.apache.hadoop.mapreduce.Mapper`, que es la que va a contar las palabras y por cada una que encuentre suma un 1.
- Al ejecutar Hadoop, se recibe cada línea del fichero de entrada como input. La función map devuelve por cada palabra (word) un (word,1) como salida.

```
public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable> {

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context) throws IOException, InterruptedException {
        StringTokenizer itr = new StringTokenizer(value.toString());
        while (itr.hasMoreTokens()) {
            word.set(itr.nextToken());
            context.write(word, one);
        }
    }
}
```

# 8. MAPREDUCE DESDE JAVA

---

Paso 3. La función reduce recibe todos los valores que tienen la misma clave como entrada y devuelve la clave y el número de ocurrencias como salida

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

# 8. MAPREDUCE DESDE JAVA

---

Paso 4. La clase main que lo único que hace es recibir un fichero y generar una salida de acuerdo.

```
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
    if (otherArgs.length != 2) {
        System.err.println("Uso: ContarPalabras <in> <out>");
        System.exit(2);
    }
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(ContarPalabras.class);
    job.setMapperClass(TokenizerMapper.class);

    /*** Dejarlo tal cual ****/
    // job.setCombinerClass(IntSumReducer.class);

    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
    FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```

# 8. MAPREDUCE DESDE JAVA

---

Paso 5. Compilaremos este fichero a manualmente con Hadoop:

- a) update-alternatives -list java → donde esta el JDK de java
- b) export HADOOP\_CLASSPATH=/usr/lib/jvm/java-8-openjdk-amd64/lib/tools.jar → para que Hadoop sepa encontrar el JDK
- c) hadoop com.sun.tools.javac.Main ContarPalabras.java
- d) Deberia generar 3 clases uno para el main, otro para el Reducer y otro para el Mapper.

```
hadoop@nodo1:~/practicas$ update-alternatives --list java
/usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java
hadoop@nodo1:~/practicas$ export HADOOP_CLASSPATH=/usr/lib/jvm/java-8-openjdk-amd64/lib/tools.jar
hadoop@nodo1:~/practicas$ hadoop com.sun.tools.javac.Main ContarPalabras.java
hadoop@nodo1:~/practicas$ ls -l
total 16
-rw-rw-r-- 1 hadoop hadoop 1754 feb 20 08:34 'ContarPalabras$IntSumReducer.class'
-rw-rw-r-- 1 hadoop hadoop 1751 feb 20 08:34 'ContarPalabras$TokenizerMapper.class'
-rw-rw-r-- 1 hadoop hadoop 1846 feb 20 08:34 ContarPalabras.class
-rw-rw-r-- 1 hadoop hadoop 2934 feb 20 08:31 ContarPalabras.java
hadoop@nodo1:~/practicas$ █
```

# 8. MAPREDUCE DESDE JAVA

---

Paso 6. Hadoop solo admite ficheros JAR. Creamos un fichero jar que contenga los class generados

```
hadoop@nodo1:~/practicas$ jar cf ContarPalabras.jar Contar*.class
hadoop@nodo1:~/practicas$ ls -l
total 20
-rw-rw-r-- 1 hadoop hadoop 1754 feb 20 08:34 'ContarPalabras$IntSumReducer.class'
-rw-rw-r-- 1 hadoop hadoop 1751 feb 20 08:34 'ContarPalabras$TokenizerMapper.class'
-rw-rw-r-- 1 hadoop hadoop 1846 feb 20 08:34 ContarPalabras.class
-rw-rw-r-- 1 hadoop hadoop 3276 feb 20 09:08 ContarPalabras.jar
-rw-rw-r-- 1 hadoop hadoop 2934 feb 20 08:31 ContarPalabras.java
hadoop@nodo1:~/practicas$
```

# 8. MAPREDUCE DESDE JAVA

Paso 7. Antes de ejecutarlo debemos hacer dos pasos.

- Asegurarnos de que la página web desde la que se gestionan las aplicaciones y nodos, esta abierta para ver cómo se lanza
- Activar el proceso History Server, que nos permite crear un histórico con el cual luego poder buscar posibles problemas o todos los pasos que se han ido realizando a lo largo de la ejecución

The screenshot shows the Hadoop Cluster Management UI at `localhost:8088/cluster`. The main title is "All Applications". On the left, there's a sidebar with "Cluster Metrics" (About, Nodes, Node Labels, Applications), "Cluster Nodes Metrics" (Active Nodes, Decommissioning Nodes, Decommissioned Nodes, Lost Nodes), "Scheduler Metrics" (Scheduler Type, Capacity Scheduler), and "Tools". The main area displays a table of applications:

ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	All VCore
application_1676758030441_0003	hadoop	word count	MAPREDUCE	default	0	Sun Feb 19 11:43:14 +0000 2023	Sun Feb 19 11:43:15 +0000 2023	Sun Feb 19 11:44:01 +0000 2023	FINISHED	SUCCEEDED	N/A	N/A
application_1676758030441_0002	hadoop	word count	MAPREDUCE	default	0	Sun Feb 19 11:41:13 +0000 2023	Sun Feb 19 11:41:13 +0000 2023	Sun Feb 19 11:41:53 +0000 2023	FINISHED	SUCCEEDED	N/A	N/A
application_1676758030441_0001	hadoop	word count	MAPREDUCE	default	0	Sat Feb 18 22:07:29 +0000 2023	Sat Feb 18 22:07:31 +0000 2023	Sat Feb 18 22:08:19 +0000 2023	FINISHED	SUCCEEDED	N/A	N/A

# 8. MAPREDUCE DESDE JAVA

---

Paso 8. Para activar el history server usaremos el Shell- script mr-jobhistory-daemon.sh que se encuentra en el directorio /opt/hadoop/sbin

**mr-jobhistory-daemon.sh start historyserver**

De esta manera cuando lancemos una aplicación, pinchando en Application ID y en History tenemos un histórico de cuántos procesos Mapper ha ejecutado

```
hadoop@nodo1:~/practicas$ ls /opt/hadoop/sbin/
distribute-exclude.sh  mr-jobhistory-daemon.sh  start-secure-dns.sh  stop-secure-dns.sh
FederationStateStore   refresh-namenodes.sh    start-yarn.cmd      stop-yarn.cmd
hadoop-daemon.sh       slaves.sh              start-yarn.sh       stop-yarn.sh
hadoop-daemons.sh     start-all.cmd        stop-all.cmd       yarn-daemon.sh
hdfs-config.cmd        start-all.sh         stop-all.sh        yarn-daemons.sh
hdfs-config.sh         start-balancer.sh   stop-balancer.sh
httpfs.sh               start-dfs.cmd      stop-dfs.cmd
kms.sh                 start-dfs.sh       stop-dfs.sh
hadoop@nodo1:~/practicas$
```

# 8. MAPREDUCE DESDE JAVA

---

Paso 9. Lanzamos el proceso java:

**hadoop jar ContarPalabras.jar ContarPalabras  
/temporal/access\_log /salida\_java**

Vemos que se conecta con el ResourceManager. Y lanza un job que ejecutará el proceso. Empieza a hacer los procesos Mappers. Cuando termine con estos, hará los procesos Reduce

```
hadoop@nodo1:~/practicas$ hdfs dfs -ls /temporal
Found 1 items
-rw-r--r-- 1 hadoop supergroup 504941532 2023-02-05 13:50 /temporal/access_log
hadoop@nodo1:~/practicas$ hadoop jar ContarPalabras.jar ContarPalabras /temporal/access_log /salida_java
23/02/20 13:08:13 INFO client.RMProxy: Connecting to ResourceManager at localhost/127.0.0.1:8032
23/02/20 13:08:15 INFO input.FileInputFormat: Total input files to process : 1
23/02/20 13:08:15 INFO mapreduce.JobSubmitter: number of splits:4
23/02/20 13:08:16 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1676758030441_0004
23/02/20 13:08:17 INFO conf.Configuration: resource-types.xml not found
23/02/20 13:08:17 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
23/02/20 13:08:17 INFO resource.ResourceUtils: Adding resource type - name = memory-mb, units = Mi, type = COUNTABLE
23/02/20 13:08:17 INFO resource.ResourceUtils: Adding resource type - name = vcores, units = , type = COUNTABLE
23/02/20 13:08:17 INFO impl.YarnClientImpl: Submitted application application_1676758030441_0004
23/02/20 13:08:17 INFO mapreduce.Job: The url to track the job: http://nodo1:8088/proxy/application_1676758030441_0004/
23/02/20 13:08:17 INFO mapreduce.Job: Running job: job_1676758030441_0004
23/02/20 13:08:35 INFO mapreduce.Job: Job job_1676758030441_0004 running in uber mode : false
23/02/20 13:08:35 INFO mapreduce.Job: map 0% reduce 0%
23/02/20 13:09:51 INFO mapreduce.Job: map 8% reduce 0%
23/02/20 13:09:57 INFO mapreduce.Job: map 17% reduce 0%
23/02/20 13:09:58 INFO mapreduce.Job: map 20% reduce 0%
23/02/20 13:10:30 INFO mapreduce.Job: map 24% reduce 0%
23/02/20 13:10:36 INFO mapreduce.Job: map 34% reduce 0%
```

# 8. MAPREDUCE DESDE JAVA

Paso 10. Si ahora refrescamos la pagina web y podemos comprobar que se ha lanzado el programa java ContarPalabras que tiene la etiqueta Word count. Como los procesos empiezan a ser bastante pesados (se lanzan varios mappers de 1GB cada uno), el refresco de la web puede costar. Teníamos que haber puesto un poquito más de recursos.

Cluster Nodes Metrics								
Active Nodes	Decommissioning Nodes	Decommissioned Nodes						
1	0	0						
Scheduler Metrics								
Scheduling Resource Type								
Capacity Scheduler [ <code>&lt;name=memory-mb default-unit=Mi type=COUNTABLE&gt;, &lt;name=vcores default-unit= Mi type=COUN</code>								
Show 20 entries								
ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime
<a href="#">application_1676758030441_0004</a>	hadoop	word count	MAPREDUCE	default	0	Mon Feb 20 13:08:17 +0000 2023	Mon Feb 20 13:08:18 +0000 2023	Mon Feb 20 13:19:52 +0000 2023
<a href="#">application_1676758030441_0003</a>	hadoop	word count	MAPREDUCE	default	0	Sun Feb 19 11:43:14 +0000 2023	Sun Feb 19 11:43:15 +0000 2023	Sun Feb 19 11:44:01 +0000 2023

# 8. MAPREDUCE DESDE JAVA

Paso 11. Si tuviéramos recursos suficientes podríamos ver en la web que la aplicación esta RUNNING. Si hacemos click en Application ID se ve que ha arrancado una ApplicationMaster, es el proceso general con el que trabajaba MapReduce

Con Yarn teníamos el ResourceManager el NodeManager y el ApplicationMaster que se arrancaba por cada aplicación que se lanzaba

The screenshot shows the Hadoop YARN ResourceManager UI. On the left, the 'All Applications' page displays cluster and scheduler metrics. On the right, a detailed view of a specific application is shown.

**All Applications**

**Cluster Metrics**

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	De
5	0	1	4	5	6 GB	8 GB	0 B	5	8	0	1	0

**Scheduler Metrics**

Scheduler Type	Scheduling Resource Type
Capacity Scheduler	[MEMORY] <memory:1

Show 20 entries

ID	User	Name	Application Type	Queue	StartTime
application_1459101853983_0005	hadoop	word count	MAPREDUCE	default	Mon Mar 28 00:10:38 +0200

**Application application\_1459101853983\_0005**

<b>User:</b>	hadoop
<b>Name:</b>	word count
<b>Application Type:</b>	MAPREDUCE
<b>Application Tags:</b>	
<b>YarnApplicationState:</b>	RUNNING: AM has registered with RM and started running.
<b>FinalStatus Reported by AM:</b>	Application has not completed yet.
<b>Started:</b>	lun mar 28 00:10:38 +0200 2016
<b>Elapsed:</b>	58sec
<b>Tracking URL:</b>	<a href="#">ApplicationMaster</a>
<b>Diagnostics:</b>	

# 8. MAPREDUCE DESDE JAVA

---

## Paso 12. Esperamos que termine.

```
23/02/20 13:13:51 INFO mapreduce.Job:  map 76% reduce 0%
23/02/20 13:13:56 INFO mapreduce.Job:  map 77% reduce 0%
23/02/20 13:13:59 INFO mapreduce.Job:  map 78% reduce 0%
23/02/20 13:14:08 INFO mapreduce.Job:  map 79% reduce 0%
    Combine output records=0
    Reduce input groups=2475645
    Reduce shuffle bytes=773627690
    Reduce input records=44778500
    Reduce output records=2475645

Spilled Records=134335500
Shuffled Maps =4
Failed Shuffles=0
Merged Map outputs=4
GC time elapsed (ms)=9893
CPU time spent (ms)=375970
Physical memory (bytes) snapshot=1289498624
Virtual memory (bytes) snapshot=9408188416
Total committed heap usage (bytes)=901251072

Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0

File Input Format Counters
    Bytes Read=504953820
File Output Format Counters
    Bytes Written=57779668

[1]+  Terminated                  firefox
hadoop@nodo1:~/practicas$
```

Primero hace los Mappers y después los Reduce. Al final obtenemos un resumen:

- n° de Jobs que ha hecho
- n° bytes leidos de los ficheros
- 4 mappers lanzados y 1 Reducer

Ha lanzado 4 procesos que se corresponden con el n° de bloques que tiene nuestro fichero

# 9. HISTORY

Paso 1. En nodo1:8088 observamos todas las aplicaciones lanzadas en Hadoop. Hacemos click en la Application ID que queremos comprobar. En Tracking URL tenemos el historial de los procesos que se han lanzado de esta aplicación. Si no hubiéramos puesto lo del History Server al pinchar aquí no saldría nada, y no podría ver los procesos que se han hecho

The screenshot shows the Hadoop Web UI interface. On the left, the 'All Applications' page displays cluster metrics like 'Apps Submitted' (4), 'Containers Running' (0), and a table of active nodes. On the right, a detailed view for application `application_1676758030441_0004` is shown. This view includes the application's configuration (User: hadoop, Name: word count, Application Type: MAPREDUCE), its history (Started: 2023-02-20 13:08:17, Finished: 2023-02-20 13:19:52), and its tracking URL (History). The bottom section shows application metrics like total resource preempted and aggregate resource allocation.

**All Applications**

Cluster Metrics	Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Used Resources	Total
	4	0	0	4	0		

**Application application\_1676758030441\_0004**

Cluster	User:	hadoop
About	Name:	word count
Nodes	Application Type:	MAPREDUCE
Node Labels	Application Tags:	
Applications	Application Priority:	0 (Higher Integer value indicates higher priority)
NEW	YarnApplicationState:	FINISHED
NEW SAVING	Queue:	default
SUBMITTED	FinalStatus Reported by AM:	SUCCEEDED
ACCEPTED	Started:	2023-02-20 13:08:17 +0000 2023
RUNNING	Launched:	2023-02-20 13:08:18 +0000 2023
FINISHED	Finished:	2023-02-20 13:19:52 +0000 2023
FAILED	Elapsed:	11mins, 34sec
KILLED	Tracking URL:	History
Scheduler	Log Aggregation Status:	DISABLED
Tools	Application Timeout (Remaining Time):	Unlimited
	Diagnostics:	
	Unmanaged Application:	false
	Application Node Label expression:	<Not set>
	AM container Node Label expression:	<DEFAULT_PARTITION>

**Application Metrics**

Total Resource Preempted:	<memory:0, vCores:0>
Total Number of Non-AM Containers Preempted:	0
Total Number of AM Containers Preempted:	0
Resource Preempted from Current Attempt:	<memory:0, vCores:0>
Number of Non-AM Containers Preempted from Current Attempt:	0
Aggregate Resource Allocation:	3646357 MB-seconds, 2846 vcore-seconds
Aggregate Preempted Resource Allocation:	0 MB-seconds, 0 vcore-seconds

# 9. HISTORY

Paso 2. En la pagina de History, podemos ver los contadores realizados, los mappers y los reducers que se han hecho, etc

En Overview vemos que el resultado wordcount ha sido satisfactorio. El proceso ha utilizado 4 procesos mappers y un procesos reducer. También nos dice si hubiera alguno fallido o eliminado.

The screenshot shows the Hadoop Job History interface. The main title is "MapReduce Job job\_1676758030441\_0004". The left sidebar has sections for Application, Job (with sub-options: Overview, Counters, Configuration, Map tasks, Reduce tasks), and Tools. The right panel is titled "Job Overview" and displays the following details:

Job Name:	word count
User Name:	hadoop
Queue:	default
State:	SUCCEEDED
Uberized:	false
Submitted:	Mon Feb 20 13:08:17 UTC 2023
Started:	Mon Feb 20 13:08:33 UTC 2023
Finished:	Mon Feb 20 13:19:46 UTC 2023
Elapsed:	11mins, 12sec
Diagnostics:	
Average Map Time	7mins, 19sec
Average Shuffle Time	1mins, 42sec
Average Merge Time	1sec
Average Reduce Time	1mins, 4sec

Below this is the "ApplicationMaster" table:

Attempt Number	Start Time	Node	Logs
1	Mon Feb 20 13:08:20 UTC 2023	node1:8042	logs

Finally, there is a "Task Type" table:

Task Type	Total	Complete
Map	4	4
Reduce	1	1

At the bottom, there is another table for "Attempt Type":

Attempt Type	Failed	Killed	Successful
Maps	0	0	4

# 9. HISTORY

Paso 3. Si hacemos click en los satisfactorios (en el link del 4). Podemos ver los 4 map que se han hecho con sort y además en los nodos donde se ha ejecutado

doop SUCCESSFUL MAP attempts in job\_1676758030441\_0004

Logged in as: dr.wh

Search:									
Attempt	State	Status	Node	Logs	Start Time	Finish Time	Elapsed Time	Note	
attempt_1676758030441_0004_m_000000_0	SUCCEEDED	map > /default-rack/nodo1:8042	logs	Mon Feb 20 13:08:36 +0000 2023	Mon Feb 20 13:16:06 +0000 2023	7mins, 30sec			
attempt_1676758030441_0004_m_000001_0	SUCCEEDED	map > /default-rack/nodo1:8042	logs	Mon Feb 20 13:08:36 +0000 2023	Mon Feb 20 13:16:06 +0000 2023	7mins, 30sec			
attempt_1676758030441_0004_m_000002_0	SUCCEEDED	map > /default-rack/nodo1:8042	logs	Mon Feb 20 13:08:36 +0000 2023	Mon Feb 20 13:16:06 +0000 2023	7mins, 30sec			
attempt_1676758030441_0004_m_000003_0	SUCCEEDED	map > /default-rack/nodo1:8042	logs	Mon Feb 20 13:08:36 +0000 2023	Mon Feb 20 13:15:23 +0000 2023	6mins, 47sec			
Attempt	State	Status	Node	Logs	Start Time	Finish Time	Elapsed	Note	
Showing 1 to 4 of 4 entries									
First Previous 1 Next Last									

# 9. HISTORY

Paso 4. Si vamos al menú Job, en Overview nos indica el resumen de lo que se ha hecho. En counters podemos ver los contadores o todos los procesos que se han lanzado en la fase de map y en la fase Reduce

The screenshot displays two browser windows for a MapReduce job.

**Left Window: Counters for job\_1676758030441\_0004**

- Left Sidebar:** Application, Job (Overview, Counters, Configuration, Map tasks, Reduce tasks), Tools.
- Content:** Counter Group table with two rows:
  - File System Counters:** FILE: Number of bytes read (Map: 773.627.780, Reduce: 773.627.690, Total: 1.547.255.470); FILE: Number of bytes written (Map: 1.548.095.582, Reduce: 773.837.663, Total: 2.321.933.245).
  - Job Counters:** HDFS: Number of bytes read, HDFS: Number of bytes written, HDFS: Number of large read operations, HDFS: Number of read operations, HDFS: Number of write operations.

**Right Window: MapReduce Job job\_1676758030441\_0004**

- Left Sidebar:** Application, Job (Overview, Counters, Configuration, Map tasks, Reduce tasks), Tools.
- Content:** Job Overview table with job details:

Job Name:	word count
User Name:	hadoop
Queue:	default
State:	SUCCEEDED
Uberized:	false
Submitted:	Mon Feb 20 13:08:17 UTC 2023
Started:	Mon Feb 20 13:08:33 UTC 2023
Finished:	Mon Feb 20 13:19:46 UTC 2023
Elapsed:	11mins, 12sec
Diagnostics:	
Average Map Time:	7mins, 19sec
Average Shuffle Time:	1mins, 42sec
Average Merge Time:	1sec
Average Reduce Time:	1mins, 4sec

# 9. HISTORY

Paso 5. En Configuration vemos la configuración que se ha utilizado, las tareas Maps que se han utilizado y las tareas Reducers Si se hace click en una tarea ya sea de tipo mapa o de tipo Radius podéis averiguar en qué nodo se han ejecutado

The image displays three screenshots of the Hadoop Job History UI, showing the configuration, map tasks, and reduce tasks for job\_1676758030441\_0004.

- Configuration for MapReduce Job job\_1676758030441\_0004**: Shows the configuration properties for the job, such as adl.feature.ownerandgroup.enableupn, datanode.https.port, dfs.balancer.address, dfs.balancer.block-move.timeout, dfs.balancer.dispatcherThreads, dfs.balancer.getBlocks.min-block-size, dfs.balancer.getBlocks.size, and dfs.balancer.keytab.enabled.
- Map Tasks for job\_1676758030441\_0004**: Shows the map tasks for the job, with four tasks listed as SUCCEEDED. Each task has a start time of Mon Feb 20 13:08:36 +0000 2023, a finish time of Mon Feb 20 13:16:06 +0000 2023, and an elapsed time of 7mins, 30sec. The successful attempt for each task is also listed.
- Reduce Tasks for job\_1676758030441\_0004**: Shows the reduce tasks for the job, with one task listed as SUCCEEDED. The task has a start time of Mon Feb 20 13:16:54 +0000 2023, a finish time of Mon Feb 20 13:18:36 +0000 2023, and an elapsed time of 2mins, 48sec. The successful attempt for the task is also listed.

# 9. HISTORY

Paso 6. Si vamos del nodo1:8088 al nodo1:50070, entramos en la web de HDFS y dentro de Utilities/Browse the file system, podemos ver el directorio de salida `salida_java`

The screenshot shows a web browser window with the URL `nodo1:50070/explorer.html#`. The page title is "Browse Directory". The main content area shows a table of file system entries:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	hadoop	supergroup	0 B	Feb 05 20:49	0	0 B	datos
drwxr-xr-x	hadoop	supergroup	0 B	Feb 18 21:24	0	0 B	libros
drwxr-xr-x	hadoop	supergroup	0 B	Feb 20 13:19	0	0 B	salida_java
drwxr-xr-x	hadoop	supergroup	0 B	Feb 19 11:43	0	0 B	salida_libros
drwxr-xr-x	hadoop	supergroup	0 B	Feb 05 14:59	0	0 B	temporal
drwxr-xr-x	hadoop	supergroup	0 B	Feb 05 14:53	0	0 B	temporal1
drwx-----	hadoop	supergroup	0 B	Feb 18 22:03	0	0 B	tmp

At the bottom, it says "Showing 1 to 7 of 7 entries".

# 9. HISTORY

Paso 7. En /salida\_java habrá dejado la información del número de veces que aparece cada palabra

The screenshot shows the Hadoop Web UI interface. The address bar indicates the URL is `nodo1:50070/explorer.html#/salida_java`. The page title is "Browse Directory". The URL input field contains `/salida_java`. Below it, there are filters: "Show 25 entries" and a search bar. The main area is a table listing two entries:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hadoop	supergroup	0 B	Feb 20 13:19	1	128 MB	_SUCCESS
-rw-r--r--	hadoop	supergroup	55.1 MB	Feb 20 13:19	1	128 MB	part-r-00000

At the bottom, it says "Showing 1 to 2 of 2 entries".

```
130 hadoop@nodo1:~/practicas$ hdfs dfs -ls /salida_java
Found 2 items
-rw-r--r-- 1 hadoop supergroup          0 2023-02-20 13:19 /salida_java/_SUCCESS
-rw-r--r-- 1 hadoop supergroup 57779668 2023-02-20 13:19 /salida_java/part-r-00000
hadoop@nodo1:~/practicas$ hdfs dfs -get /salida_java/part-r-00000 /tmp/salida_java.txt
hadoop@nodo1:~/practicas$
```