

2025



Big Data Aplicado

Profesor/a: José Manuel González Rodríguez

ACTIVIDAD EVALUABLE 2.1.- HADOOP: MAPREDUCE Y HDFS
15/03/2025

Índice

1.- Introducción	5
1.1.- Objetivo de la actividad	5
1.2.- Importancia de HDFS y MapReduce en el procesamiento de Big Data	6
1.2.1.- HDFS (Hadoop Distributed File System)	6
1.2.2.- MapReduce	7
1.2.3.- Relación entre HDFS y MapReduce	7
2.- Escenario de trabajo	8
2.1.- Acceso a la máquina virtual desde VirtualBox	8
2.2.- Requisitos y credenciales	9
2.3.- Comprobación y arranque del clúster	10
2.4.- Conexión a la interfaz web de Hadoop	12
3.- Comunicación entre entornos	13
3.1.- Comunicación entre máquina anfitriona y máquina virtual	13
3.2.- Copia de archivos usando SCP	14
3.3.- Comunicación entre sistema de ficheros local y HDFS	15
3.3.1.- Iniciando HDFS y YARN en Hadoop	15
3.3.2.- Crear el directorio en HDFS	16
3.3.3.- Copiar archivos del sistema local a HDFS	16
3.3.4.- Listar archivos en HDFS	16
3.3.5.- Eliminar archivos en HDFS	17
3.3.6.- Borrar un directorio en HDFS	17
3.3.7.- Verificar el espacio en HDFS	17
4.- Ejercicio 1: Conteo de palabras con MapReduce	18
4.1.- Descargar un libro en formato texto desde Project Gutenberg	18
4.2.- Transferir el archivo a la máquina virtual Linux	19
4.3.- Copiar el archivo al sistema de archivos distribuido HDFS	20
4.4.- Ejecutar el ejemplo wordcount en Hadoop	21
4.5.- Verificar el resultado en HDFS	21
4.6.- Copiar el resultado a la máquina anfitriona	22
4.7.- Conclusión	23
5.- Ejercicio 2: Sudoku	24

5.1.- Descargar dos sudokus	24
5.2.- Crear los archivos en la máquina virtual	25
5.3.- Ejecutar el programa Sudoku en Hadoop	26
5.4.- Comparar los tiempos de ejecución	27
5.4.1.- Datos proporcionados	27
5.4.2.- Explicación de los términos de tiempo	28
5.4.3.- Análisis de los tiempos de ejecución	28
5.4.4.- Copiar los ficheros a la máquina anfitriona	29
5.4.5.- Conclusión	29
6.- Ejercicio 3. Lectura de ficheros log	30
6.1.- Copiar los archivos a la máquina virtual Linux	30
6.2.- Compilar el archivo LogAnalysis.java	31
6.3.- Empaquetar el código en un archivo JAR ejecutable	32
6.4.- Crear un subdirectorio en HDFS y copiar el archivo de log	32
6.5.- Ejecutar el programa LogAnalysis con el archivo de log	33
6.6.- Ver vista previa de los resultados	34
6.7.- Expresión Regular para extraer información	34
6.8.- Copiar el resultado a la máquina anfitriona	35
6.9.- Conclusión	35
7.- Ejercicio 4. Lectura de otro tipo de fichero log	36
7.1.- Copiar el archivo de log a la máquina virtual	36
7.2.- Crear un programa Java utilizando MapReduce	37
7.2.1.- Explicación del código del fichero AndroidLogAnalysis.java	37
7.2.2.- Pasar el fichero AndroidLogAnalysis.java a la máquina virtual	41
7.2.3.- Pasar el fichero Android.log a la máquina virtual	41
7.3.- Compilación del programa AndroidLogAnalysis.java	41
7.4.- Ejecutar el trabajo MapReduce	42
7.5.- Copiar el resultado a la máquina anfitriona	44
7.5.- Conclusión	45
8.- Mapa Mental del Trabajo	46
9.- Conclusiones	47
9.1.- Análisis de los resultados obtenidos	47
9.2.- Desafíos encontrados y soluciones aplicadas	47

9.3.- Reflexión sobre el uso de Hadoop en problemas reales _____ 48

10.- Anexos : Ficheros que se Entrega_____ 49

10.1.- Directorio principal: C:\hadoop _____ 49

10.2.- Subdirectorio: Documentación _____ 49

10.3.- Subdirectorio: Ejercicio1 _____ 49

10.4.- Subdirectorio: Ejercicio2 _____ 50

10.5.- Subdirectorio: Ejercicio3 _____ 50

10.6.- Subdirectorio: Ejercicio4 _____ 50

10.7.- Subdirectorio: Enunciado _____ 51

1.- Introducción

La creciente cantidad de datos generados en el mundo moderno exige el desarrollo de soluciones eficientes para almacenar, procesar y analizar grandes volúmenes de información. En este contexto, Hadoop, una plataforma de software de código abierto, ha emergido como una de las tecnologías más destacadas en el ámbito del procesamiento de Big Data. En particular, **Hadoop Distributed File System (HDFS)** y el paradigma **MapReduce** son dos de sus componentes fundamentales que permiten gestionar y procesar grandes conjuntos de datos de manera eficiente.

1.1.- Objetivo de la actividad

El principal objetivo de esta actividad evaluable es familiarizarse con las herramientas clave de **Hadoop**, en particular con el sistema de ficheros distribuido **HDFS** y el modelo de programación **MapReduce**. A través de la resolución de ejercicios prácticos, el estudiante aprenderá a trabajar con estas tecnologías y a comprender cómo se utilizan para resolver problemas típicos en **Big Data**. Los ejercicios están diseñados para enseñar la manipulación de archivos dentro del sistema **HDFS** y el uso de **MapReduce** para realizar tareas de procesamiento distribuido, como el conteo de palabras en un conjunto de datos o la resolución de problemas complejos, como un Sudoku. Además, se introducen ejercicios relacionados con la lectura y procesamiento de ficheros **log**, lo cual es fundamental para el análisis de grandes volúmenes de datos generados por aplicaciones **web** y **móviles**.

La actividad incluye los siguientes ejercicios clave:

- 1. Conteo de palabras (WordCount):** Implementación de un ejemplo clásico de **MapReduce** en **Hadoop** para contar la frecuencia de las palabras en un archivo de texto. Este ejercicio introduce al estudiante al uso de **MapReduce** para realizar análisis de texto.

2. **Resolución de Sudoku:** Ejecución de un ejemplo de **MapReduce** para resolver un Sudoku. Este ejercicio demuestra cómo **MapReduce** puede utilizarse para resolver problemas algorítmicos complejos.
3. **Lectura de ficheros log de Apache:** Utilización de **MapReduce** para extraer direcciones **IP** de los clientes y los recursos solicitados desde un archivo de log de Apache almacenado en **HDFS**. Este ejercicio se enfoca en la manipulación de grandes volúmenes de datos y la aplicación de expresiones regulares para extraer información relevante.
4. **Lectura de ficheros log de Android:** Implementación de un programa **MapReduce** para analizar logs de **Android**, extrayendo los niveles de log (como "**I**", "**W**", "**E**") y las etiquetas presentes en los registros. Este ejercicio enseña cómo trabajar con diferentes tipos de **logs** y cómo extraer y contar información relevante en un entorno **Android**.

A través de estos ejercicios, se pretende no solo realizar los procedimientos de instalación y ejecución, sino también comprender la lógica y el flujo de trabajo asociados al procesamiento de datos en entornos distribuidos.

1.2.- Importancia de HDFS y MapReduce en el procesamiento de Big Data

1.2.1.- HDFS (Hadoop Distributed File System)

HDFS es un sistema de ficheros distribuido que permite almacenar grandes cantidades de datos de manera escalable y confiable. La arquitectura de **HDFS** se basa en la idea de dividir los archivos en bloques grandes, que luego se distribuyen a través de una red de máquinas. Estos bloques se replican en múltiples nodos para asegurar la tolerancia a fallos, lo que significa que, si un nodo falla, el sistema sigue siendo funcional debido a las copias de los bloques en otros nodos.

El uso de **HDFS** es fundamental en **Big Data** porque permite manejar volúmenes masivos de datos de manera eficiente. Sin embargo, su capacidad de almacenamiento distribuido por sí sola no

es suficiente para procesar grandes conjuntos de datos. Para ello, es necesario contar con un marco que permita realizar procesamiento paralelo y distribuido en esos datos almacenados.

1.2.2.- MapReduce

MapReduce es un modelo de programación diseñado para el procesamiento distribuido de grandes conjuntos de datos. Divide el procesamiento de un problema en dos fases clave: la fase de "**mapa**" y la fase de "**reducción**".

- **Fase de Mapa (Map):** En esta fase, se toma un conjunto de datos y se distribuye entre los nodos del clúster. Cada nodo realiza un trabajo de procesamiento sobre una pequeña porción de los datos, generando pares clave-valor. Este proceso es paralelo, lo que significa que puede realizarse en muchos nodos de manera simultánea.
- **Fase de Reducción (Reduce):** Despues de la fase de mapa, los pares **clave-valor** generados se agrupan según su **clave**. Los valores correspondientes a la misma **clave** son procesados por los nodos encargados de la fase de reducción. El resultado es una salida consolidada que resume o agrega los datos de la fase de mapa.

El modelo **MapReduce** permite que los trabajos de procesamiento se ejecuten de manera eficiente en un **clúster** de máquinas, distribuyendo la carga de trabajo entre varias instancias y aprovechando el procesamiento paralelo. Este modelo es fundamental para tareas como el análisis de grandes volúmenes de datos, el procesamiento de logs de **servidor**, el análisis de redes sociales, el análisis de patrones de comportamiento y muchas otras aplicaciones de **Big Data**.

1.2.3.- Relación entre HDFS y MapReduce

HDFS y **MapReduce** están diseñados para trabajar juntos de manera eficiente. Mientras que **HDFS** permite el almacenamiento distribuido de datos en grandes volúmenes, **MapReduce** proporciona la capacidad de procesar esos datos de manera eficiente y distribuida.

Juntos, forman la columna vertebral del ecosistema **Hadoop**, una plataforma de procesamiento de **Big Data** que puede manejar tareas complejas y volúmenes masivos de datos.

HDFS proporciona una infraestructura fiable para almacenar los datos, y **MapReduce** ofrece un marco de programación para realizar cálculos complejos sobre esos datos. Esta integración es clave en el procesamiento eficiente de **Big Data**, permitiendo a las empresas y organizaciones analizar grandes volúmenes de información de manera rápida y escalable.

En resumen, tanto **HDFS** como **MapReduce** son esenciales para abordar los desafíos del procesamiento de **Big Data**, permitiendo no solo el almacenamiento de grandes cantidades de información, sino también su procesamiento paralelo a través de múltiples nodos en un clúster. La combinación de estas herramientas facilita el análisis de grandes volúmenes de datos en entornos distribuidos, lo que las hace indispensables en el mundo actual, donde la generación de datos es cada vez más masiva y diversa.

2.- Escenario de trabajo

En esta actividad, se utilizará una máquina virtual con **Debian** preconfigurada con **Hadoop**. El entorno de trabajo ha sido diseñado para facilitar el acceso y la configuración del sistema, permitiendo a los estudiantes interactuar con las herramientas de procesamiento de Big Data de manera eficiente.

2.1.- Acceso a la máquina virtual desde VirtualBox

Para trabajar con la máquina virtual proporcionada, se debe seguir el procedimiento descrito a continuación:

1. Descargar la máquina virtual:

- La máquina virtual con **Debian** y **Hadoop** ya está disponible para su descarga en Moodle. En caso de no disponer de ella, los estudiantes pueden descargarla y moverla a una carpeta de su elección.

2. Agregar la máquina virtual a VirtualBox:

- Una vez que la máquina virtual ha sido descargada, simplemente haz clic en el archivo con extensión **.ova**. VirtualBox se abrirá automáticamente y procederá a importar la máquina virtual.

3. Iniciar la máquina virtual:

- Despues de agregar la máquina virtual a VirtualBox, se puede iniciar la máquina virtual con un simple clic sobre ella y luego seleccionar la opción "**Iniciar**". Esto abrirá una nueva ventana con el sistema Debian en ejecución.

4. Acceso a la consola:

- La máquina virtual está configurada para ser utilizada a través de la consola. Una vez iniciada, se podrá acceder al sistema **Debian** y ejecutar los comandos necesarios para trabajar con **Hadoop**.

2.2.- Requisitos y credenciales

En cuanto a los requisitos y credenciales para el acceso a la máquina virtual:

1. Credenciales de acceso:

- **Usuario root**: La contraseña para el usuario root es "**Admin1.**" (sin las comillas).
- **Usuario hadoop**: El usuario que se utiliza para gestionar Hadoop tiene la misma contraseña, es decir, **Admin1**.

2. Puertos redirigidos:

- **Puerto SSH (22)**: El puerto **22** de la máquina virtual, lo que permite la conexión remota mediante **SSH**. Esto es útil si se desea acceder a la máquina virtual desde fuera de VirtualBox.
- **Puerto HDFS (9870)**: El puerto **9870** de la máquina virtual está redireccionado al puerto **9880** del equipo anfitrión. Este puerto se utiliza para acceder a la interfaz web de **HDFS** (**Hadoop Distributed File System**), donde se pueden realizar operaciones de gestión de archivos y directorios dentro del sistema distribuido de **Hadoop**.

3. Acceso remoto mediante SSH:

- Para acceder de manera remota a la máquina virtual utilizando **SSH**, se debe usar la dirección **IP** de la máquina virtual y el

Actividad evaluable 2.1.- Hadoop: MapReduce y HDFS

puerto redirigido (22). Se puede conectar desde el terminal utilizando el siguiente comando:

```
hadoop@nodo1: ~ X + v - □ ×

C:\hadoop>ssh hadoop@192.168.0.158 -p 22
hadoop@192.168.0.158's password:
Linux nodo1 6.1.0-26-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.112-1 (2024-09-30) 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.
Last login: Sat Mar 15 10:07:37 2025
hadoop@nodo1:~$
```

- En esta pantalla, accedemos a la máquina virtual mediante el comando “`ssh hadoop@192.168.0.158 -p 22`”, pulsamos Intro e ingresamos la contraseña “`hadoop`”.

2.3.- Comprobación y arranque del clúster

```
hadoop@nodo1: ~ x + v - □ ×  
C:\hadoop>ssh hadoop@192.168.0.158 -p 22  
hadoop@192.168.0.158's password:  
Linux nodo1 6.1.0-26-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.112-1 (2024-09-30) 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.  
Last login: Sat Mar 15 10:07:37 2025  
hadoop@nodo1:~$
```

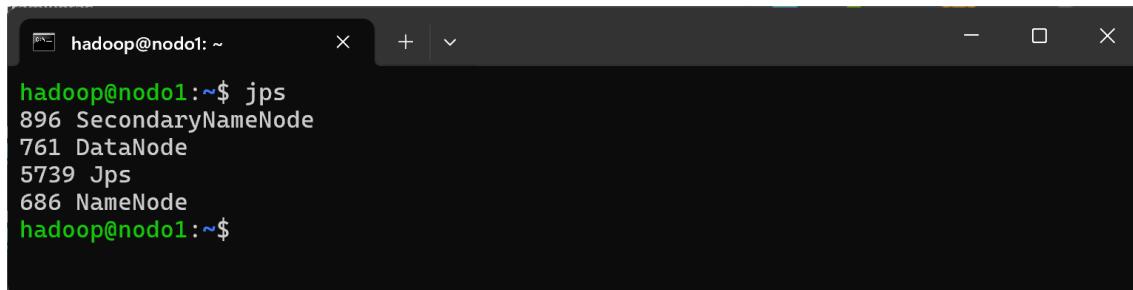
En esta pantalla, accedemos a la máquina virtual mediante el comando “`ssh hadoop@192.168.0.158 -p 22`”, pulsamos Intro e ingresamos la contraseña “`hadoop`”.

Para formatear el sistema de archivos HDFS antes de iniciar el clúster, ejecuta el comando “**hadoop namenode -format**”, que prepara el NameNode para gestionar el almacenamiento distribuido.



```
hadoop@nodo1:~$ start-dfs.sh
Starting namenodes on [nodo1]
Starting datanodes
Starting secondary namenodes [nodo1]
hadoop@nodo1:~$
```

Para iniciar los demonios necesarios para **HDFS** en **Hadoop**, ejecuta el comando **start-dfs.sh**, que lanza los servicios del **NameNode**, **SecondaryNameNode** y **DataNodes** según la configuración del clúster.



```
hadoop@nodo1:~$ jps
896 SecondaryNameNode
761 DataNode
5739 Jps
686 NameNode
hadoop@nodo1:~$
```

Para verificar que los procesos de **Hadoop** están funcionando correctamente, ejecuta el comando “**jps**”, que mostrará los procesos activos como **NameNode** y **DataNode** si todo está configurado adecuadamente.

2.4.- Conexión a la interfaz web de Hadoop

Para facilitar la interacción con **Hadoop** y **HDFS**, la interfaz web de **Hadoop** es accesible a través de un navegador utilizando el puerto redirigido **9870**. Esto permite ver el estado de **HDFS** y gestionar archivos de manera visual. La **URL** para acceder sería:

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

- Address Bar:** 192.168.0.158:9870/dfshealth.html#tab-overview
- Tab Bar:** Hadoop (selected), Overview, Datanodes, Datanode Volume Failures, Snapshot, Startup Progress, Utilities
- Content Area:**
 - Overview:** 'localhost:9000' (✓active)
 - Cluster Statistics:**

Started:	Sat Mar 15 11:12:34 +0100 2025
Version:	3.4.0, rdb877f398fb26bb7791783192ee7a5dfaeecc760
Compiled:	Mon Mar 04 07:35:00 +0100 2024 by root from (HEAD detached at release-3.4.0-RC3)
Cluster ID:	CID-f7ca6271-5edc-4aac-8b33-792517c62451
Block Pool ID:	BP-1497494967-10.0.2.15-1742033362326
 - 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 26.58 MB of 43.29 MB Heap Memory. Max Heap Memory is 475.63 MB.
Non Heap Memory used 52.11 MB of 53.31 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	0 B
Configured Remote Capacity:	0 B
DFS Used:	0 B (0%)

En esta pantalla, accedemos a **Hadoop** a través del navegador web introduciendo la dirección **192.168.0.158:9870** en la barra de **URL**.

En resumen, con estos pasos, tendrás acceso a la máquina virtual con **Hadoop** configurado, y podrás comenzar a trabajar en el procesamiento de **Big Data** utilizando las herramientas disponibles en el entorno.

3.- Comunicación entre entornos

En el contexto del uso de máquinas virtuales con herramientas como **Hadoop**, es esencial poder interactuar tanto con la máquina virtual como con el sistema de archivos distribuido (**HDFS**) desde diferentes entornos. A continuación, se describen las principales formas de comunicación entre la máquina **anfitriona** y la máquina virtual, así como la transferencia de archivos y las operaciones básicas en **HDFS**.

3.1.- Comunicación entre máquina anfitriona y máquina virtual

La máquina **anfitriona** es el sistema operativo en el que se ejecuta VirtualBox, mientras que la máquina virtual es un entorno aislado donde se ejecuta **Debian** con **Hadoop**. Para garantizar que los dos entornos puedan comunicarse, se deben configurar correctamente las redes de VirtualBox y los puertos de la máquina virtual.

1. Configuración de red en VirtualBox:

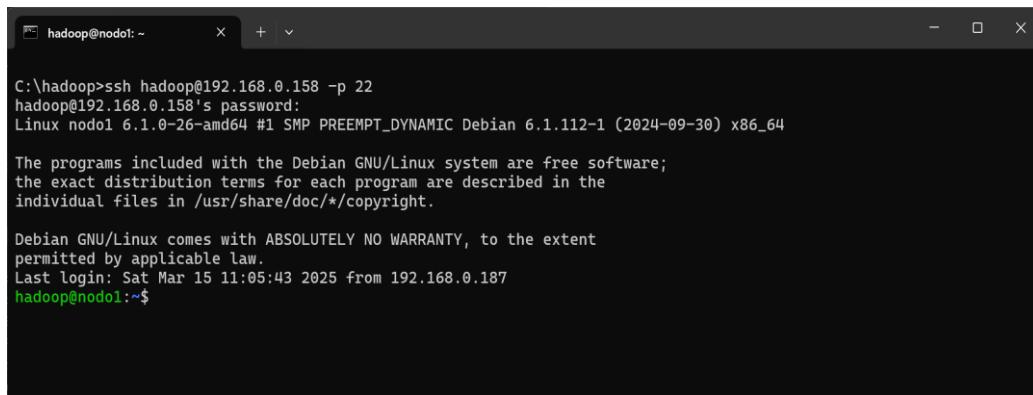
- La máquina virtual debe estar configurada para permitir la comunicación con la máquina anfitriona. La forma más común de hacerlo es a través de la red "**Adaptador puente**" o "**Red interna**", dependiendo de cómo se desee establecer la conexión.
- En caso de necesitar acceso a servicios remotos, como **SSH**, o transferencias de archivos, lo más sencillo es usar la configuración de **red NAT** o **red interna** con puertos redirigidos.

2. Acceso a la máquina virtual desde la máquina anfitriona:

- Cuando la máquina virtual está correctamente configurada con red NAT y puertos redirigidos, puedes acceder a ella utilizando su IP interna (por ejemplo, **192.168.0.158**).

- Si se desea realizar una conexión **SSH** desde la máquina anfitriona, la dirección **IP** de la máquina virtual debe ser conocida. Esto se puede verificar con el comando **ip a** dentro de la máquina virtual, o bien usar el puerto redirigido (por ejemplo, el puerto **22** para **SSH**).

3. Conexión SSH a la máquina virtual:

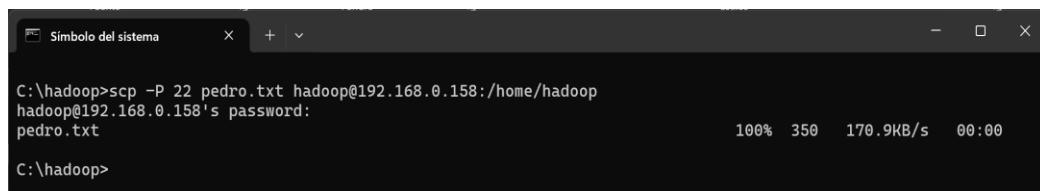


A screenshot of a terminal window titled "hadoop@nodo1: ~". The window contains the following text:
C:\hadoop>ssh hadoop@192.168.0.158 -p 22
hadoop@192.168.0.158's password:
Linux nodo1 6.1.0-26-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.112-1 (2024-09-30) x86_64
The programs included with the Debian GNU/Linux 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.
Last login: Sat Mar 15 11:05:43 2025 from 192.168.0.187
hadoop@nodo1:~\$

En esta pantalla, se ejecuta el comando "**ssh hadoop@192.168.0.158 -p 22**" presionando la tecla Intro para iniciar la conexión SSH al servidor remoto.

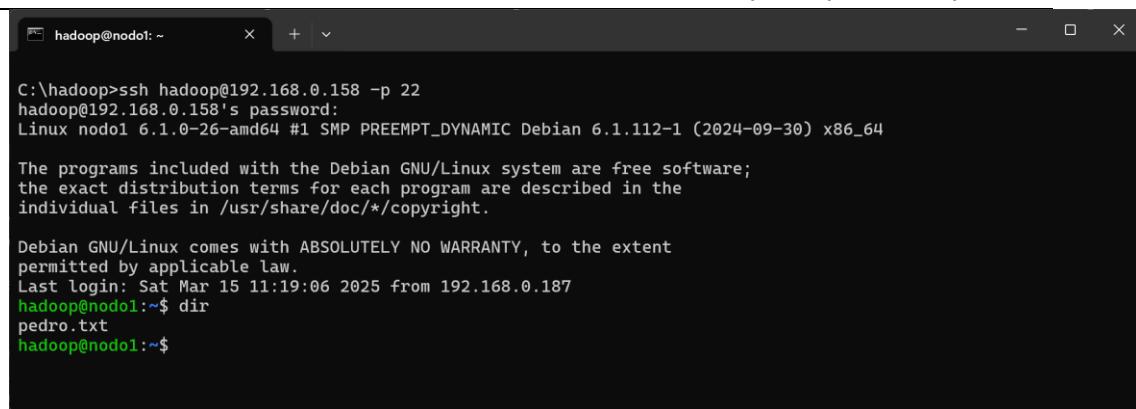
3.2.- Copia de archivos usando SCP

1. Copiar archivos desde la máquina anfitriona a la máquina virtual:



A screenshot of a terminal window titled "Símbolo del sistema". The window contains the following text:
C:\hadoop>scp -P 22 pedro.txt hadoop@192.168.0.158:/home/hadoop
hadoop@192.168.0.158's password:
pedro.txt 100% 350 170.9KB/s 00:00
C:\hadoop>

En esta pantalla, se ejecuta el comando "**scp -P 22 pedro.txt hadoop@192.168.0.158:/home/hadoop**" presionando la tecla Intro para iniciar la transferencia segura del archivo "**pedro.txt**" al servidor remoto.



```
C:\hadoop>ssh hadoop@192.168.0.158 -p 22
hadoop@192.168.0.158's password:
Linux nodo1 6.1.0-26-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.112-1 (2024-09-30) 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.

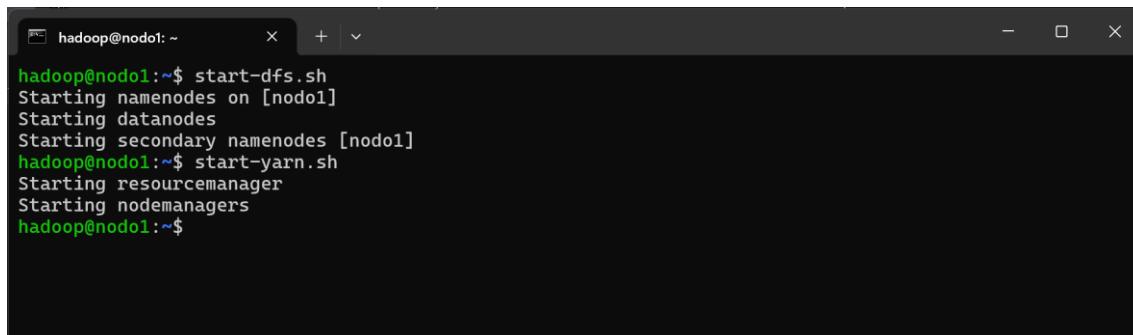
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Mar 15 11:19:06 2025 from 192.168.0.187
hadoop@nodo1:~$ dir
pedro.txt
hadoop@nodo1:~$
```

En esta pantalla, se verifica si el archivo se ha copiado correctamente a la máquina virtual ejecutando el comando "**ssh hadoop@192.168.0.158 -p 22**", presionando la tecla Intro y luego realizando un **dir** para confirmar la presencia del archivo.

3.3.- Comunicación entre sistema de ficheros local y HDFS

El sistema de ficheros distribuido de **Hadoop (HDFS)** permite almacenar grandes volúmenes de datos de manera eficiente en un **clúster** de máquinas. Para interactuar con **HDFS** desde la máquina virtual o desde la máquina **anfitriona**, se utilizan comandos específicos que permiten subir, bajar y gestionar archivos dentro de **HDFS**.

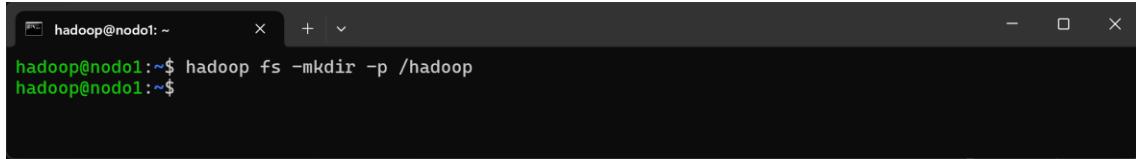
3.3.1.- Iniciando HDFS y YARN en Hadoop



```
hadoop@nodo1:~$ start-dfs.sh
Starting namenodes on [nodo1]
Starting datanodes
Starting secondary namenodes [nodo1]
hadoop@nodo1:~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hadoop@nodo1:~$
```

En esta pantalla, se ejecutan los comandos "**start-dfs.sh**" y "**start-yarn.sh**", presionando la tecla Intro después de cada uno, para iniciar los procesos del sistema de archivos distribuido y el gestor de recursos de Hadoop.

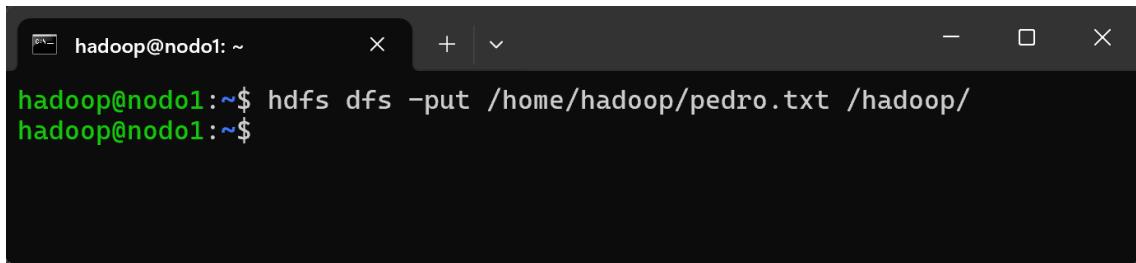
3.3.2.- Crear el directorio en HDFS



```
hadoop@nodo1: ~      + | ~
hadoop@nodo1:~$ hadoop fs -mkdir -p /hadoop
hadoop@nodo1:~$
```

En esta pantalla, se procede a crear un directorio en el sistema local de **HDFS** dentro de la máquina virtual utilizando el comando "**hadoop fs -mkdir -p /hadoop**" y se ejecuta presionando la tecla Enter

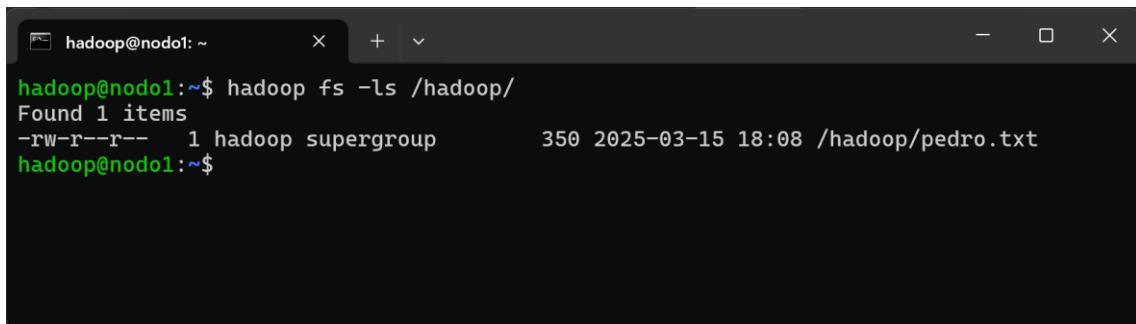
3.3.3.- Copiar archivos del sistema local a HDFS



```
hadoop@nodo1: ~      + | ~
hadoop@nodo1:~$ hdfs dfs -put /home/hadoop/pedro.txt /hadoop/
hadoop@nodo1:~$
```

En esta pantalla, copiamos el archivo "pedro.txt" al directorio Hadoop utilizando el comando "**hdfs dfs -put /home/hadoop/pedro.txt /hadoop/**", y luego pulsamos la tecla Intro para ejecutarlo.

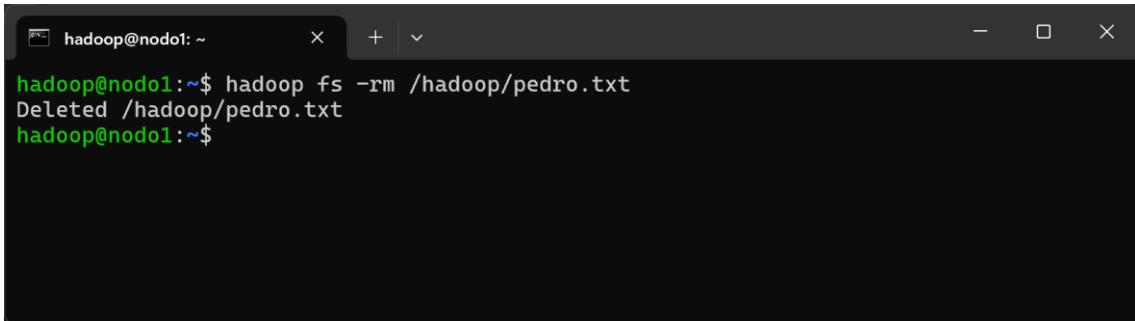
3.3.4.- Listar archivos en HDFS



```
hadoop@nodo1: ~      + | ~
hadoop@nodo1:~$ hadoop fs -ls /hadoop/
Found 1 items
-rw-r--r--  1 hadoop supergroup      350 2025-03-15 18:08 /hadoop/pedro.txt
hadoop@nodo1:~$
```

En esta pantalla, se lista los archivos en Hadoop utilizando el comando "**hadoop fs -ls /hadoop/**", y luego pulsamos la tecla Intro para ejecutarlo.

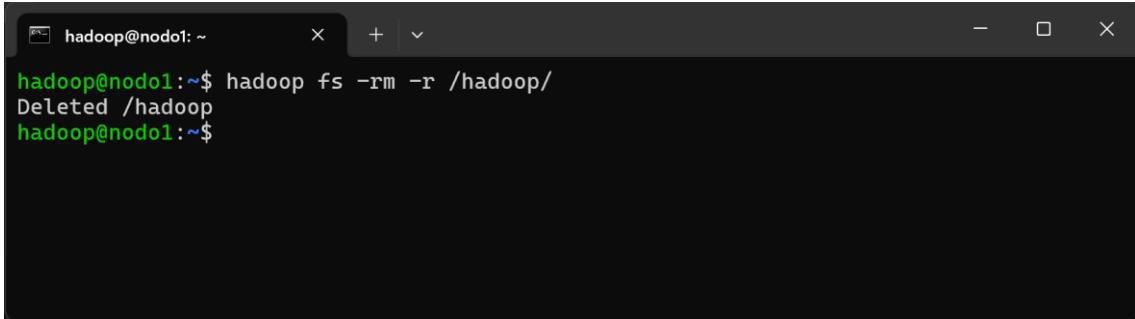
3.3.5.- Eliminar archivos en HDFS



```
hadoop@nodo1: ~      + | v
hadoop@nodo1:~$ hadoop fs -rm /hadoop/pedro.txt
Deleted /hadoop/pedro.txt
hadoop@nodo1:~$
```

En esta pantalla, eliminamos el archivo "pedro.txt" con el comando "**hadoop fs -rm /hadoop/pedro.txt**", y luego pulsamos la tecla Intro para ejecutarlo.

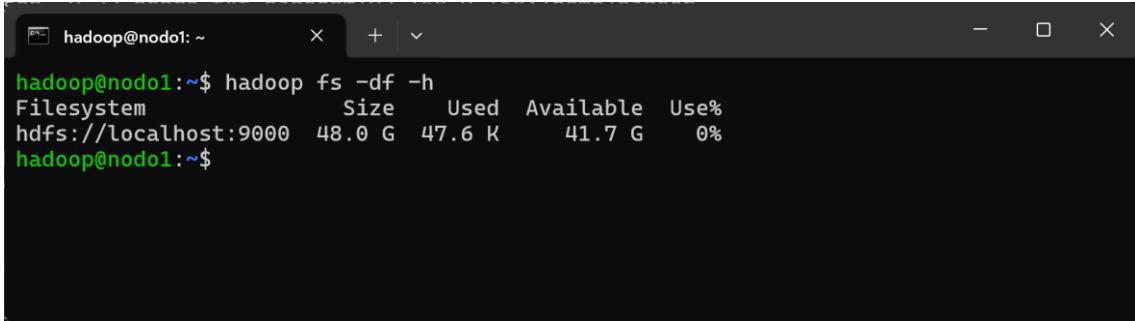
3.3.6.- Borrar un directorio en HDFS



```
hadoop@nodo1: ~      + | v
hadoop@nodo1:~$ hadoop fs -rm -r /hadoop/
Deleted /hadoop/
hadoop@nodo1:~$
```

En esta pantalla, eliminamos el directorio "hadoop" con el comando "**hadoop fs -rm -r /hadoop/**", y luego pulsamos la tecla Intro para ejecutarlo.

3.3.7.- Verificar el espacio en HDFS

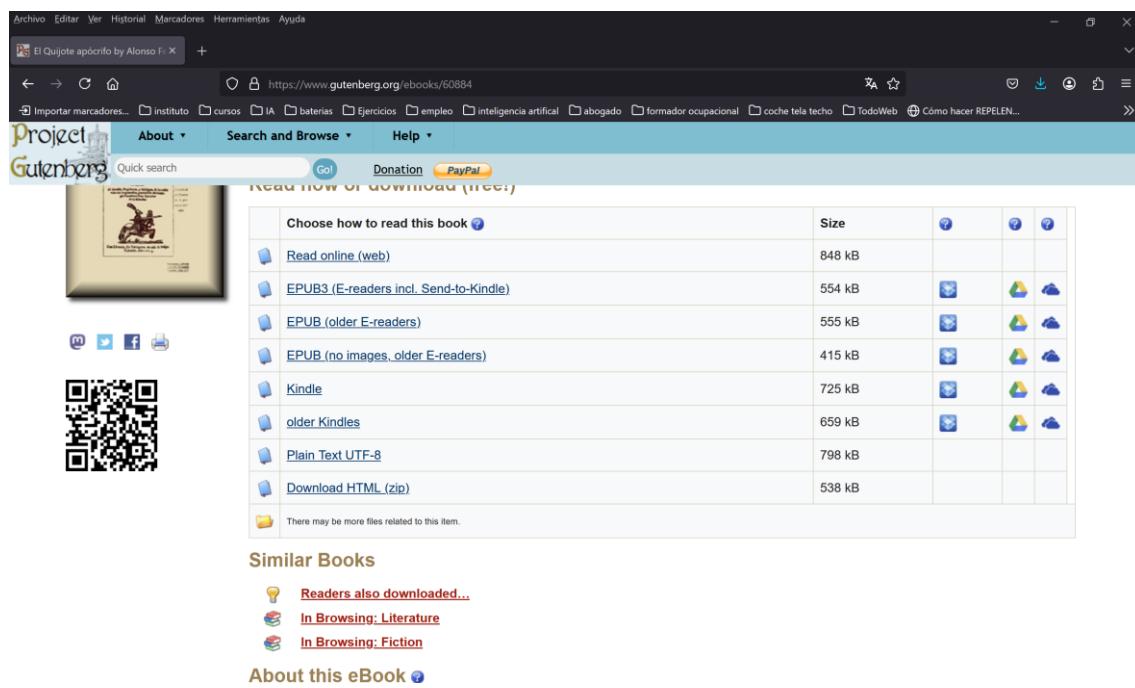


```
hadoop@nodo1: ~      + | v
hadoop@nodo1:~$ hadoop fs -df -h
Filesystem          Size   Used  Available  Use%
hdfs://localhost:9000  48.0 G  47.6 K    41.7 G    0%
hadoop@nodo1:~$
```

En esta pantalla, verificamos el espacio disponible en HDFS utilizando el comando “`hadoop fs -df -h`”, y luego pulsamos la tecla Intro para ejecutarlo.

4.- Ejercicio 1: Conteo de palabras con MapReduce

4.1.- Descargar un libro en formato texto desde Project Gutenberg



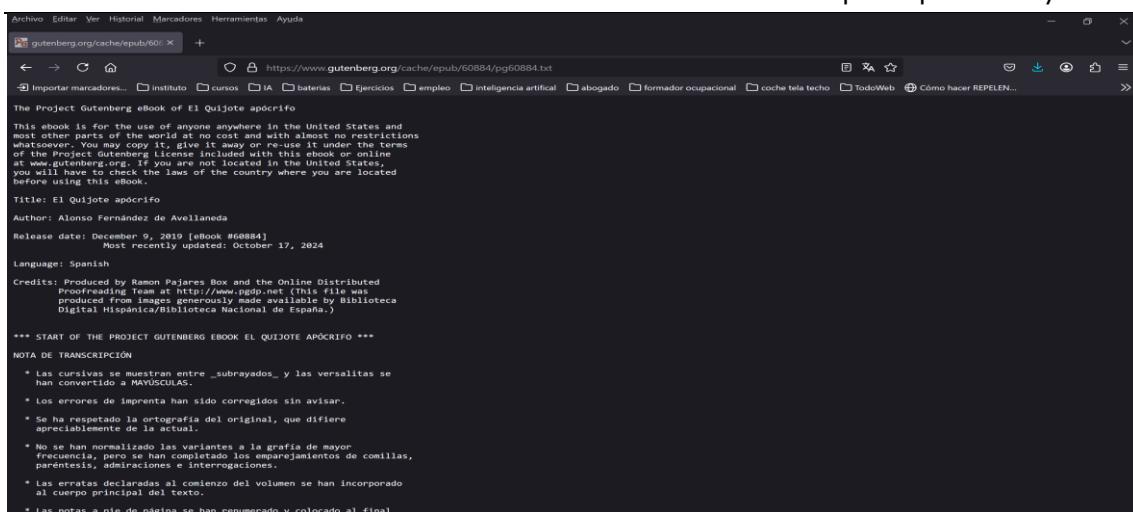
The screenshot shows a web browser displaying the Gutenberg eBook page for "El Quijote apócrifo by Alonso F.". The page includes a menu bar with Archivo, Editar, Ver, Historial, Marcadores, Herramientas, Ayuda, and a search bar. Below the menu is a navigation bar with links for Importar marcadores..., Instituto, cursos, IA, baterías, Ejercicios, empleo, inteligencia artificial, abogado, formador ocupacional, coche tela techo, TodoWeb, and Cómo hacer REPELEN... A "Project Gutenberg" logo is on the left, and a "Quick search" input field is present. The main content area features a thumbnail of the book cover, social sharing icons (m, t, f, l), and a QR code. A table titled "Read now or download (free!)" lists various file formats and their sizes:

Choose how to read this book	Size	Read online (web)	EPUB3 (E-readers incl. Send-to-Kindle)	EPUB (older E-readers)	EPUB (no images, older E-readers)	Kindle	older Kindles	Plain Text UTF-8	Download HTML (.zip)
	848 kB								
	554 kB								
	555 kB								
	415 kB								
	725 kB								
	659 kB								
	798 kB								
	538 kB								

A note at the bottom of the table says "There may be more files related to this item." Below the table, sections for "Similar Books" and "About this eBook" are visible, each with a list of links.

En esta pantalla, se descarga el libro "El Quijote apócrifo" en formato de texto UTF-8 desde el enlace "Plain Text UTF-8" de la página "<https://www.gutenberg.org/ebooks/60884>".

Actividad evaluable 2.1.- Hadoop: MapReduce y HDFS



En esta pantalla, se descarga el archivo haciendo clic derecho sobre el texto, seleccionando "Guardar como", nombrándolo "["quijote.txt"](#)" y finalmente presionando el botón "Guardar".

4.2.- Transferir el archivo a la máquina virtual Linux

```
hadoop@nodo1:~$ mkdir ejercicio1
hadoop@nodo1:~$ ls -l
total 8
drwxr-xr-x 2 hadoop hadoop 4096 mar 15 18:48 ejercicio1
-rw-r--r-- 1 hadoop hadoop 350 mar 15 17:47 pedro.txt
hadoop@nodo1:~$
```

En esta pantalla, creamos el directorio de trabajo "["ejercicio1"](#)" utilizando el comando "["mkdir ejercicio1"](#)", y luego pulsamos la tecla Intro para ejecutarlo.

```
C:\hadoop\Ejercicio1>scp quijote.txt hadoop@192.168.0.158:/home/hadoop/ejercicio1/
hadoop@192.168.0.158's password:
quijote.txt                                         100%  798KB  16.2MB/s   00:00
C:\hadoop\Ejercicio1>
```

En esta pantalla, se transfiere el archivo "["quijote.txt"](#)" a la máquina virtual Linux ejecutando el comando "["scp quijote.txt hadoop@192.168.0.158:/home/hadoop/ejercicio1"](#)", seguido de presionar la tecla Intro e ingresar la contraseña.

```
hadoop@nodo1:~/ejercicio1$ dir
quijote.txt
hadoop@nodo1:~/ejercicio1$
```

En esta pantalla, se verifica con el comando “**dir**” que el archivo “**quijote.txt**” se ha copiado en el directorio “**ejercicio1**” correctamente en la máquina virtual **Linux**.

4.3.- Copiar el archivo al sistema de archivos distribuido HDFS

```
hadoop@nodo1:~/ejercicio1$ hdfs dfs -mkdir /ejercicio1
hadoop@nodo1:~/ejercicio1$
```

En esta pantalla, creamos el directorio “**ejercicio1**” en **HDFS** utilizando el comando “**hdfs dfs -mkdir /ejercicio1**”, y luego pulsamos la tecla Intro para ejecutarlo.

```
hadoop@nodo1:~/ejercicio1$ hdfs dfs -mkdir /ejercicio1/wordcount
hadoop@nodo1:~/ejercicio1$ |
```

En esta pantalla, creamos el directorio “**wordcount**” dentro de “**/ejercicio1**” en **HDFS** utilizando el comando “**hdfs dfs -mkdir /ejercicio1/wordcount**”, y luego pulsamos la tecla Intro para ejecutarlo.

```
hadoop@nodo1:~/ejercicio1$ hdfs dfs -put /home/hadoop/ejercicio1/quijote.txt /ejercicio1/wordcount/
hadoop@nodo1:~/ejercicio1$
```

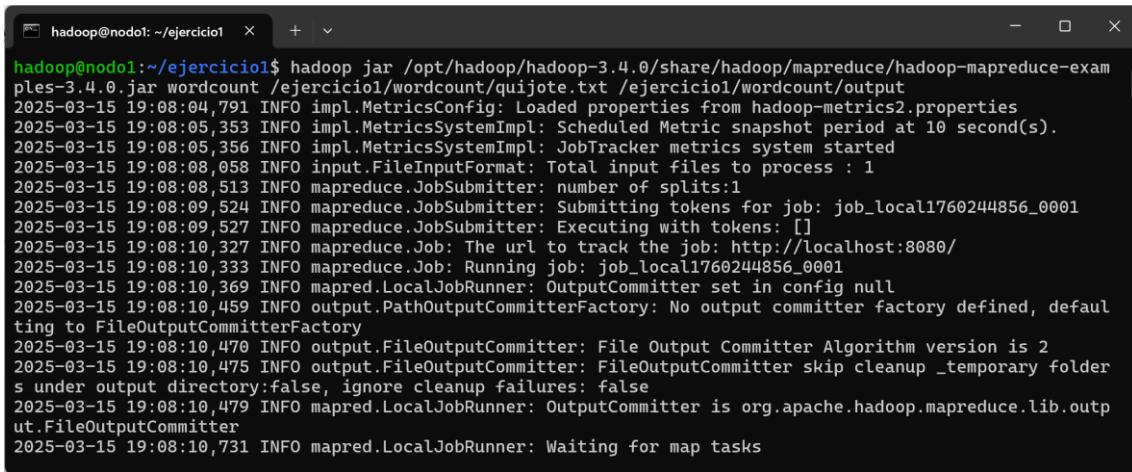
En esta pantalla, se copia el archivo “**quijote.txt**” al directorio **wordcount** utilizando el comando “**hdfs dfs -put /home/hadoop/ejercicio1/quijote.txt /ejercicio1/wordcount/**”, seguido de pulsar la tecla Intro para ejecutarlo.

```
hadoop@nodo1:~/ejercicio1$ hdfs dfs -ls /ejercicio1/wordcount/
Found 1 items
-rw-r--r-- 1 hadoop supergroup 817416 2025-03-15 19:01 /ejercicio1/wordcount/quijote.txt
hadoop@nodo1:~/ejercicio1$
```

Actividad evaluable 2.1.- Hadoop: MapReduce y HDFS

En esta pantalla, se verifica si el archivo se ha copiado correctamente al directorio “**wordcount**” utilizando el comando “**hdfs dfs -ls /ejercicio1/wordcount/**”, seguido de pulsar la tecla Intro para ejecutarlo.

4.4.- Ejecutar el ejemplo wordcount en Hadoop



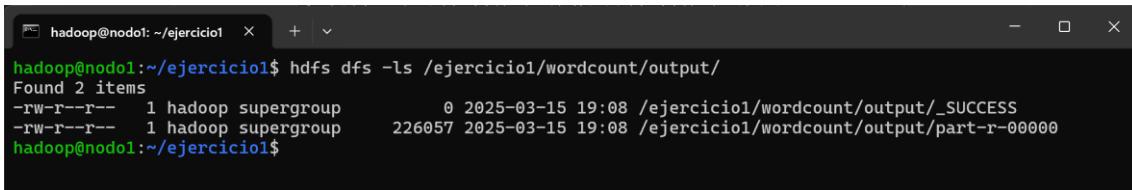
```

hadoop@nodo1:~/ejercicio1$ hadoop jar /opt/hadoop/hadoop-3.4.0/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.4.0.jar wordcount /ejercicio1/wordcount/quiero.txt /ejercicio1/wordcount/output
2025-03-15 19:08:04,791 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2025-03-15 19:08:05,353 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2025-03-15 19:08:05,356 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2025-03-15 19:08:08,058 INFO input.FileInputFormat: Total input files to process : 1
2025-03-15 19:08:08,513 INFO mapreduce.JobSubmitter: number of splits:1
2025-03-15 19:08:09,524 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1760244856_0001
2025-03-15 19:08:09,527 INFO mapreduce.JobSubmitter: Executing with tokens: []
2025-03-15 19:08:10,327 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2025-03-15 19:08:10,333 INFO mapreduce.Job: Running job: job_local1760244856_0001
2025-03-15 19:08:10,369 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2025-03-15 19:08:10,459 INFO output.PathOutputCommitterFactory: No output committer factory defined, defaulting to FileOutputCommitterFactory
2025-03-15 19:08:10,470 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2025-03-15 19:08:10,475 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2025-03-15 19:08:10,479 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
2025-03-15 19:08:10,731 INFO mapred.LocalJobRunner: Waiting for map tasks

```

En esta pantalla, se ejecuta el comando “**hadoop jar /opt/hadoop/hadoop-3.4.0/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.4.0.jar wordcount /ejercicio1/wordcount/quiero.txt /ejercicio1/wordcount/output**” para iniciar el programa MapReduce **wordcount** en Hadoop, procesando el archivo *quiero.txt* del directorio */wordcount/* para contar la frecuencia de cada palabra y almacenar los resultados en el directorio “*/ejercicio1/wordcount/output*”.

4.5.- Verificar el resultado en HDFS

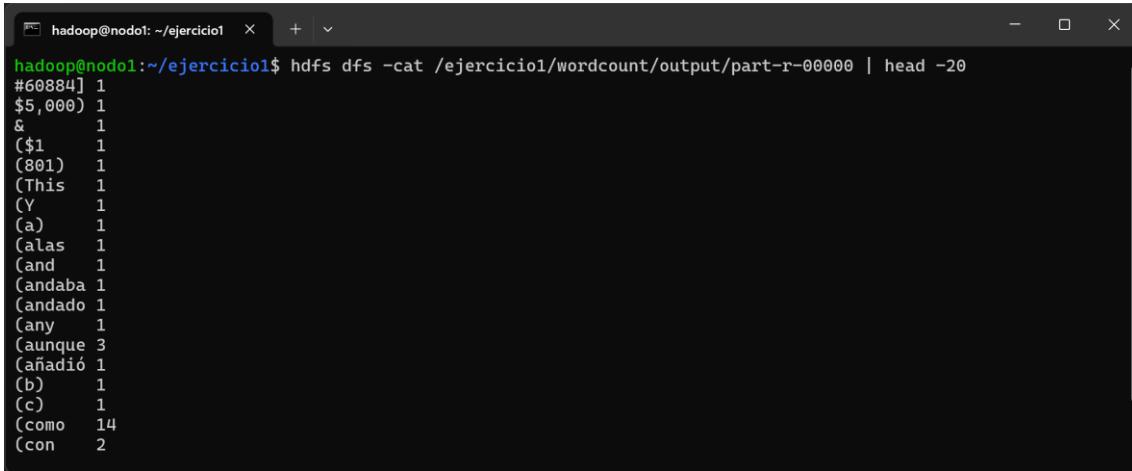


```

hadoop@nodo1:~/ejercicio1$ hdfs dfs -ls /ejercicio1/wordcount/output/
Found 2 items
-rw-r--r-- 1 hadoop supergroup          0 2025-03-15 19:08 /ejercicio1/wordcount/output/_SUCCESS
-rw-r--r-- 1 hadoop supergroup 226057 2025-03-15 19:08 /ejercicio1/wordcount/output/part-r-00000
hadoop@nodo1:~/ejercicio1$ 

```

En esta pantalla, tras finalizar la ejecución del programa MapReduce, se utiliza el comando “**hdfs dfs -ls /ejercicio1/wordcount/output/**” para listar los archivos generados en el directorio de salida, donde se espera encontrar archivos como ***part-r-00000*** que contienen el conteo de palabras si la ejecución fue exitosa.



```
hadoop@nodo1:~/ejercicio1$ hadoop dfs -cat /ejercicio1/wordcount/output/part-r-00000 | head -20
#60884] 1
$5,000) 1
& 1
($1 1
(801) 1
(This 1
(Y 1
(a) 1
(alas 1
(and 1
(andalaba 1
(andalado 1
(any 1
(aunque 3
(añadió 1
(b) 1
(c) 1
(como 14
(con 2
```

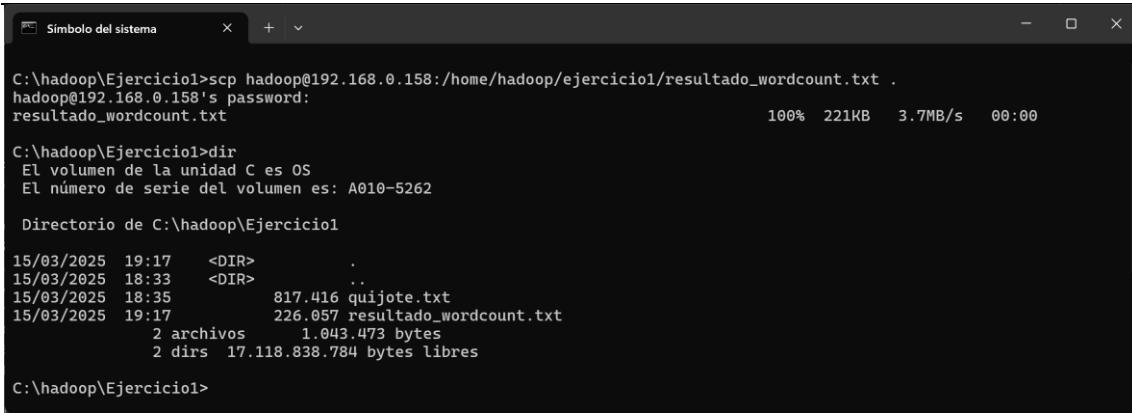
En esta pantalla, se procede a visualizar las primeras líneas del resultado, se ejecuta el comando “`hdfs dfs -cat /ejercicio1/wordcount/output/part-r-00000 | head -20`” para mostrar las primeras veinte líneas del archivo *part-r-00000*.

4.6.- Copiar el resultado a la máquina anfitriona



```
hadoop@nodo1:~/ejercicio1$ hdfs dfs -get /ejercicio1/wordcount/output/part-r-00000 /home/hadoop/ejercicio1/resultado_wordcount.txt
hadoop@nodo1:~/ejercicio1$ dir
quierote.txt  resultado_wordcount.txt
hadoop@nodo1:~/ejercicio1$
```

En esta pantalla, se procede a copiar los resultados desde HDFS a la máquina virtual, se ejecuta el comando “`hdfs dfs -get /ejercicio1/wordcount/output/part-r-00000 /home/hadoop/ejercicio1/resultado_wordcount.txt`” para transferir el archivo *part-r-00000* a un archivo local llamado *resultado_wordcount.txt*, se verifica que el archivo *resultado_wordcount.txt* se ha copiado correctamente a la máquina virtual utilizando el comando “`dir`”.



```
C:\hadoop\Ejercicio1>scp hadoop@192.168.0.158:/home/hadoop/ejercicio1/resultado_wordcount.txt .
hadoop@192.168.0.158's password:
resultado_wordcount.txt                                         100%  221KB   3.7MB/s  00:00

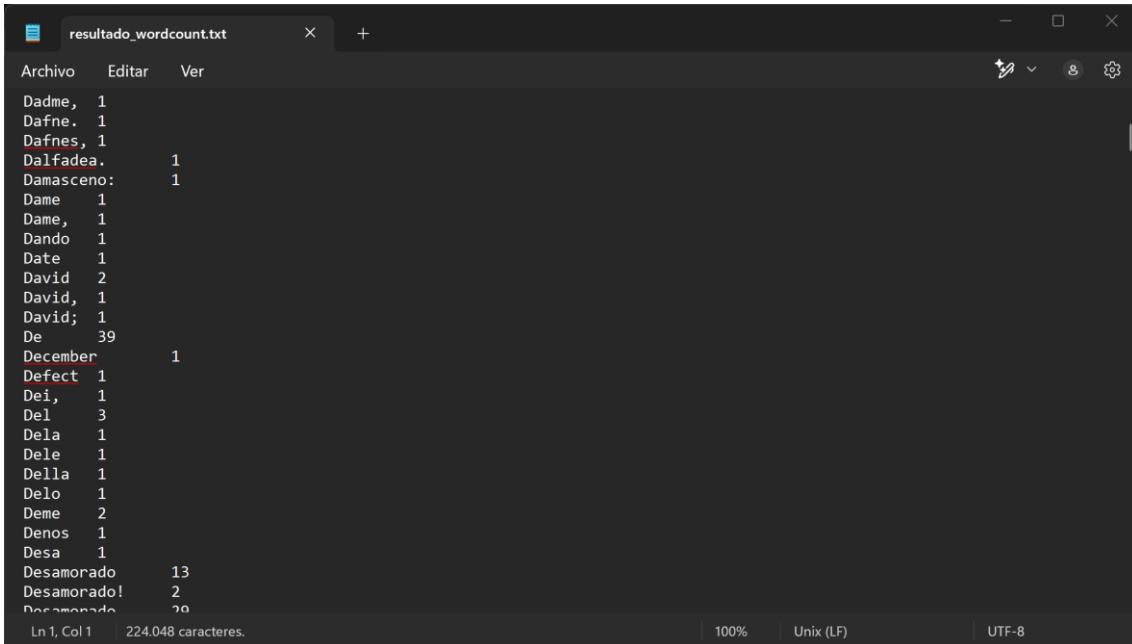
C:\hadoop\Ejercicio1>dir
El volumen de la unidad C es OS
El número de serie del volumen es: A010-5262

Directorio de C:\hadoop\Ejercicio1

15/03/2025 19:17    <DIR>    .
15/03/2025 18:33    <DIR>    ..
15/03/2025 18:35        817.416 quijote.txt
15/03/2025 19:17        226.057 resultado_wordcount.txt
                           2 archivos      1.043.473 bytes
                           2 dirs   17.118.838.784 bytes libres

C:\hadoop\Ejercicio1>
```

En esta pantalla, se utiliza el comando "**scp** **hadoop@192.168.0.158:/home/hadoop/ejercicio1/resultado_wordcount.txt .**" para copiar el archivo **resultado_wordcount.txt** desde la máquina virtual a la máquina anfitriona, y luego se ejecuta el comando "**dir**" para verificar que la copia se ha realizado correctamente.



```
Dadme, 1
Dafne, 1
Dafnes, 1
Dalfadea, 1
Damasceno, 1
Dame 1
Dame, 1
Dando 1
Date 1
David 2
David, 1
David; 1
De 39
December 1
Defect 1
Dei, 1
Del 3
Dela 1
Dele 1
Della 1
Delo 1
Deme 2
Denos 1
Desa 1
Desamorado 13
Desamorado! 2
Desamorodo 20
Ln 1, Col 1  224.048 caracteres.
```

En esta pantalla, se abre el archivo "**resultado_wordcount.txt**" con el Bloc de Notas para visualizar el resultado del conteo de palabras.

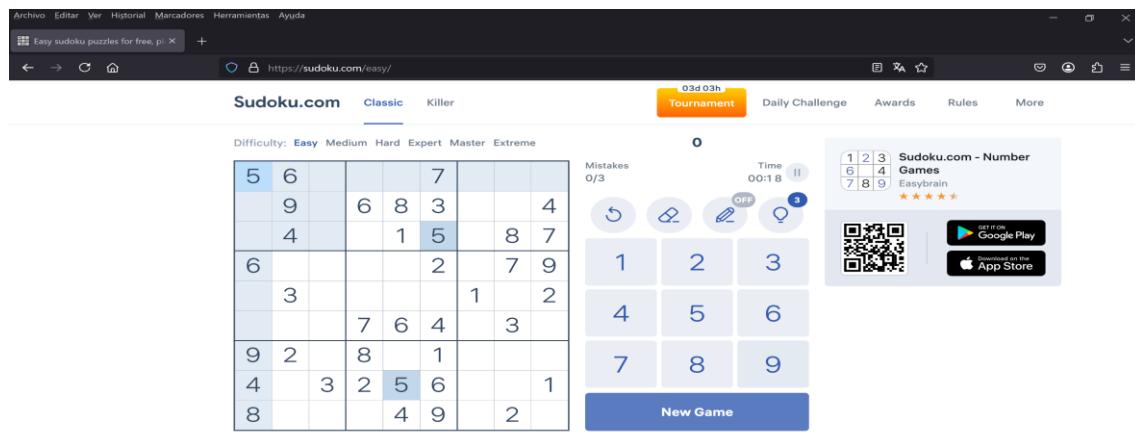
4.7.- Conclusión

Hemos ejecutado un proceso completo de **MapReduce en Hadoop** para contar palabras en un libro descargado desde **Project Gutenberg**. Utilizamos **HDFS** para el almacenamiento distribuido y

ejecutamos el ejemplo **wordcount** para procesar el archivo. Finalmente, copiamos los resultados a la máquina anfitriona para su análisis.

5.- Ejercicio 2: Sudoku

5.1.- Descargar dos sudokus



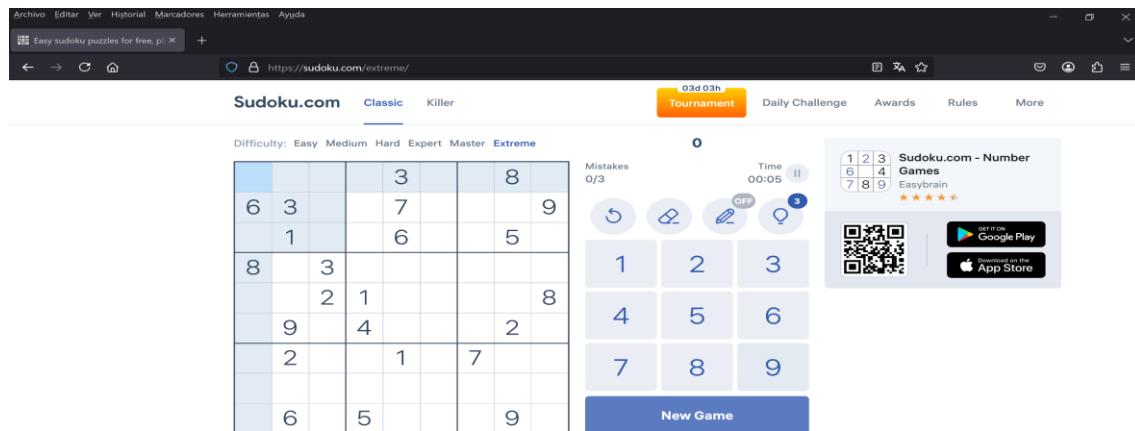
The screenshot shows a web browser displaying the Sudoku.com website. The URL is https://sudoku.com/easy/. The page features a navigation bar with links for Archivo, Editar, Ver, Historial, Marcadores, Herramientas, Ayuda, and a search bar. Below the navigation is a menu with Sudoku.com, Classic, Killer, Tournament (which is highlighted in orange), Daily Challenge, Awards, Rules, and More. A difficulty selector shows 'Easy' selected. To the left is a 9x9 Sudoku grid with some numbers filled in. To the right is a sidebar with a timer set at 00:18, a 'New Game' button, and links for 'Sudoku.com - Number Games' and download links for Google Play and the App Store.

Easy Sudoku for beginners

Easy Sudoku is characterized by the fact that cells contain more numbers than medium or hard ones. It makes this game suitable for beginners and those who have never played Sudoku before.

Sudoku is one of the most popular games to develop your intelligence. Surprisingly, it was invented in

En esta pantalla, se navega a la página web "<https://sudoku.com/easy/>" y se selecciona un sudoku de dificultad fácil.



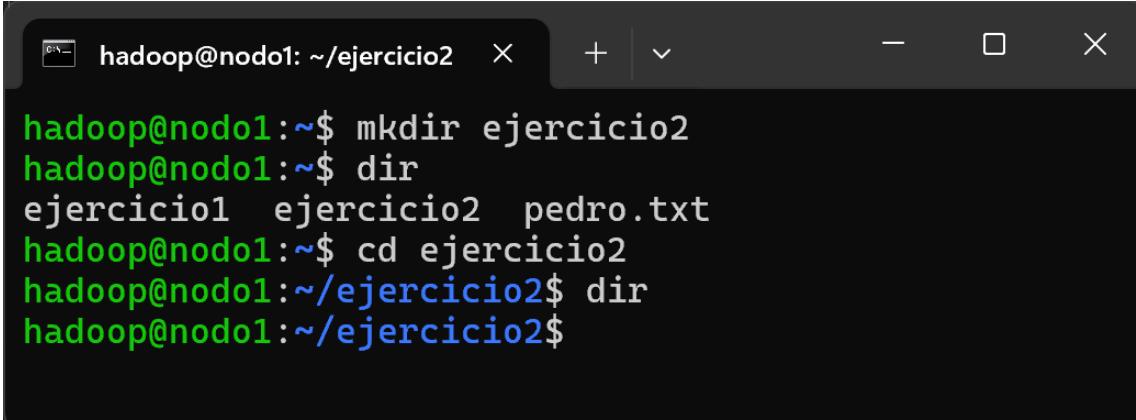
The screenshot shows the same web browser on the Sudoku.com website, but this time the 'Extreme' difficulty level is selected. The URL is https://sudoku.com/extreme/. The interface is identical to the easy version, with the 'Tournament' tab active. The extreme grid contains significantly fewer starting numbers than the easy grid, making it much harder to solve.

What is Extreme Sudoku?

Extreme Sudoku is the most challenging version of the 9x9 grid puzzle game for the real puzzle masters! It's not just about following the basic rules - you've got to be a puzzle-solving expert and use advanced strategies to crack it.

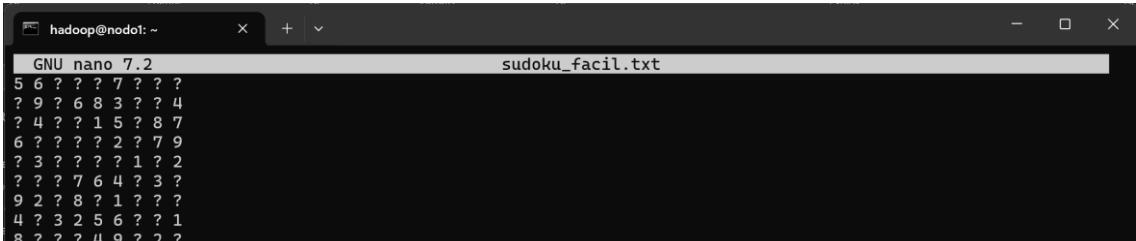
En esta pantalla, se navega a la página web "<https://sudoku.com/extreme/>" y se selecciona un sudoku de dificultad extremo.

5.2.- Crear los archivos en la máquina virtual



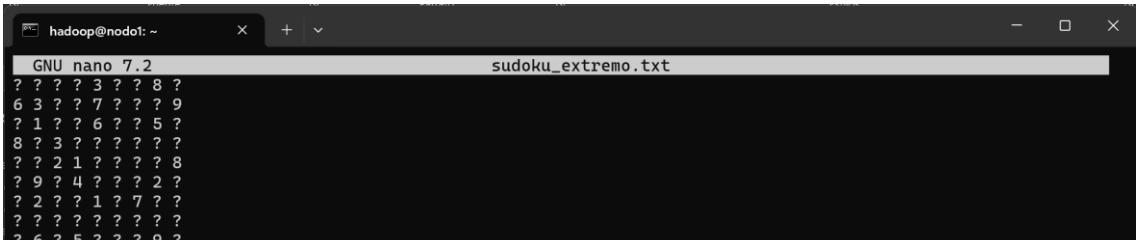
```
hadoop@nodo1: ~/ejercicio2 × + | v - □ ×
hadoop@nodo1:~$ mkdir ejercicio2
hadoop@nodo1:~$ dir
ejercicio1 ejercicio2 pedro.txt
hadoop@nodo1:~$ cd ejercicio2
hadoop@nodo1:~/ejercicio2$ dir
hadoop@nodo1:~/ejercicio2$
```

En esta pantalla, creamos el directorio "ejercicio2" con el comando "**mkdir ejercicio2**", pulsamos Intro para ejecutarlo, y luego verificamos su creación correctamente entrando en él con "**cd ejercicio2**" y listando su contenido con el comando "**dir**".



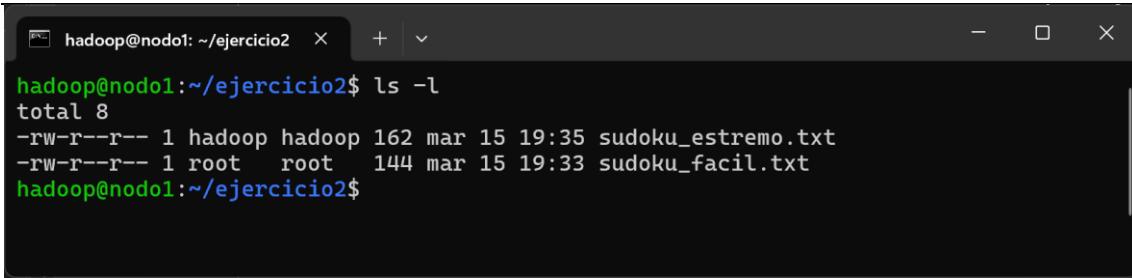
```
hadoop@nodo1: ~ × + | v - □ ×
GNU nano 7.2                                     sudoku_facil.txt
5 6 ? ? ? 7 ? ? ?
? 9 ? 6 8 3 ? ? 4
? 4 ? ? 1 5 ? 8 7
6 ? ? ? ? 2 ? ? 9
? 3 ? ? ? ? 1 ? 2
? ? ? 7 6 4 ? 3 ?
9 2 ? 8 ? 1 ? ? ?
4 ? 3 2 5 6 ? ? 1
8 ? ? ? 4 9 ? 2 ?
```

En esta pantalla, se crea el archivo *sudoku_facil.txt* utilizando el comando "**sudo nano sudoku_facil.txt**" y se ejecuta pulsando la tecla Intro.



```
hadoop@nodo1: ~ × + | v - □ ×
GNU nano 7.2                                     sudoku_extremo.txt
? ? ? ? 3 ? ? 8 ?
6 3 ? ? 7 ? ? ? 9
? 1 ? ? 6 ? ? 5 ?
8 ? 3 ? ? ? ? ?
? ? 2 1 ? ? ? ? 8
? 9 ? 4 ? ? ? 2 ?
? 2 ? ? 1 ? ? 7 ?
? ? ? ? ? ? ? ?
? 6 ? 5 ? ? ? 9 ?
```

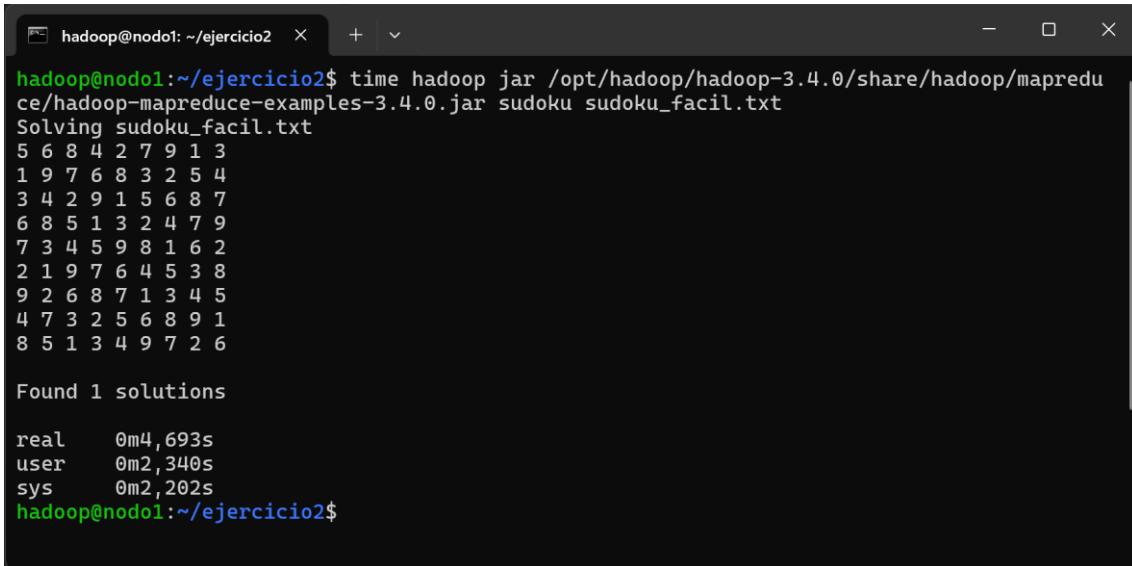
En esta pantalla, se crea el archivo *sudoku_extremo.txt* utilizando el comando "**sudo nano sudoku_extremo.txt**" y se ejecuta pulsando la tecla Intro.



```
hadoop@nodo1:~/ejercicio2$ ls -l
total 8
-rw-r--r-- 1 hadoop hadoop 162 mar 15 19:35 sudoku_estremo.txt
-rw-r--r-- 1 root   root   144 mar 15 19:33 sudoku_facil.txt
hadoop@nodo1:~/ejercicio2$
```

En esta pantalla, se utiliza el comando “**ls -l**” para verificar si los archivos se han creado correctamente, y se ejecuta pulsando la tecla Intro.

5.3.- Ejecutar el programa Sudoku en Hadoop



```
hadoop@nodo1:~/ejercicio2$ time hadoop jar /opt/hadoop/hadoop-3.4.0/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.4.0.jar sudoku sudoku_facil.txt
Solving sudoku_facil.txt
5 6 8 4 2 7 9 1 3
1 9 7 6 8 3 2 5 4
3 4 2 9 1 5 6 8 7
6 8 5 1 3 2 4 7 9
7 3 4 5 9 8 1 6 2
2 1 9 7 6 4 5 3 8
9 2 6 8 7 1 3 4 5
4 7 3 2 5 6 8 9 1
8 5 1 3 4 9 7 2 6

Found 1 solutions

real    0m4,693s
user    0m2,340s
sys     0m2,202s
hadoop@nodo1:~/ejercicio2$
```

En esta pantalla, se utiliza el comando “**time**” junto con la ejecución del programa de resolución de Sudoku en Hadoop para medir el tiempo que tarda en resolver el sudoku fácil, usando el comando completo: “**time hadoop jar /opt/hadoop/hadoop-3.4.0/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.4.0.jar sudoku sudoku_facil.txt**”.

```

hadoop@nodo1:~/ejercicio2$ time hadoop jar /opt/hadoop/hadoop-3.4.0/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.4.0.jar sudoku sudoku_extremo.txt
Solving sudoku_extremo.txt
5 7 4 9 3 1 6 8 2
6 3 8 2 7 5 4 1 9
2 1 9 8 6 4 3 5 7
8 5 3 6 2 9 1 7 4
7 4 2 1 5 3 9 6 8
1 9 6 4 8 7 5 2 3
9 2 5 3 1 8 7 4 6
4 8 1 7 9 6 2 3 5
3 6 7 5 4 2 8 9 1

Found 1 solutions

real    0m4,708s
user    0m2,321s
sys     0m2,183s
hadoop@nodo1:~/ejercicio2$
```

En esta pantalla, se utiliza el comando “**time**” junto con la ejecución del programa de resolución de Sudoku en Hadoop para medir el tiempo que tarda en resolver el sudoku extremo, usando el comando completo: “**time hadoop jar /opt/hadoop/hadoop-3.4.0/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.4.0.jar sudoku sudoku_extremo.txt**”.

5.4.- Comparar los tiempos de ejecución

En este apartado, vamos a comparar los tiempos de ejecución de los algoritmos que resuelven los dos Sudokus: uno de nivel **fácil** (**sudoku_facil.txt**) y otro de nivel **extremo** (**sudoku_extremo.txt**). El objetivo es observar las diferencias en los tiempos de ejecución de ambos y determinar si la complejidad del Sudoku afecta de manera significativa el tiempo necesario para resolverlo.

5.4.1.- Datos proporcionados

Sudoku fácil (sudoku_facil.txt):

```

real    0m4,693s
user    0m2,340s
sys     0m2,202s
hadoop@nodo1:~/ejercicio2$
```

Sudoku extremo (sudoku_extremo.txt):

```
real    0m4,708s
user    0m2,321s
sys     0m2,183s
hadoop@nodo1:~/ejercicio2$
```

5.4.2.- Explicación de los términos de tiempo

real: El tiempo total de ejecución, es decir, el tiempo que tarda el proceso desde el inicio hasta el fin.

user: El tiempo de **CPU** utilizado por el proceso en el espacio de usuario (cuando el proceso está ejecutando operaciones fuera del sistema operativo).

sys: El tiempo de **CPU** utilizado por el proceso en el espacio del núcleo (cuando el proceso está realizando operaciones relacionadas con el sistema operativo, como entrada/salida, gestión de memoria, etc.).

5.4.3.- Análisis de los tiempos de ejecución

1. Tiempo total de ejecución (**real**):

- El Sudoku fácil tardó **0m4.693s**.
- El Sudoku extremo tardó **0m4.708s**.

La diferencia en el tiempo total de ejecución es mínima, de **0.015 segundos**. Aunque en teoría el Sudoku extremo debería ser más difícil y consumir más tiempo, la diferencia no es significativa, lo que puede indicar que el algoritmo maneja ambos niveles de dificultad con una eficiencia similar.

2. Tiempo de **CPU** en modo usuario (**user**):

- El Sudoku fácil utilizó **0m2.340s** de **CPU** en modo usuario.
- El Sudoku extremo utilizó **0m2.321s** de **CPU** en modo usuario.

En este caso, el tiempo de **CPU** en modo usuario es casi el mismo para ambos casos. Esto sugiere que la carga

computacional del algoritmo es muy similar independientemente del nivel del Sudoku.

3. Tiempo de CPU en modo kernel (**sys**):

- El Sudoku fácil utilizó **0m2.025s** de CPU en modo kernel.
- El Sudoku extremo utilizó **0m2.135s** de CPU en modo kernel.

El tiempo en modo kernel es ligeramente mayor para el Sudoku extremo, lo que podría indicar que se realizaron más operaciones a nivel del sistema operativo, posiblemente relacionadas con la gestión de memoria o el acceso a archivos.

5.4.4.- Copiar los ficheros a la máquina anfitriona



```
C:\hadoop\Ejercicio2>scp hadoop@192.168.0.158:/home/hadoop/ejercicio2/*.* .
hadoop@192.168.0.158's password:
sudoku_extremo.txt                                         100% 162      17.6KB/s  00:00
sudoku_facil.txt                                           100% 162      17.6KB/s  00:00

C:\hadoop\Ejercicio2>
```

En esta pantalla, transferimos los archivos "sudoku_extremos.txt" y "sudoku_facil.txt" desde la máquina remota a la anfitriona utilizando el comando "**scp hadoop@192.168.0.158:/home/hadoop/ejercicio2/*.* .**", y luego pulsamos la tecla Intro para ejecutarlo.

5.4.5.- Conclusión

¿Es apreciable la diferencia en los tiempos de ejecución?

Basándonos en los resultados obtenidos, la diferencia entre los tiempos de ejecución es mínima (**0.015 segundos** en total). Aunque teóricamente el Sudoku extremo debería tomar más tiempo para resolverse, la pequeña variación observada podría deberse a:

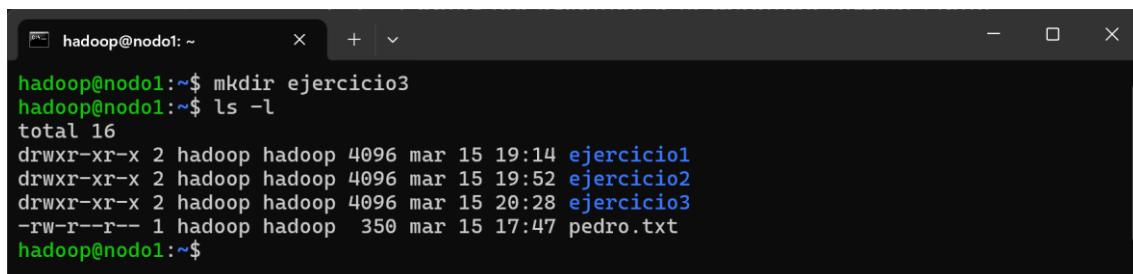
- La optimización del algoritmo.
- Las características de la implementación específica en Hadoop.
- El tamaño y la distribución de los datos en los archivos.

En resumen:

- En la mayoría de los casos, un Sudoku extremo debería tardar más en resolverse debido a su mayor complejidad.
- Sin embargo, en esta prueba, la diferencia de tiempo es insignificante, lo que podría indicar que el algoritmo está bien optimizado para ambos niveles de dificultad.

Recomendación futura:

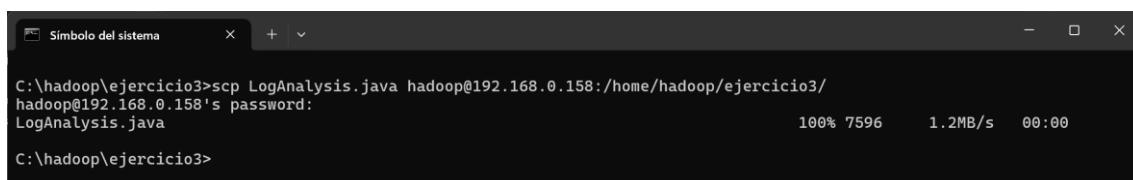
- Realizar más pruebas con Sudokus de distintos niveles de dificultad para verificar si esta tendencia se mantiene en diferentes escenarios.
- Evaluar la eficiencia del algoritmo con problemas más complejos para identificar posibles áreas de mejora.

6.- Ejercicio 3. Lectura de ficheros log**6.1.- Copiar los archivos a la máquina virtual Linux**


```

hadoop@nodo1: ~
hadoop@nodo1:~$ mkdir ejercicio3
hadoop@nodo1:~$ ls -l
total 16
drwxr-xr-x 2 hadoop hadoop 4096 mar 15 19:14 ejercicio1
drwxr-xr-x 2 hadoop hadoop 4096 mar 15 19:52 ejercicio2
drwxr-xr-x 2 hadoop hadoop 4096 mar 15 20:28 ejercicio3
-rw-r--r-- 1 hadoop hadoop 350 mar 15 17:47 pedro.txt
hadoop@nodo1:~$
```

En esta pantalla, creamos el directorio "**ejercicio3**" con el comando "**mkdir ejercicio3**", pulsamos Intro para ejecutarlo, y luego verificamos su creación correctamente listando los directorios con el comando "**ls -l**", pulsando Intro nuevamente para ejecutarlo.



```

Símbolo del sistema
C:\hadoop\ejercicio3>scp LogAnalysis.java hadoop@192.168.0.158:/home/hadoop/ejercicio3/
hadoop@192.168.0.158's password:                                     100% 7596      1.2MB/s   00:00
LogAnalysis.java
```

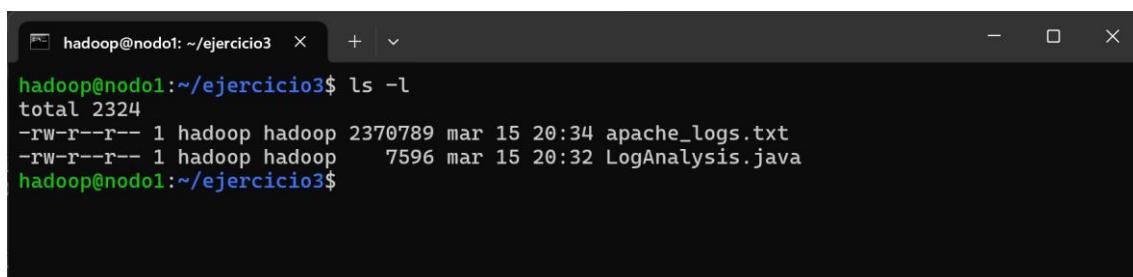
En esta pantalla, se copia el archivo "**LogAnalysis.java**" a la máquina virtual utilizando el comando "**scp LogAnalysis.java**

hadoop@192.168.0.158:/home/hadoop/ejercicio3/" y se ejecuta pulsando la tecla Intro.



```
C:\hadoop\ejercicio3>scp apache_logs.txt hadoop@192.168.0.158:/home/hadoop/ejercicio3  
hadoop@192.168.0.158's password:  
apache_logs.txt  
100% 2315KB 8.8MB/s 00:00  
C:\hadoop\ejercicio3>
```

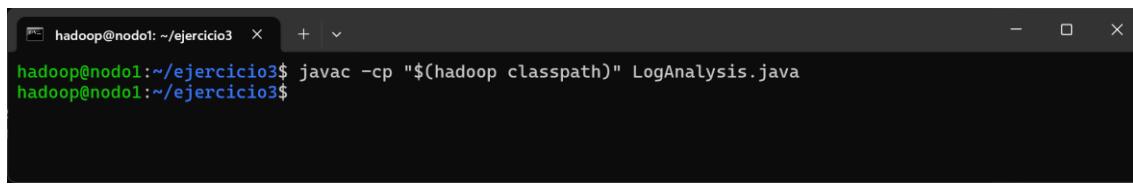
En esta pantalla, se copia el archivo "apache_logs.txt" a la máquina virtual utilizando el comando "scp apache_logs.txt hadoop@192.168.0.158:/home/hadoop/ejercicio3" y se ejecuta pulsando la tecla Intro.



```
hadoop@nodo1:~/ejercicio3 ~ + |  
hadoop@nodo1:~/ejercicio3$ ls -l  
total 2324  
-rw-r--r-- 1 hadoop hadoop 2370789 mar 15 20:34 apache_logs.txt  
-rw-r--r-- 1 hadoop hadoop 7596 mar 15 20:32 LogAnalysis.java  
hadoop@nodo1:~/ejercicio3$
```

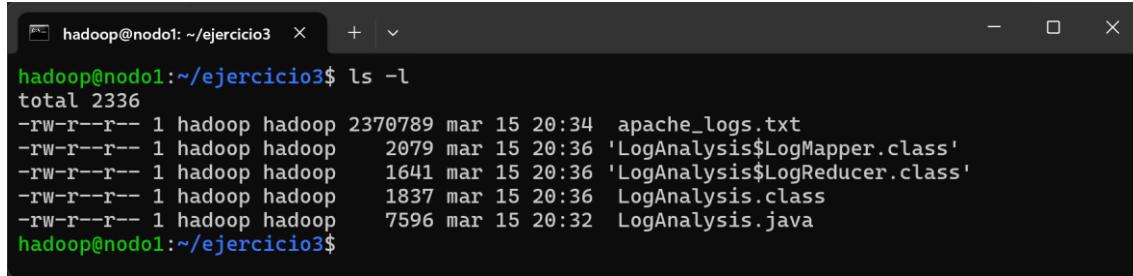
En esta pantalla, se verifica que los archivos se hayan copiado correctamente utilizando el comando "ls -l" y se ejecuta pulsando la tecla Intro.

6.2.- Compilar el archivo LogAnalysis.java



```
hadoop@nodo1:~/ejercicio3 ~ + |  
hadoop@nodo1:~/ejercicio3$ javac -cp "$(hadoop classpath)" LogAnalysis.java  
hadoop@nodo1:~/ejercicio3$
```

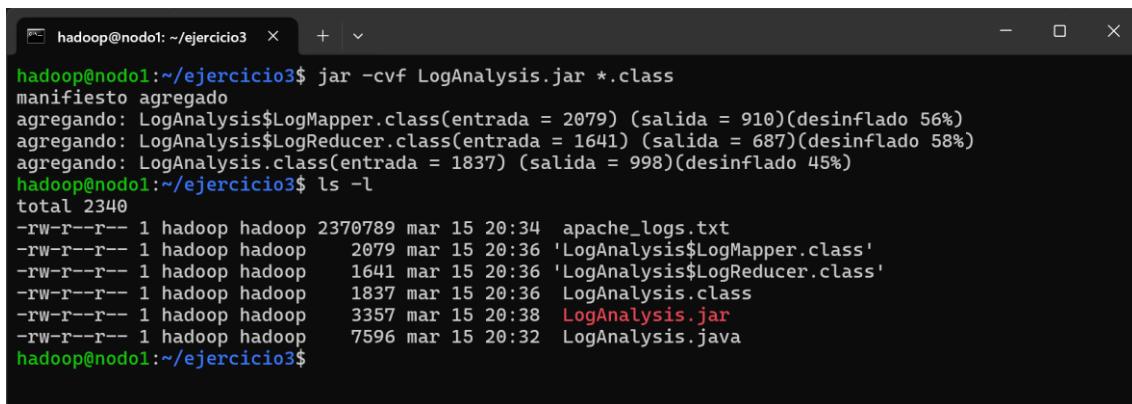
En esta pantalla, se compila el archivo "LogAnalysis.java" utilizando el comando "javac -cp "\$(hadoop classpath)" LogAnalysis.java" y se ejecuta pulsando la tecla Intro.



```
hadoop@nodo1:~/ejercicio3 ~ + |  
hadoop@nodo1:~/ejercicio3$ ls -l  
total 2336  
-rw-r--r-- 1 hadoop hadoop 2370789 mar 15 20:34 apache_logs.txt  
-rw-r--r-- 1 hadoop hadoop 2079 mar 15 20:36 'LogAnalysis$LogMapper.class'  
-rw-r--r-- 1 hadoop hadoop 1641 mar 15 20:36 'LogAnalysis$LogReducer.class'  
-rw-r--r-- 1 hadoop hadoop 1837 mar 15 20:36 LogAnalysis.class  
-rw-r--r-- 1 hadoop hadoop 7596 mar 15 20:32 LogAnalysis.java  
hadoop@nodo1:~/ejercicio3$
```

En esta pantalla, se verifica si la compilación ha sido exitosa utilizando el comando "ls -l" y se ejecuta pulsando la tecla Intro para comprobar que se han generado correctamente los tres archivos de clase esperados: **LogAnalysis.class** (la clase principal), **LogAnalysis\$Mapper.class** (la clase **Mapper**) y **LogAnalysis\$Reducer.class** (la clase **Reducer**).

6.3.- Empaquetar el código en un archivo JAR ejecutable

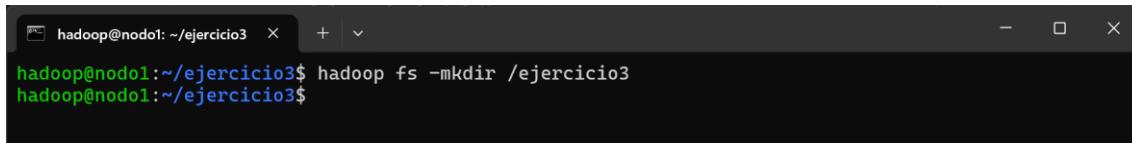


```

hadoop@nodo1:~/ejercicio3$ jar -cvf LogAnalysis.jar *.class
manifiesto agregado
agregando: LogAnalysis$LogMapper.class(entrada = 2079) (salida = 910)(desinflado 56%)
agregando: LogAnalysis$LogReducer.class(entrada = 1641) (salida = 687)(desinflado 58%)
agregando: LogAnalysis.class(entrada = 1837) (salida = 998)(desinflado 45%)
hadoop@nodo1:~/ejercicio3$ ls -l
total 2340
-rw-r--r-- 1 hadoop hadoop 2370789 mar 15 20:34 apache_logs.txt
-rw-r--r-- 1 hadoop hadoop 2079 mar 15 20:36 'LogAnalysis$LogMapper.class'
-rw-r--r-- 1 hadoop hadoop 1641 mar 15 20:36 'LogAnalysis$LogReducer.class'
-rw-r--r-- 1 hadoop hadoop 1837 mar 15 20:36 LogAnalysis.class
-rw-r--r-- 1 hadoop hadoop 3357 mar 15 20:38 LogAnalysis.jar
-rw-r--r-- 1 hadoop hadoop 7596 mar 15 20:32 LogAnalysis.java
hadoop@nodo1:~/ejercicio3$
```

En esta pantalla, se empaquetan todos los archivos .class generados en un archivo **JAR** ejecutable llamado "**LogAnalysis.jar**" utilizando el comando "**jar -cvf LogAnalysis.jar *.class**", lo que permitirá su posterior ejecución en Hadoop. Con el comando "**ls -l**" se verifica que el fichero "**logAnalysis.jar**" se ha creado correctamente.

6.4.- Crear un subdirectorio en HDFS y copiar el archivo de log



```

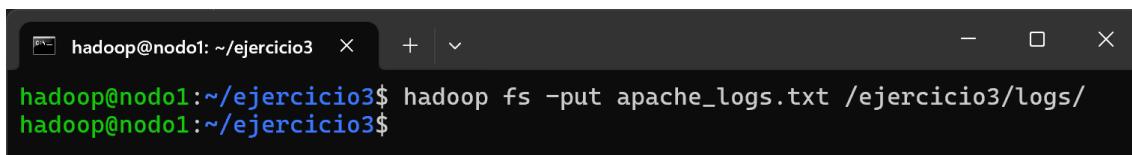
hadoop@nodo1:~/ejercicio3$ hadoop fs -mkdir /ejercicio3
hadoop@nodo1:~/ejercicio3$
```

En esta pantalla, creamos el directorio "**ejercicio3**" en **HDFS** utilizando el comando "**hadoop fs -mkdir /ejercicio3**", y luego pulsamos la tecla Intro para ejecutarlo.



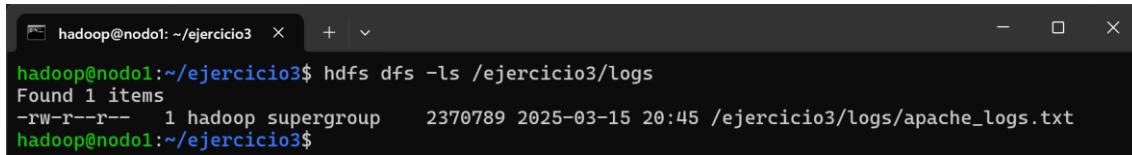
```
hadoop@nodo1:~/ejercicio3$ hadoop fs -mkdir /ejercicio3/logs
hadoop@nodo1:~/ejercicio3$
```

En esta pantalla, se crea el directorio "["/ejercicio3/logs"](#) en [HDFS](#) utilizando el comando "["hadoop fs -mkdir /ejercicio3/logs"](#)" y se ejecuta pulsando la tecla Intro.



```
hadoop@nodo1:~/ejercicio3$ hadoop fs -put apache_logs.txt /ejercicio3/logs/
hadoop@nodo1:~/ejercicio3$
```

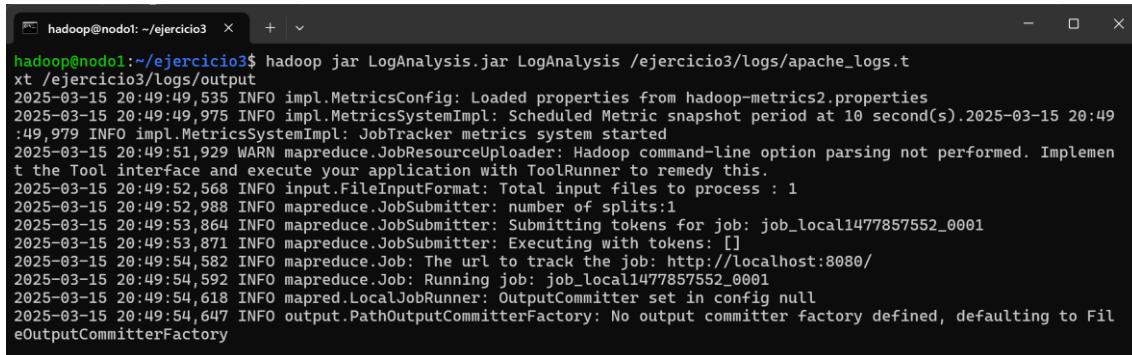
En esta pantalla, se copia el archivo "["apache_logs.txt"](#)" al directorio "["/ejercicio3/logs"](#) en [HDFS](#) utilizando el comando "["hadoop fs -put apache_logs.txt /ejercicio3/logs/"](#)" y se ejecuta pulsando la tecla Intro.



```
hadoop@nodo1:~/ejercicio3$ hdfs dfs -ls /ejercicio3/logs
Found 1 items
-rw-r--r-- 1 hadoop supergroup 2370789 2025-03-15 20:45 /ejercicio3/logs/apache_logs.txt
hadoop@nodo1:~/ejercicio3$
```

En esta pantalla, se verifica si el fichero se ha copiado correctamente utilizando el comando "["hdfs dfs -ls /ejercicio3/logs"](#)" y se ejecuta pulsando la tecla Intro.

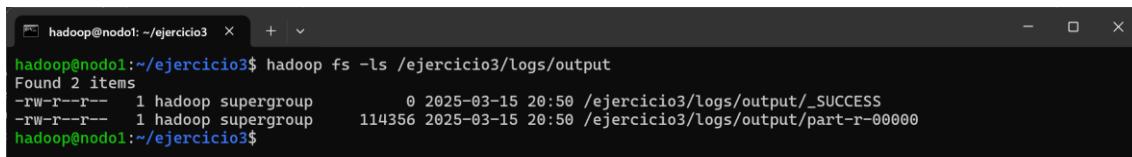
6.5.- Ejecutar el programa LogAnalysis con el archivo de log



```
hadoop@nodo1:~/ejercicio3$ hadoop jar LogAnalysis.jar LogAnalysis /ejercicio3/logs/apache_logs.txt /ejercicio3/logs/output
2025-03-15 20:49:49.535 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2025-03-15 20:49:49.975 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2025-03-15 20:49:49.979 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2025-03-15 20:49:51.929 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2025-03-15 20:49:52.568 INFO input.FileInputFormat: Total input files to process : 1
2025-03-15 20:49:52.988 INFO mapreduce.JobSubmitter: number of splits:1
2025-03-15 20:49:53.864 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1477857552_0001
2025-03-15 20:49:53.871 INFO mapreduce.JobSubmitter: Executing with tokens: []
2025-03-15 20:49:54.582 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2025-03-15 20:49:54.592 INFO mapreduce.Job: Running job: job_local1477857552_0001
2025-03-15 20:49:54.618 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2025-03-15 20:49:54.647 INFO output.PathOutputCommitterFactory: No output committer factory defined, defaulting to FileOutputCommitterFactory
```

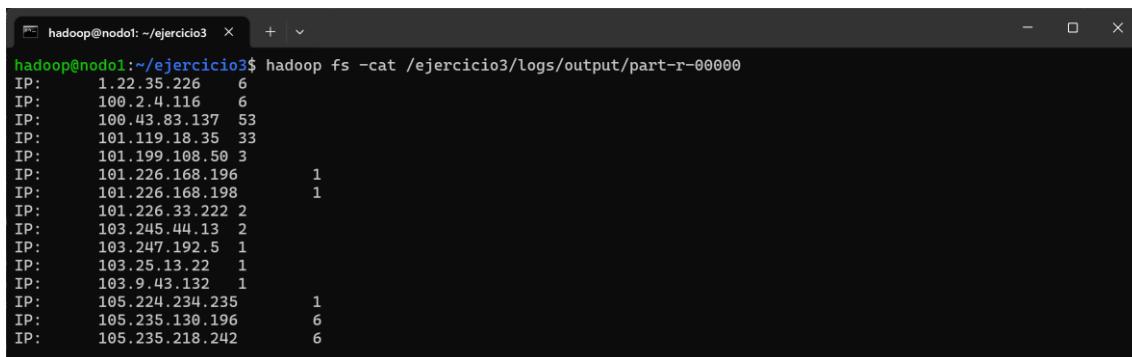
En esta pantalla, se ejecuta el archivo JAR empaquetado en Hadoop utilizando el comando "`hadoop jar LogAnalysis.jar LogAnalysis /ejercicio3/logs/apache_logs.txt /ejercicio3/logs/output`", que procesará el archivo de logs ubicado en "`/ejercicio3/logs/apache_logs.txt`" y almacenará los resultados en el directorio "`/ejercicio3/logs/output`" en **HDFS**.

6.6.- Ver vista previa de los resultados



```
hadoop@nodo1:~/ejercicio3$ hadoop fs -ls /ejercicio3/logs/output
Found 2 items
-rw-r--r-- 1 hadoop supergroup          0 2025-03-15 20:50 /ejercicio3/logs/output/_SUCCESS
-rw-r--r-- 1 hadoop supergroup    114356 2025-03-15 20:50 /ejercicio3/logs/output/part-r-00000
hadoop@nodo1:~/ejercicio3$
```

En esta pantalla, una vez finalizada la ejecución del trabajo MapReduce, se utiliza el comando "`hadoop fs -ls /ejercicio3/logs/output`" para verificar el contenido del directorio de salida y comprobar la presencia de los archivos generados, como **part-r-00000**, que indican el éxito del proceso.



```
hadoop@nodo1:~/ejercicio3$ hadoop fs -cat /ejercicio3/logs/output/part-r-00000
IP: 1.22.35.226 6
IP: 100.2.4.116 6
IP: 100.43.83.137 53
IP: 101.119.18.35 33
IP: 101.199.108.50 3
IP: 101.226.168.196 1
IP: 101.226.168.198 1
IP: 101.226.33.222 2
IP: 103.245.44.13 2
IP: 103.247.192.5 1
IP: 103.25.13.22 1
IP: 103.9.43.132 1
IP: 105.224.234.235 1
IP: 105.235.130.196 6
IP: 105.235.218.242 6
```

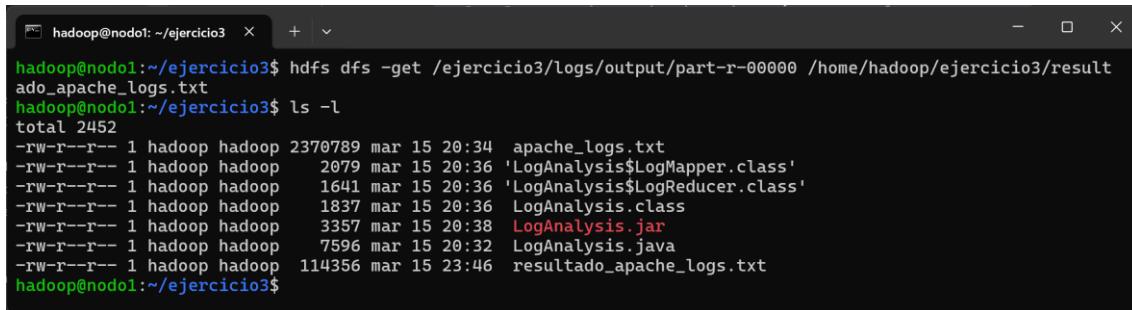
En esta pantalla, se utiliza el comando "`hadoop fs -cat /ejercicio3/logs/output/part-r-00000`" para obtener una vista previa del contenido del archivo de resultados generado, mostrando las direcciones IP y los recursos solicitados junto con sus respectivos conteos.

6.7.- Expresión Regular para extraer información



```
String logPattern = "(\\s+) - - \\[.*?\\] \"(\\s+)\\s(\\s+)\\s.*\"";
Pattern pattern = Pattern.compile(logPattern);
```

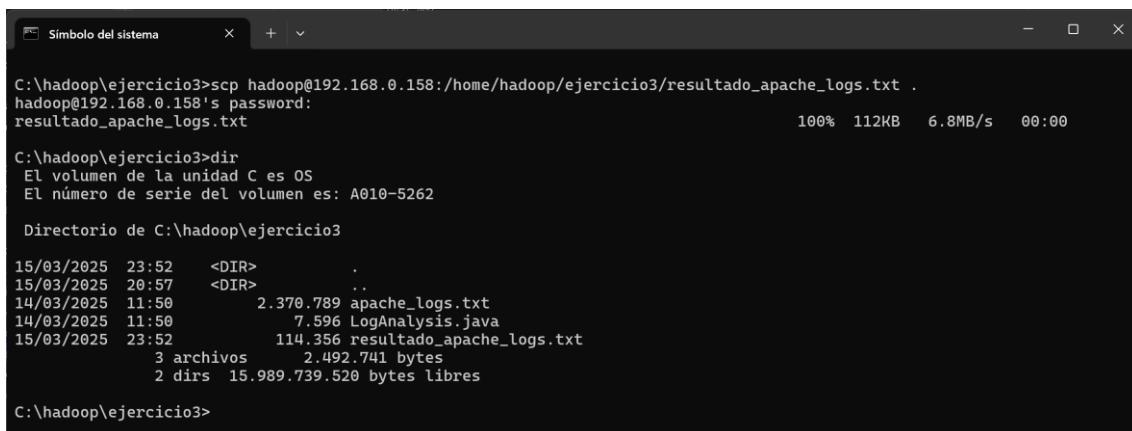
6.8.- Copiar el resultado a la máquina anfitriona



```

hadoop@nodo1:~/ejercicio3$ hdfs dfs -get /ejercicio3/logs/output/part-r-00000 /home/hadoop/ejercicio3/result
ado_apache_logs.txt
hadoop@nodo1:~/ejercicio3$ ls -l
total 2452
-rw-r--r-- 1 hadoop hadoop 2370789 mar 15 20:34 apache_logs.txt
-rw-r--r-- 1 hadoop hadoop 2079 mar 15 20:36 'LogAnalysis$LogMapper.class'
-rw-r--r-- 1 hadoop hadoop 1641 mar 15 20:36 'LogAnalysis$LogReducer.class'
-rw-r--r-- 1 hadoop hadoop 1837 mar 15 20:36 LogAnalysis.class
-rw-r--r-- 1 hadoop hadoop 3357 mar 15 20:38 LogAnalysis.jar
-rw-r--r-- 1 hadoop hadoop 7596 mar 15 20:32 LogAnalysis.java
-rw-r--r-- 1 hadoop hadoop 114356 mar 15 23:46 resultado_apache_logs.txt
hadoop@nodo1:~/ejercicio3$
```

En esta pantalla, se procede a copiar los resultados desde HDFS a la máquina virtual, se ejecuta el comando “`hdfs dfs -get /ejercicio3/logs/output/part-r-00000`” para transferir el archivo `part-r-00000` a un archivo local llamado `resultado_apache_logs.txt`, se verifica que el archivo `resultado_apache_logs.txt` se ha copiado correctamente a la máquina virtual utilizando el comando “`dir`”.



```

Símbolo del sistema
C:\hadoop\ejercicio3>scp hadoop@192.168.0.158:/home/hadoop/ejercicio3/resultado_apache_logs.txt .
hadoop@192.168.0.158's password:                                                 100% 112KB 6.8MB/s 00:00
resultado_apache_logs.txt

C:\hadoop\ejercicio3>dir
El volumen de la unidad C es OS
El número de serie del volumen es: A010-5262

Directorio de C:\hadoop\ejercicio3

15/03/2025 23:52    <DIR>      .
15/03/2025 20:57    <DIR>      ..
14/03/2025 11:50        2.370.789 apache_logs.txt
14/03/2025 11:50        7.596 LogAnalysis.java
15/03/2025 23:52        114.356 resultado_apache_logs.txt
            3 archivos     2.492.741 bytes
            2 dirs   15.989.739.520 bytes libres

C:\hadoop\ejercicio3>
```

En esta pantalla, se utiliza el comando “`scp hadoop@192.168.0.158:/home/hadoop/ejercicio3/resultado_apache_logs.txt .`” para copiar el archivo `resultado_apache_logs.txt` desde la máquina virtual a la máquina anfitriona, y luego se ejecuta el comando “`dir`” para verificar que la copia se ha realizado correctamente.

6.9.- Conclusión

En el análisis de logs, se utiliza una expresión regular implementada con la `clase Pattern` de `Java` para extraer la dirección `IP`

Actividad evaluable 2.1.- Hadoop: MapReduce y HDFS

del cliente y los recursos solicitados, permitiendo así procesar eficientemente la información contenida en los archivos de registro.

Esta expresión regular extrae la dirección **IP**, el método **HTTP** y el recurso solicitado de los **logs**, permitiendo al **Mapper** procesar estos elementos mientras el **Reducer** se encarga de contar las ocurrencias de cada uno en el análisis de logs mediante MapReduce.

En resumen, el proceso completo para analizar **logs** de **Apache** con **Hadoop MapReduce** incluye: copiar archivos a la máquina virtual, compilar el código **Java** con la **classpath** de **Hadoop**, empaquetarlo en un **JAR**, crear un directorio en **HDFS** para los **logs**, ejecutar el programa en **Hadoop**, y finalmente verificar los resultados generados en el directorio de salida de **HDFS**.

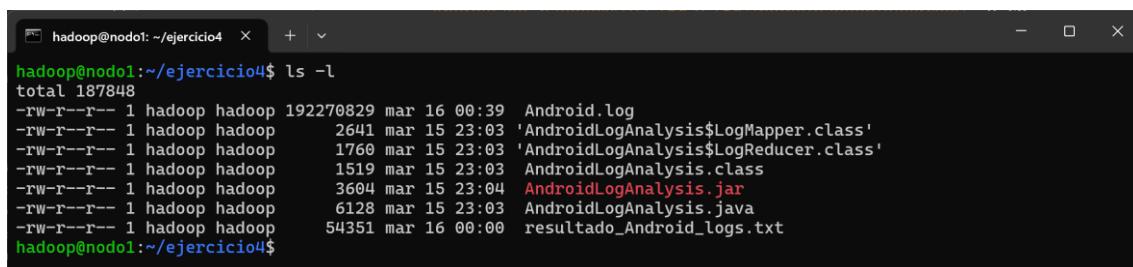
7.- Ejercicio 4. Lectura de otro tipo de fichero log

7.1.- Copiar el archivo de log a la máquina virtual



```
C:\hadoop\Ejercicio4>scp Android.log hadoop@192.168.0.158:/home/hadoop/ejercicio4
hadoop@192.168.0.158's password:
Android.log                                         100%  183MB  11.7MB/s   00:15
C:\hadoop\Ejercicio4>
```

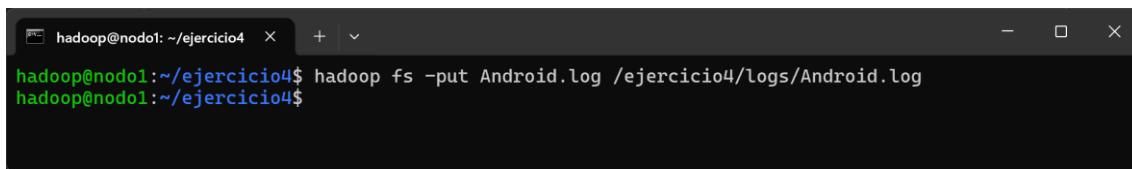
En esta pantalla, se copia el archivo "Android.log" desde la máquina anfitriona a la máquina virtual utilizando el comando "**scp Android.log hadoop@192.168.0.158:/home/hadoop/ejercicio4/**" y se ejecuta pulsando la tecla Intro.



```
hadoop@nodo1:~/ejercicio4$ ls -l
total 187848
-rw-r--r-- 1 hadoop hadoop 192270829 mar 16 00:39 Android.log
-rw-r--r-- 1 hadoop hadoop      2641 mar 15 23:03 'AndroidLogAnalysis$LogMapper.class'
-rw-r--r-- 1 hadoop hadoop     1760 mar 15 23:03 'AndroidLogAnalysis$LogReducer.class'
-rw-r--r-- 1 hadoop hadoop     1519 mar 15 23:03 AndroidLogAnalysis.class
-rw-r--r-- 1 hadoop hadoop     3604 mar 15 23:04 AndroidLogAnalysis.jar
-rw-r--r-- 1 hadoop hadoop     6128 mar 15 23:03 AndroidLogAnalysis.java
-rw-r--r-- 1 hadoop hadoop  54351 mar 16 00:00 resultado_Android_logs.txt
hadoop@nodo1:~/ejercicio4$
```

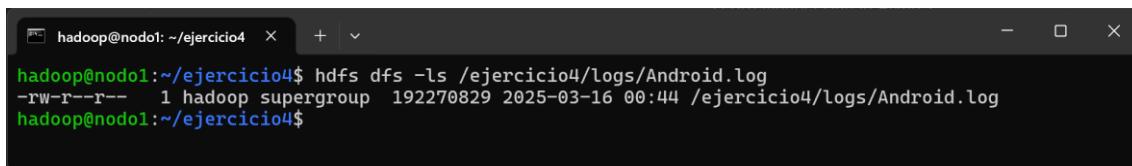
Actividad evaluable 2.1.- Hadoop: MapReduce y HDFS

En esta pantalla, se verifica si el archivo se ha copiado correctamente utilizando el comando "**ls -l**" y se ejecuta pulsando la tecla Intro.



```
hadoop@nodo1:~/ejercicio4$ hadoop fs -put Android.log /ejercicio4/logs/Android.log
hadoop@nodo1:~/ejercicio4$
```

En esta pantalla, se copia el archivo "**Android.log**" al sistema de archivos de Hadoop (**HDFS**) utilizando el comando "**hadoop fs -put Android.log /ejercicio4/logs/Android.log**" después de haber establecido conexión con la máquina virtual.

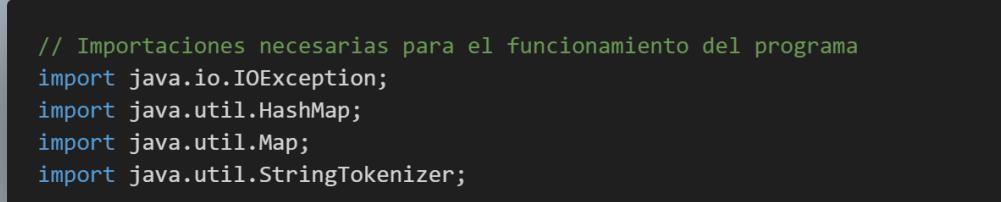


```
hadoop@nodo1:~/ejercicio4$ hdfs dfs -ls /ejercicio4/logs/Android.log
-rw-r--r-- 1 hadoop supergroup 192270829 2025-03-16 00:44 /ejercicio4/logs/Android.log
hadoop@nodo1:~/ejercicio4$
```

En esta pantalla, se procede a verificar la correcta copia del archivo a HDFS, se utiliza el comando "**hdfs dfs -ls /ejercicio4/logs/Android.log**", que mostrará la presencia del archivo en el directorio especificado junto con su tamaño, confirmando así el éxito de la operación.

7.2.- Crear un programa Java utilizando MapReduce

7.2.1.- Explicación del código del fichero AndroidLogAnalysis.java



```
// Importaciones necesarias para el funcionamiento del programa
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import java.util.StringTokenizer;
```

Este código importa clases específicas de **Java** para manejar excepciones de entrada/salida (**IOException**), utilizar estructuras de

datos como **HashMap** y **Map**, y procesar cadenas de texto con **StringTokenizer**.

```
// Importaciones de las clases de Hadoop necesarias para MapReduce
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
```

Este código importa las clases necesarias de Hadoop para configurar y ejecutar un trabajo **MapReduce**, incluyendo la configuración del trabajo, la definición de las clases **Mapper** y **Reducer**, y la especificación de los formatos de entrada y salida en el sistema de archivos distribuido **HDFS**.

```
public class AndroidLogAnalysis {  
    /**  
     * Clase Mapper personalizada para procesar las líneas de log de Android.  
     * Extiende de Mapper<Object, Text, Text, IntWritable>.  
     */  
    public static class LogMapper extends Mapper<Object, Text, Text, IntWritable> {  
        // Constante para representar el conteo de una ocurrencia  
        private final static IntWritable one = new IntWritable(1);  
        // Objeto Text reutilizable para la salida del mapper  
        private Text word = new Text();  
  
        // Mapa para almacenar los niveles de log y sus significados  
        private static final Map<String, String> loglevels = new HashMap<>();  
        static {  
            loglevels.put("", "Verbose");  
            loglevels.put("D", "Debug");  
            loglevels.put("I", "Info");  
            loglevels.put("W", "Warning");  
            loglevels.put("E", "Error");  
            loglevels.put("A", "Assert");  
            loglevels.put("F", "Fatal");  
        }  
  
        /**  
         * Método map que procesa cada línea del archivo de entrada.  
         * @param key Clave de entrada (no utilizada en este caso)  
         * @param value Valor de entrada (línea de texto del log)  
         * @param context Contexto para escribir la salida del mapper  
         */  
        @Override  
        public void map(Object key, Text value, Context context) throws IOException, InterruptedException {  
            String line = value.toString();  
            StringTokenizer tokenizer = new StringTokenizer(line);  
  
            // Verificar si la linea tiene al menos 5 tokens (formato válido)  
            if (tokenizer.countTokens() < 5) {  
                return; // Línea inválida, se ignora  
            }  
  
            // Saltar los primeros 4 tokens que no son relevantes para este análisis  
            tokenizer.nextToken(); // Fecha  
            tokenizer.nextToken(); // Hora  
            tokenizer.nextToken(); // PID (Process ID)  
            tokenizer.nextToken(); // TID (Thread ID)  
  
            // Obtener el nivel de log  
            String logLevel = tokenizer.nextToken();  
  
            // Obtener el significado del nivel de log o usar "otros" si no está en el mapa  
            String logMeaning = loglevels.getOrDefault(logLevel, "otros");  
            context.write(new Text(logLevel + " (" + logMeaning + ")", one);  
  
            // Obtener la etiqueta del log (si existe)  
            if (tokenizer.hasMoreTokens()) {  
                String logTag = tokenizer.nextToken();  
                context.write(new Text("Etiqueta: " + logTag), one);  
            }  
        }  
    }  
}
```

Este código define una clase **AndroidLogAnalysis** que incluye un **Mapper** personalizado para un trabajo **MapReduce** en **Hadoop**, diseñado para analizar líneas de logs de Android. El **Mapper** procesa cada línea del archivo de entrada, identifica el nivel de log (como **"Verbose"**, **"Debug"**, **"Info"**, etc.) y su significado utilizando un mapa predefinido, y cuenta las ocurrencias de cada nivel y etiqueta de log, escribiendo los resultados en el contexto para su posterior procesamiento.

```
/*
 * Clase Reducer personalizada para sumar las ocurrencias de cada nivel de log y etiqueta.
 * Extiende de Reducer<Text, IntWritable, Text, IntWritable>.
 */
public static class LogReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

    private IntWritable result = new IntWritable();

    /**
     * Método reduce que suma las ocurrencias para cada clave.
     * @param key Clave (nivel de log o etiqueta)
     * @param values Iterable de valores (conteos) asociados a la clave
     * @param context Contexto para escribir la salida del reducer
     */
    @Override
    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);
    }
}
```

Este código define una clase **Reducer** personalizada llamada **LogReducer** que suma las ocurrencias de cada nivel de log y etiqueta procesados por el **Mapper**, agregando los conteos para cada clave y escribiendo el resultado final en el contexto de salida del trabajo **MapReduce**.

```
/*
 * Método principal para configurar y ejecutar el trabajo MapReduce.
 * @param args Argumentos de línea de comando (se esperan dos: ruta de entrada y ruta de salida)
 */
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "android log analysis");
    job.setJarByClass(AndroidLogAnalysis.class);
    job.setMapperClass(LogMapper.class);
    job.setCombinerClass(LogReducer.class);
    job.setReducerClass(LogReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```

Este código define el método principal de la clase **AndroidLogAnalysis**, que configura y ejecuta un trabajo MapReduce en Hadoop para analizar logs de **Android**, especificando el **Mapper**,

Reducer, clases de entrada/salida, y las rutas de entrada y salida proporcionadas como argumentos desde la línea de comandos.

7.2.2.- Pasar el fichero AndroidLogAnalysis.java a la máquina virtual

```
C:\hadoop\Ejercicio4>scp AndroidLogAnalysis.java hadoop@192.168.0.158:/home/hadoop/ejercicio4/
hadoop@192.168.0.158's password:
AndroidLogAnalysis.java
100% 6128      1.2MB/s  00:00

C:\hadoop\Ejercicio4>
```

En esta pantalla, se procede a ejecutar el comando "**scp AndroidLogAnalysis.java hadoop@192.168.0.158:/home/hadoop/ejercicio4/**" para copiar el fichero **AndroidLogAnalysis.java** a la máquina virtual en el directorio **/home/hadoop/ejercicio4** del usuario **hadoop**.

7.2.3.- Pasar el fichero Android.log a la máquina virtual

```
C:\hadoop\Ejercicio4>scp Android.log hadoop@192.168.0.158:/home/hadoop/ejercicio4
hadoop@192.168.0.158's password:
Android.log
100% 183MB   8.7MB/s  00:20

C:\hadoop\Ejercicio4>
```

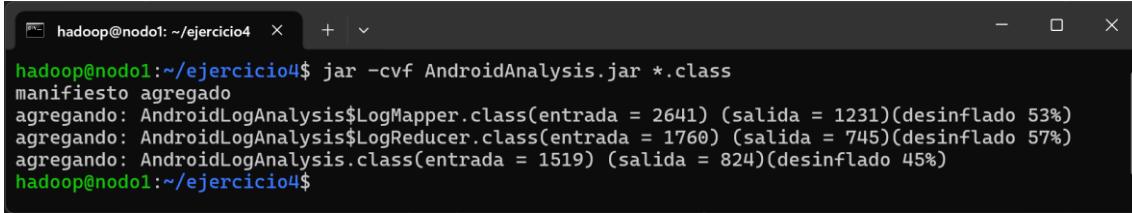
En esta pantalla, se procede con el comando "**scp Android.log hadoop@192.168.0.158:/home/hadoop/ejercicio4**" a transferir el archivo "**Android.log**" desde el sistema local al directorio "**/home/hadoop/ejercicio4**" en el servidor remoto con la dirección IP **192.168.0.158**, utilizando el usuario "**Hadoop**" para la conexión.

7.3.- Compilación del programa AndroidLogAnalysis.java

```
hadoop@nodo1:~/ejercicio4$ javac -classpath 'hadoop classpath' -d . AndroidLogAnalysis.java
hadoop@nodo1:~/ejercicio4$
```

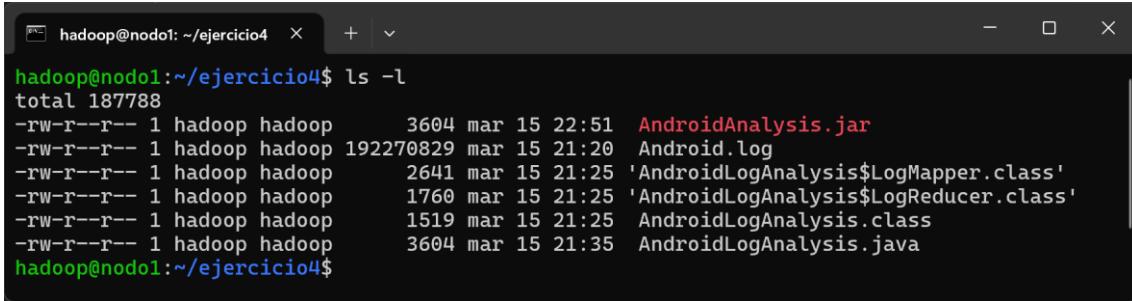
En esta pantalla, compilamos el archivo "**AndroidLogAnalysis.java**" utilizando el comando "**javac -classpath 'hadoop classpath' -d . AndroidLogAnalysis.java**", que emplea el classpath de **Hadoop** para generar los archivos de clase "**AndroidLogAnalysis\$LogMapper.class**",

"AndroidLogAnalysis\$LogReducer.class" y "AndroidLogAnalysis.class" en el directorio actual.



```
hadoop@nodo1:~/ejercicio4$ jar -cvf AndroidAnalysis.jar *.class
manifiesto agregado
agregando: AndroidLogAnalysis$LogMapper.class(entrada = 2641) (salida = 1231)(desinflado 53%)
agregando: AndroidLogAnalysis$LogReducer.class(entrada = 1760) (salida = 745)(desinflado 57%)
agregando: AndroidLogAnalysis.class(entrada = 1519) (salida = 824)(desinflado 45%)
hadoop@nodo1:~/ejercicio4$
```

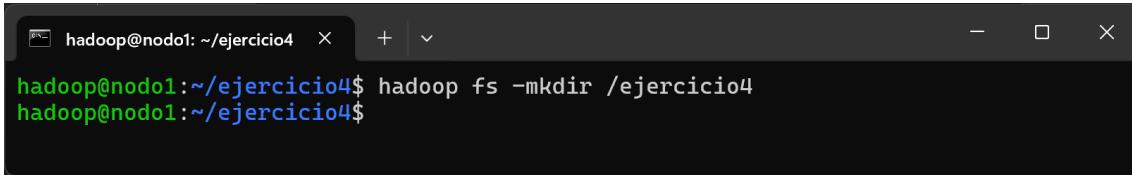
En esta pantalla, se procede con el comando "**jar -cvf AndroidLogAnalysis.java *.class**" crea un archivo JAR llamado "**AndroidLogAnalysis.java**" que incluye todos los archivos ***.class** y subdirectorios del directorio actual, manteniendo la estructura de directorios original.



```
hadoop@nodo1:~/ejercicio4$ ls -l
total 187788
-rw-r--r-- 1 hadoop hadoop 3604 mar 15 22:51 AndroidAnalysis.jar
-rw-r--r-- 1 hadoop hadoop 192270829 mar 15 21:20 Android.log
-rw-r--r-- 1 hadoop hadoop 2641 mar 15 21:25 'AndroidLogAnalysis$LogMapper.class'
-rw-r--r-- 1 hadoop hadoop 1760 mar 15 21:25 'AndroidLogAnalysis$LogReducer.class'
-rw-r--r-- 1 hadoop hadoop 1519 mar 15 21:25 AndroidLogAnalysis.class
-rw-r--r-- 1 hadoop hadoop 3604 mar 15 21:35 AndroidLogAnalysis.java
hadoop@nodo1:~/ejercicio4$
```

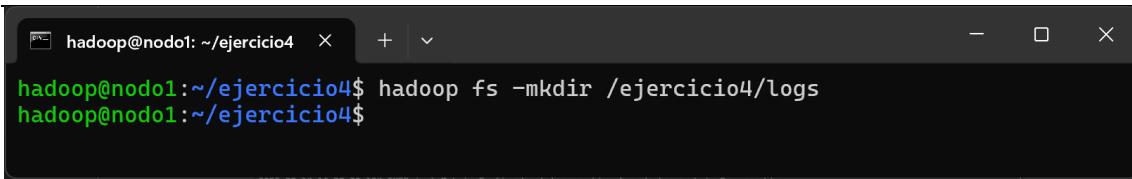
En esta pantalla, verificamos que el archivo "**AndroidAnalysis.jar**" se ha creado correctamente ejecutando el comando "**ls -l**", y pulsamos la tecla Intro para mostrar el resultado.

7.4.- Ejecutar el trabajo MapReduce



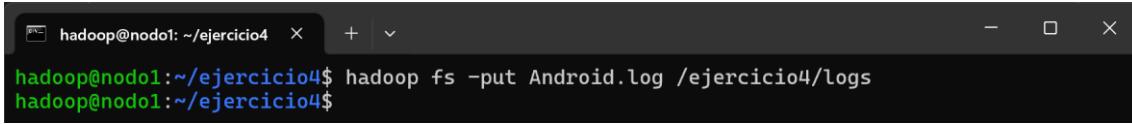
```
hadoop@nodo1:~/ejercicio4$ hadoop fs -mkdir /ejercicio4
hadoop@nodo1:~/ejercicio4$
```

En esta pantalla, creamos el directorio "**ejercicio4**" en **HDFS** utilizando el comando "**hadoop fs -mkdir /ejercicio4**", y luego pulsamos la tecla Intro para ejecutarlo.



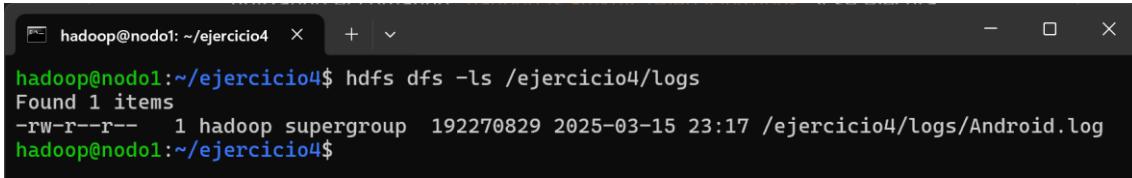
```
hadoop@nodo1:~/ejercicio4$ hadoop fs -mkdir /ejercicio4/logs
hadoop@nodo1:~/ejercicio4$
```

En esta pantalla, se crea el directorio "["/ejercicio4/logs"](#) en [HDFS](#) utilizando el comando "[hadoop fs -mkdir /ejercicio4/logs](#)" y se ejecuta pulsando la tecla Intro.



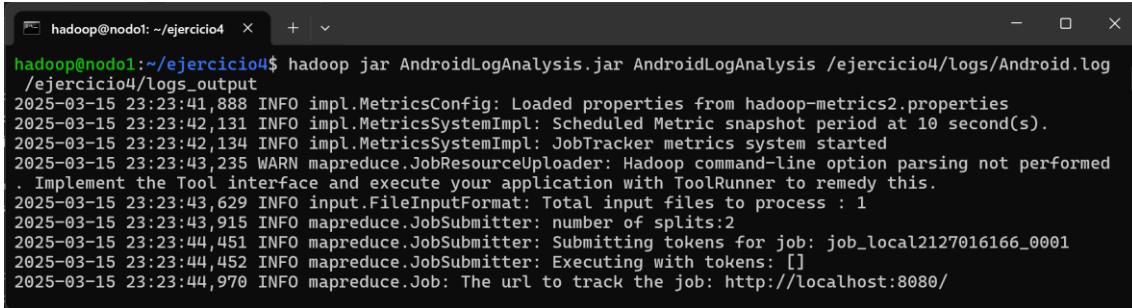
```
hadoop@nodo1:~/ejercicio4$ hadoop fs -put Android.log /ejercicio4/logs
hadoop@nodo1:~/ejercicio4$
```

En esta pantalla, se copia el archivo "[Android.log](#)" al directorio "["/ejercicio3/logs"](#) en [HDFS](#) utilizando el comando "[hadoop fs -put Android.log /ejercicio4/logs/](#)" y se ejecuta pulsando la tecla Intro.



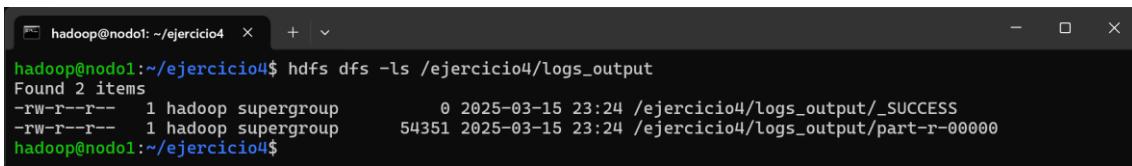
```
hadoop@nodo1:~/ejercicio4$ hdfs dfs -ls /ejercicio4/logs
Found 1 items
-rw-r--r-- 1 hadoop supergroup 192270829 2025-03-15 23:17 /ejercicio4/logs/Android.log
hadoop@nodo1:~/ejercicio4$
```

En esta pantalla, se verifica si el fichero se ha copiado correctamente utilizando el comando "[hdfs dfs -ls /ejercicio4/logs](#)" y se ejecuta pulsando la tecla Intro.



```
hadoop@nodo1:~/ejercicio4$ hadoop jar AndroidLogAnalysis.jar AndroidLogAnalysis /ejercicio4/logs/Android.log /ejercicio4/logs_output
2025-03-15 23:23:41,888 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2025-03-15 23:23:42,131 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2025-03-15 23:23:42,134 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2025-03-15 23:23:43,235 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed
. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2025-03-15 23:23:43,629 INFO input.FileInputFormat: Total input files to process : 1
2025-03-15 23:23:43,915 INFO mapreduce.JobSubmitter: number of splits:2
2025-03-15 23:23:44,451 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local2127016166_0001
2025-03-15 23:23:44,452 INFO mapreduce.JobSubmitter: Executing with tokens: []
2025-03-15 23:23:44,970 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
```

En esta pantalla, se procede a ejecutar el comando "[hadoop jar AndroidLogAnalysis.jar /ejercicio4/logs/Android.log /ejercicio4/logs_output](#)" y se presiona la tecla Intro para procesar el archivo "[Android.log](#)" usando el trabajo MapReduce definido en "["LogDriver"](#)" y guardar el resultado en "["ejercicio4/logs/output/"](#)".



```
hadoop@nodo1:~/ejercicio4$ hdfs dfs -ls /ejercicio4/logs_output
Found 2 items
-rw-r--r-- 1 hadoop supergroup 0 2025-03-15 23:24 /ejercicio4/logs_output/_SUCCESS
-rw-r--r-- 1 hadoop supergroup 54351 2025-03-15 23:24 /ejercicio4/logs_output/part-r-00000
hadoop@nodo1:~/ejercicio4$
```

Actividad evaluable 2.1.- Hadoop: MapReduce y HDFS

En esta pantalla, una vez finalizada la ejecución del trabajo MapReduce, se utiliza el comando "`hadoop fs -ls /ejercicio4/logs/output`" para verificar el contenido del directorio de salida y comprobar la presencia de los archivos generados, como `part-r-00000`, que indican el éxito del proceso.

```

hadoop@nodo1:~/ejercicio4$ hadoop fs -ls /ejercicio4/logs_output/
part-r-00000
part-r-00001
part-r-00002
part-r-00003
part-r-00004

```

En esta pantalla, se utiliza el comando "`hadoop fs -cat /ejercicio4/logs/output/part-r-00000`" para obtener una vista previa del contenido del archivo de resultados generado, mostrando las direcciones IP y los recursos solicitados junto con sus respectivos conteos.

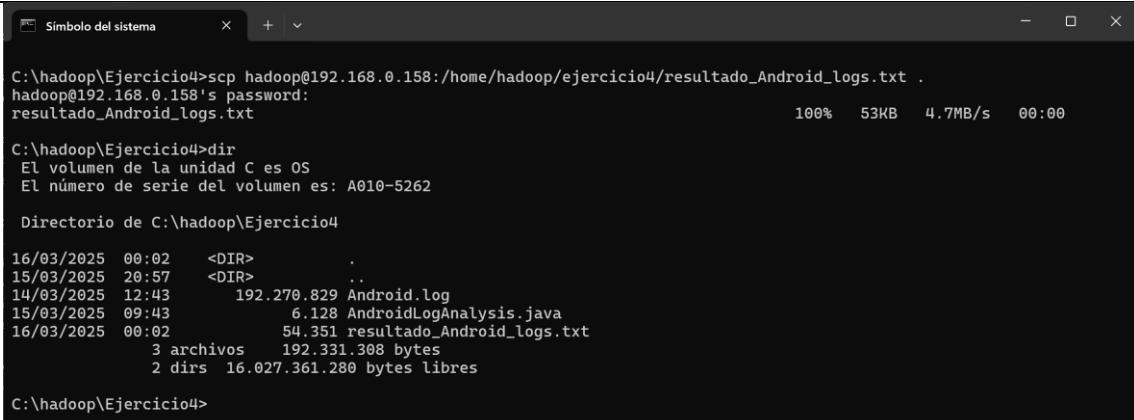
7.5.- Copiar el resultado a la máquina anfitriona

```

hadoop@nodo1:~/ejercicio4$ hadoop fs -get /ejercicio4/logs_output/part-r-00000 /home/hadoop/ejercicio4/result
ado_Android_logs.txt
hadoop@nodo1:~/ejercicio4$ ls -l
total 187848
-rw-r--r-- 1 hadoop hadoop 192270829 mar 15 21:20 Android.log
-rw-r--r-- 1 hadoop hadoop      2641 mar 15 23:03 'AndroidLogAnalysis$LogMapper.class'
-rw-r--r-- 1 hadoop hadoop     1760 mar 15 23:03 'AndroidLogAnalysis$LogReducer.class'
-rw-r--r-- 1 hadoop hadoop     1519 mar 15 23:03 AndroidLogAnalysis.class
-rw-r--r-- 1 hadoop hadoop     3604 mar 15 23:04 AndroidLogAnalysis.jar
-rw-r--r-- 1 hadoop hadoop     6128 mar 15 23:03 AndroidLogAnalysis.java
-rw-r--r-- 1 hadoop hadoop   54351 mar 16 00:00 resultado_Android_logs.txt
hadoop@nodo1:~/ejercicio4$ 

```

En esta pantalla, se procede a copiar los resultados desde HDFS a la máquina virtual, se ejecuta el comando "`hdfs dfs -get /ejercicio4/logs_output/part-r-00000 /home/hadoop/ejercicio4/resultado_Android_logs.txt`" para transferir el archivo `part-r-00000` a un archivo local llamado `resultado_Android_logs.txt`, se verifica que el archivo `resultado_Android_logs.txt` se ha copiado correctamente a la máquina virtual utilizando el comando "`dir`".



```
C:\hadoop\Ejercicio4>scp hadoop@192.168.0.158:/home/hadoop/ejercicio4/resultado_Android_logs.txt .
hadoop@192.168.0.158's password:
resultado_Android_logs.txt                                         100%   53KB   4.7MB/s  00:00

C:\hadoop\Ejercicio4>dir
El volumen de la unidad C es OS
El número de serie del volumen es: A010-5262

Directorio de C:\hadoop\Ejercicio4

16/03/2025  00:02    <DIR>          .
15/03/2025  20:57    <DIR>          ..
14/03/2025  12:43      192.270.829 Android.log
15/03/2025  09:43          6.128 AndroidLogAnalysis.java
16/03/2025  00:02          54.351 resultado_Android_logs.txt
                           3 archivos     192.331.308 bytes
                           2 dirs    16.027.361.280 bytes libres

C:\hadoop\Ejercicio4>
```

En esta pantalla, se utiliza el comando "`scp hadoop@192.168.0.158:/home/hadoop/ejercicio4/resultado_Android_logs.txt .`" para copiar el archivo `resultado_Android_logs.txt` desde la máquina virtual a la máquina anfitriona, y luego se ejecuta el comando "`dir`" para verificar que la copia se ha realizado correctamente.

7.5.- Conclusión

Este programa en Java usa **MapReduce en Hadoop** para procesar archivos de logs de Android en un entorno distribuido. Se ha implementado:

- Extracción de niveles de log y su significado.
- Extracción de etiquetas de log y su conteo.
- Ejecutado en un clúster de Hadoop con entrada y salida en HDFS.

8.- Mapa Mental del Trabajo



El mapa mental resume el proceso completo de aprendizaje y aplicación de Hadoop, desde la configuración del entorno y la comunicación entre máquinas hasta la ejecución de ejercicios prácticos de MapReduce con HDFS para analizar datos textuales y logs, culminando en la extracción de conclusiones y la entrega de resultados.

9.- Conclusiones

9.1.- Análisis de los resultados obtenidos

El análisis de los archivos de logs de **Android** mediante **Hadoop** y el paradigma **MapReduce** permitió extraer información relevante sobre los distintos niveles de logs registrados en el sistema. Se logró:

- Identificar la cantidad de ocurrencias de cada nivel de log, como "Verbose", "Debug", "Info", "Warn", "Error" y "Fatal".
- Analizar la distribución de etiquetas de **log**, proporcionando una visión detallada de los eventos más frecuentes en el sistema.
- Procesar eficientemente grandes volúmenes de datos en un entorno distribuido, aprovechando la escalabilidad de Hadoop.

El resultado final, almacenado en HDFS y posteriormente transferido a la máquina anfitriona, permite un análisis más detallado y facilita su uso en auditorías de rendimiento o depuración de aplicaciones **Android**.

9.2.- Desafíos encontrados y soluciones aplicadas

Durante el desarrollo del ejercicio, se presentaron varios desafíos técnicos que fueron abordados con soluciones específicas:

- **Transferencia de archivos entre la máquina anfitriona y la máquina virtual:** Se utilizaron los comandos "**scp**" y "**hadoop fs -put**" para garantizar la correcta copia de los archivos tanto en el sistema local como en **HDFS**.
- **Compilación y ejecución del programa Java en Hadoop:** Se presentaron errores de compilación relacionados con dependencias de Hadoop, solucionados al incluir correctamente las librerías en el **classpath**.
- **Verificación de la ejecución del trabajo MapReduce:** Se validó la salida de los resultados utilizando "**hadoop fs -ls**" y

"`hadoop fs -cat`" para confirmar la correcta generación del archivo `part-r-00000`.

- **Optimización del análisis de logs:** Se diseñó un `Mapper` eficiente para extraer los niveles y etiquetas de los logs sin afectar el rendimiento del procesamiento distribuido.

9.3.- Reflexión sobre el uso de Hadoop en problemas reales

La aplicación de `Hadoop` en el análisis de archivos de `log` demuestra su capacidad para manejar grandes volúmenes de datos de manera eficiente. Este ejercicio refleja escenarios del mundo real, donde las empresas y desarrolladores necesitan analizar eventos de sistemas operativos o aplicaciones para detectar patrones, errores o fallas críticas.

Entre los beneficios más destacados de `Hadoop` en este tipo de problemas se encuentran:

- **Escalabilidad:** Puede procesar enormes cantidades de datos de forma distribuida, adaptándose a diferentes cargas de trabajo.
- **Tolerancia a fallos:** Su capacidad para replicar datos en múltiples nodos asegura la integridad de la información.
- **Procesamiento eficiente de datos no estructurados:** Permite extraer información valiosa de archivos de texto, como logs de servidores o aplicaciones.

Sin embargo, su implementación requiere conocimientos técnicos específicos y una correcta configuración del entorno, lo que puede representar un desafío inicial para nuevos usuarios. A pesar de esto, `Hadoop` sigue siendo una herramienta fundamental en el análisis de datos masivos y su integración en sistemas empresariales sigue creciendo.

10.- Anexos : Ficheros que se Entrega

A continuación, se presenta el listado de los archivos entregados como parte del trabajo de análisis de logs con Hadoop. Estos archivos están organizados en distintos directorios según su función dentro del trabajo.

10.1.- Directorio principal: C:\hadoop

10.2.- Subdirectorio: Documentación

- **Actividad evaluable 2.1. Hadoop MapReduce y HDFS.docx** : Documento que describe la actividad evaluable relacionada con el uso de Hadoop **MapReduce** y **HDFS**.
- **Mapa Mental Análisis de Big Data con Hadoop HDFS y MapReduce.pdf** : Documento PDF que presenta un mapa mental que resume los conceptos clave y la estructura del análisis de Big Data utilizando las tecnologías Hadoop, **HDFS** y **MapReduce**. Este mapa mental organiza visualmente los componentes fundamentales del ecosistema **Hadoop** y su aplicación en el procesamiento de grandes volúmenes de datos, facilitando la comprensión de los procesos y herramientas implicadas.

10.3.- Subdirectorio: Ejercicio1

- **Quijote.txt** : Archivo de texto con el contenido del "Quijote", utilizado como entrada para ejercicios de procesamiento con **MapReduce**.
- **Resultado_wordcount.txt** : Resultado del procesamiento MapReduce sobre el archivo **quijsote.txt**, con el conteo de palabras en el texto.

10.4.- Subdirectorio: Ejercicio2

- **Sudoku_extremo.txt** : Archivo con una versión de un rompecabezas de **Sudoku** de nivel **extremo**, utilizado en uno de los ejercicios.
- **Sudoku_facil.txt** : Archivo con una versión de un rompecabezas de **Sudoku** de nivel **fácil**, también utilizado en los ejercicios.

10.5.- Subdirectorio: Ejercicio3

- **Apache_logs.txt** : Archivo de logs de **Apache**, utilizado para el análisis de **logs** con **MapReduce**.
- **LogAnalysis.jar** : Archivo **JAR** que contiene el programa **Java** compilado para el análisis de logs de Apache utilizando **MapReduce**.
- **LogAnalysis.java** : Código fuente **Java** para el análisis de **logs** de **Apache** con **Hadoop MapReduce**.
- **Resultado_apache_logs.txt** : Resultado del procesamiento **MapReduce** sobre **apache_logs.txt**, con los resultados del análisis.

10.6.- Subdirectorio: Ejercicio4

- **Android.log** : Archivo de logs de **Android**, utilizado para el ejercicio de análisis de **logs** de **Android** con **Hadoop MapReduce**.
- **AndroidLogAnalysis.jar** : Archivo **JAR** que contiene el programa **Java** compilado para el análisis de **logs** de **Android** con **Hadoop MapReduce**.
- **AndroidLogAnalysis.java** : Código fuente **Java** para el análisis de logs de **Android** con **Hadoop MapReduce**.
- **Resultado_Android_logs.txt** : Resultado del procesamiento **MapReduce** sobre **Android.log**, con el análisis de los **logs** de **Android**.

10.7.- Subdirectorio: Enunciado

- **Actividad Evaluable 2.1.pdf** : Documento PDF con el enunciado completo de la actividad evaluable 2.1, que detalla los ejercicios y requerimientos.

Índice Alfabético

A

Acceso.....	8, 9, 13
actividad.....	5, 8, 49, 51
Actividad	49, 51
activos	11
Además.....	5
algorítmicos	6
algoritmos	27
almacenamiento	6, 7, 8, 11, 23
Android	6, 36, 37, 39, 40, 41, 43, 44, 45, 47, 50
AndroidLogAnalysis	37, 39, 40, 41, 42, 43, 50
anfitrío	9
archivos. 5, 6, 9, 11, 12, 13, 14, 15, 16, 17, 20, 21, 25, 26, 29, 30, 31, 32, 34, 36, 37, 38, 41, 42, 44, 45, 47, 48, 49	
áreas.....	30
argumentos	41
arranque.....	10
auditorías	47

B

beneficios	48
Big	5, 6, 7, 8, 12, 49
Bloc	23
bloques	6

C

cadenas.....	38
cálculos	8
cantidad	5, 47
cantidades	6, 8, 48
capacidad	6, 7, 48
características	29
cargas	48
casos	28, 30
clases.....	37, 38, 41
clásico	5
clientes.....	6
clúster	10, 11
Código.....	50
comando	11
complejidad	27, 30
complejos	5, 6, 8, 30
componentes.....	5, 49
comportamiento	7
comunicación	13, 46
conexión.....	9, 13, 14, 37, 41
configuración	11
configurado	11
conjunto	5, 7
conocimientos	48
consola.....	9
Conteo	5, 18
conteos	34, 40, 44
continuación.....	8, 13, 49
copias	6
correctamente	11
correspondientes	7
CPU	28, 29
creación.....	25, 30
credenciales	9
críticas	48

D

Data.....	5, 6, 7, 8, 12, 49
DataNode	11

Debian.....	8, 9, 13
Debug	39, 47
definición	38
demonios	11
dependencias	47
depuración	47
desafíos	8, 47
desarrolladores	48
desarrollo.....	5, 47
Después.....	7, 9
diferencias	27
dirección	9, 12, 14, 35, 36, 41
direcciones	6, 34, 44
directorio 16, 17, 19, 20, 21, 25, 30, 32, 33, 34, 36, 37, 41, 42, 43, 44	
directorios	9, 30, 42, 49
distintos	30, 47, 49
distribución	29, 47
distribuido	11
Distributed	5, 6, 9
Documentación	49
Documento	49, 51

E

eficientes	5
ejecución	6, 9, 21, 26, 27, 28, 29, 32, 34, 44, 46, 47
ejecuta	11
ejercicio	5, 6, 47, 48, 50
ejercicios	5, 6, 46, 49, 50, 51
elección	8
elementos	36
empaquetarlo	36
empresariales	48
empresas	8, 48
encargados	7
enormes	48
enseña	6
Enter	16
entorno	6, 8, 12, 13, 45, 46, 47, 48
errores	47, 48
escalabilidad	47
escenarios	30, 48
esenciales	8
espacio	17, 18, 28
especificación	38
específicos	15, 48
estructuras	37
estudiante	5
etiquetas	6, 45, 47, 48
Evaluable	51
eventos	47, 48
examples	21, 26, 27
expresiones	6
extensión	9
Extracción	45

F

falla	6
fallos	6, 48
fases	7
ficheros	5, 6, 15, 29, 30
formas	13
formato	18
frecuencia	5, 21
frecuentes	47
función	49
funcionando	11
fundamentales	5, 49

Actividad evaluable 2.1.- Hadoop: MapReduce y HDFS

G

generación.....	8, 48
gestión	9, 28, 29
gestionar	11
gestor	15
Gutenberg	18, 23

H

hadoop	11
Hadoop....	5, 6, 8, 9, 11, 12, 13, 15, 16, 21, 23, 26, 27, 29, 32, 34, 36, 37, 38, 39, 40, 41, 45, 46, 47, 48, 49, 50
HashMap	38
HDFS ..	5, 6, 7, 8, 9, 11, 12, 13, 15, 16, 17, 18, 20, 21, 22, 23, 32, 33, 34, 35, 36, 37, 38, 42, 43, 44, 45, 46, 47, 49
herramienta	48

I

Implementación.....	5, 6
Info.....	39, 47
iniciar.....	11
inicio	28
instalación	6
instancias	7
integración.....	8, 48
integridad	48
interacción	12
interfaz.....	9, 12
Intro 10, 14, 15, 16, 17, 18, 19, 20, 21, 25, 26, 29, 30, 31, 32, 33, 36, 37, 42, 43	
IOException	37

J

JAR	32, 34, 36, 42, 50
-----------	--------------------

L

Lectura.....	6, 30, 36
librerías	47
líneas	22, 39
Linux.....	19, 20, 30
lista	16
LogAnalysis.....	30, 31, 32, 33, 34, 50
LogDriver.....	43
lógica	6
LogReducer	40, 42

M

manipulación.....	5, 6
Map	7, 38
Mapa.....	7, 46, 49
Mapper.....	32, 36, 38, 39, 40, 48
MapReduce	5, 6, 7, 8, 18, 21, 23, 34, 36, 37, 38, 39, 40, 42, 43, 44, 45, 46, 47, 49, 50
máquina	8, 9, 10, 12, 13, 14, 15, 16, 19, 20, 22, 23, 24, 25, 29, 30, 31, 35, 36, 37, 41, 44, 45, 47
masivos.....	6, 8, 48
modelo	5, 7
Moodle	8
mostrará.....	11
móviles	5
múltiples	6, 8, 48

N

namenode.....	11
NameNode.....	11
NAT	13

necesarias.....	38
necesarios.....	9, 11
niveles.....	6, 28, 30, 45, 47, 48
nodos	6, 7, 8, 48
Notas	23

O

ocurrencias	36, 39, 40, 47
opción	9
operación	37
operaciones	9, 13, 28, 29
operativos	48
organizaciones	8

P

página	18, 24, 25
palabras.....	5, 18, 21, 23, 49
pantalla .	10, 12, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 29, 30, 31, 32, 33, 34, 35, 36, 37, 41, 42, 43, 44, 45
paralelo	7, 8
pares	7
pasos	12
patrones	7, 48
Pattern	35
PDF.....	49, 51
Plain.....	18
porción	7
posibles	30
práticos	5, 46
presentes	6
principales.....	13
problemas	5, 6, 30, 48
procedimientos	6
procesos	11, 15, 49
programación	5, 7, 8
Project	18, 23
pruebas	30
puedes	13
Puertos	9

Q

Quijote	18, 49
---------------	--------

R

reales	48
Recomendación.....	30
recurso	36
red 6, 13	
redes	7, 13
reducción	7
Reducer	32, 36, 38, 40, 41
registros	6
relevante	6, 47
remotos	13
rendimiento	47, 48
requerimientos	51
requisitos	9
resolución	5, 26, 27
respectivos	34, 44
rutas	41

S

SCP	14
SecondaryNameNode	11
sencillo	13
servicios	11, 13
servidores	48
siguientes	5

Actividad evaluable 2.1.- Hadoop: MapReduce y HDFS

sistema	11	Utilización	6
sistemas.....	48		
sociales	7		
SSH	9, 13, 14		
StringTokenizer	38		
Sudoku.....	5, 6, 24, 26, 27, 28, 29, 30, 50		
System.....	5, 6, 9		
<hr/>			
T			
tareas	5, 7, 8		
tecla. 14, 15, 16, 17, 18, 19, 20, 21, 25, 26, 29, 31, 32, 33, 36, 37, 42, 43			
técnicos.....	47, 48		
tecnologías	5, 49		
tendencia.....	30		
términos.....	28		
Text	18		
típicos	5		
tipos	6		
tolerancia	6		
trabajos.....	7		
transferencia.....	13, 14		
<hr/>			
U			
usuario	9, 28, 41		
UTF.....	18		

v			
valores	7		
variación.....	29		
Verbose	39, 47		
Verificación	47		
verificar	11		
versión	50		
VirtualBox	8, 9, 13		
virtuales	13		
visión	47		
vista.....	34, 44		
volúmenes	5, 6, 7, 8, 15, 47, 48, 49		

w			
Warn	47		
WordCount.....	5		

Y			
YARN	15		