GRADO

INTRODUCCIÓN AL MANEJO DE DATOS MASIVOS CON HADOOP Y HERRAMIENTAS INSPIRADAS EN SQL

PRÁCTICA DE LA ASIGNATURA SISTEMAS DE BASES DE DATOS

2021-2022

Versión 1.0

Dr. Agustín C. Caminero Herráez — Dr. Luis Grau Fernández

GRADO EN INGENIERÍA INFORMÁTICA

CONTENIDO

1 ¿Qué son los datos masivos o *Big Data*?

Los datos masivos, también conocidos como *Big Data*, son datos que cumplen entre otras las siguientes condiciones (conocidas como las 3 Vs):

- Tienen gran **volumen**. Estamos hablando de terabytes o petabytes de información, al menos.
- Tienen gran **velocidad**. Son datos que varían muy rápidamente, lo que hace que su tiempo de vida sea muy limitado. Esto impone severas restricciones temporales a su almacenamiento y procesamiento, ya que si no se utilizan las técnicas apropiadas, estos datos no se podrán aprovechar convenientemente.
- Tienen una gran **variedad**. No están limitados a texto, sino que incluyen cualquier tipo de dato, como por ejemplo vídeo, audio, imágenes.

Estas condiciones hacen que los sistemas de almacenamiento y procesamiento de datos tradicionales (como por ejemplo las bases de datos relacionales tradicionales) no sean los más indicados para trabajar con Big Data – esto se verá en mayor detalle más adelante en este documento. Por eso, se han desarrollado tecnologías que permiten el trabajo con datos de estas características, una de las más ampliamente utilizadas es Hadoop.

El Big Data se puede aprovechar en una gran cantidad de campos. Por ejemplo, en temas médicos (expedientes médicos, resultados de pruebas, ...), negocios (compras, ventas, transacciones entre empresas, movimientos de la Bolsa, *Business Intelligence*...), redes sociales (mensajes de Twitter, Facebook, ...), o servidores de Internet (logs que recogen los accesos que reciben los servidores, ...).

En esta práctica vamos a introducir la herramienta de Big Data conocida como Hadoop. Profundizaremos en su estructura y en sus componentes principales (el sistema de archivos HDFS y el modelo de programación MapReduce), presentaremos algunos ejemplos de funcionamiento, antes de proponer una serie de ejercicios de evaluación.

2 Introducción a Hadoop

2.1 ¿Qué es Hadoop?

Hadoop es un entorno software para el almacenamiento, procesamiento y análisis de datos masivos, también conocidos como *Big Data*. Entre sus características más importantes se encuentran las siguientes:

- Hadoop es **distribuido**:
    - Se ejecuta en un *cluster* de ordenadores, un conjunto de ordenadores conectados entre ellos mediante una red de interconexión que funcionan de forma coordinada (ver Figura 1).
    - El cluster permite que el usuario del sistema no tenga que preocuparse de realizar tareas tales como decidir en qué ordenador se ejecutan los trabajos, o de iniciar sesión en la máquina adecuada.
    - El cluster ofrece una imagen única de sistema (*Single System Image*, SSI) a sus usuarios, de forma que estos no tienen que ser conscientes de las infraestructuras subyacentes.

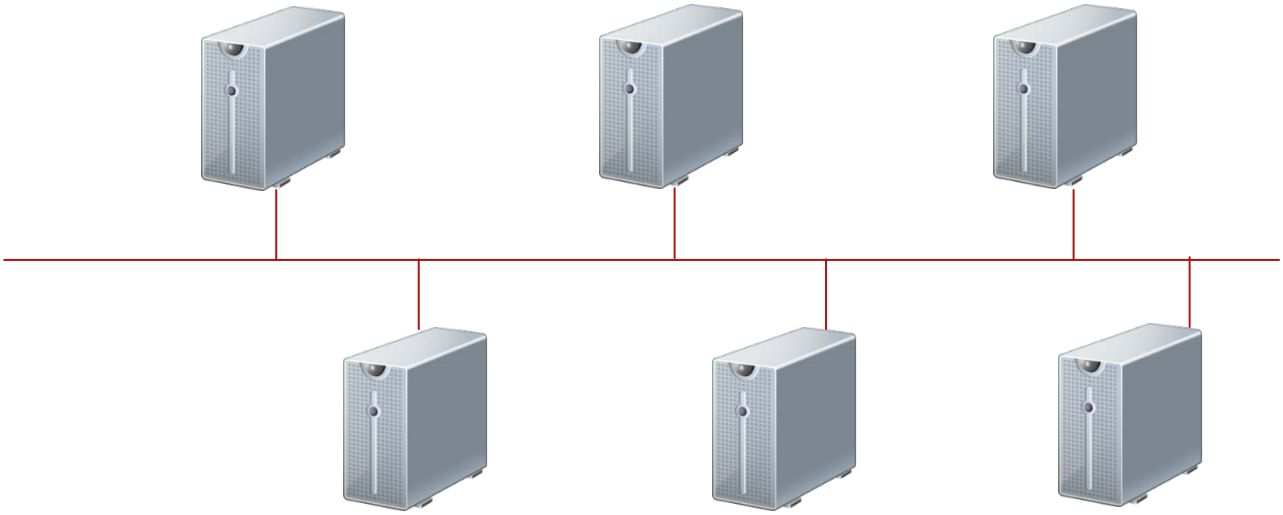

Figura 1. Cluster de ordenadores

- Hadoop es **escalable**:
    - Añadir nodos incrementa la capacidad de forma proporcional (ver Figura 2). Al contrario que otras tecnologías, que al incrementar el número de nodo implica un sobrecoste, en Hadoop el incremento de los nodos del cluster incrementa directamente la capacidad de procesamiento.

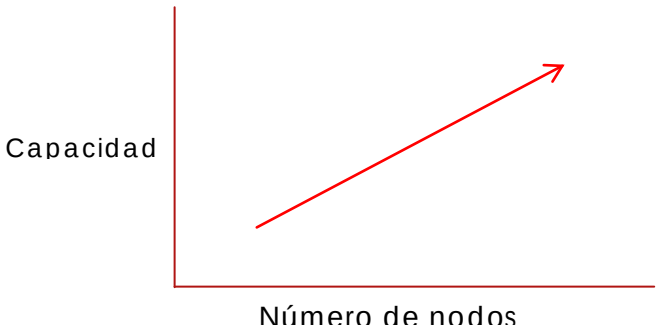

Figura 2. Escalabilidad de un sistema Hadoop

- Hadoop es **tolerante a fallos**:
    - Los fallos son normales en sistemas distribuidos. En grandes sistemas informáticos, como por ejemplo, Google, que están formados por cientos de miles de ordenadores, suceden fallos a diario.
    - El fallo de un nodo no afecta al sistema, que continúa funcionando. El sistema debe ser capaz de detectar el fallo y adaptarse a él con el fin de seguir proporcionando el servicio mientras el fallo se soluciona.
    - El maestro reasigna las tareas a otro nodo. De esta forma, el servicio sigue siendo proporcionado por las máquinas que siguen en funcionamiento.
    - No hay pérdida de datos gracias a la replicación. No solamente el servicio se debe seguir proporcionando, sino que la información almacenada debe seguir estando disponible sin pérdidas.
    - Cuando un nodo se recupera, vuelve al sistema automáticamente.

- Hadoop es **open-source**:
    - Su código fuente está disponible de forma abierta para que quien lo desee lo descargue, modifique, ...
    - Una gran cantidad de desarrollos software se publican de forma open-source (ver Figura 3), entre los más conocidos se encuentran el sistema operativo para smartphones Android, el navegador de Internet Mozilla Firefox, o el cliente de correo electrónico Mozilla Thunderbird.

Figura 3. Proyectos open-source

2.2 Necesidad de Hadoop

La creciente necesidad de datos hace que los sistemas distribuidos tradicionales no sean eficientes. El principal problema que presentan es el almacenamiento de los datos, ya que se suelen almacenar en una base de datos externa a los nodos de cómputo, tal como muestra la Figura 4.

De esta forma, cada vez que se inicia un procesamiento sobre los datos, los nodos de trabajo deben recuperar la información de la base de datos, lo cual supone un cuello de botella ya que todos los nodos implicados en esos cálculos deben recuperar los datos prácticamente al mismo tiempo y algunos nodos deberán esperar, lo cual retrasa el comienzo de los cálculos y afecta negativamente a la productividad del sistema informático.

En cambio, Hadoop proporciona una nueva gestión de los datos para eliminar ese cuello de botella, como se muestra en la Figura 5. En Hadoop, los datos se almacenan en los mismos ordenadores donde se realizarán los cálculos, de forma que se distribuyen entre ellos en el momento en que se almacenan en el sistema. Por tanto, cuando se inicia un procesamiento, los datos ya se encuentran en los nodos de trabajo, por lo que pueden empezar a trabajar sin demora. Debido a esto, la productividad del sistema informático mejora considerablemente.

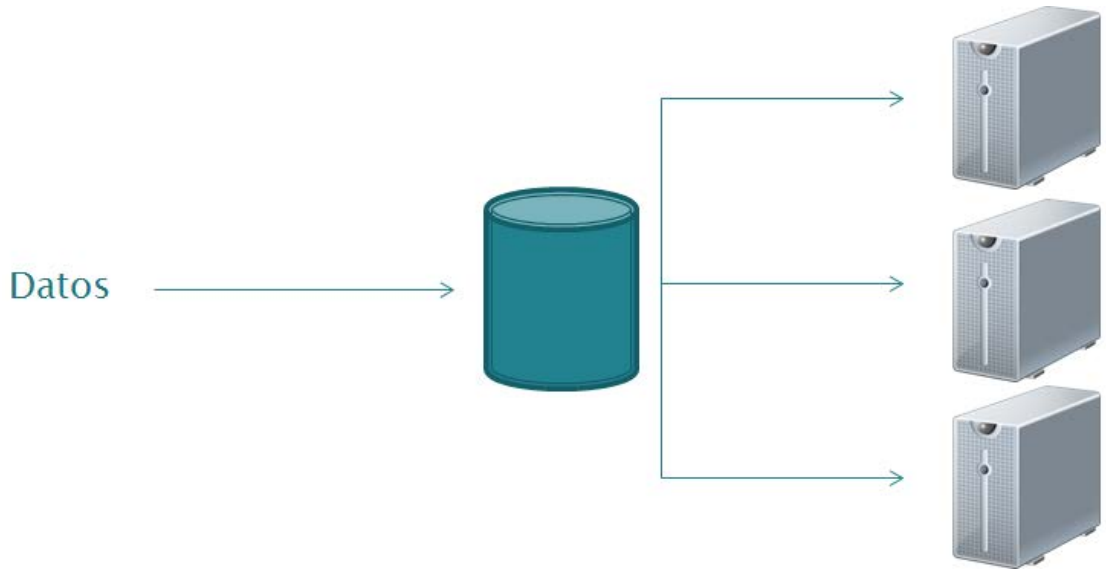

Figura 4. Arquitectura tradicional de un sistema distribuido

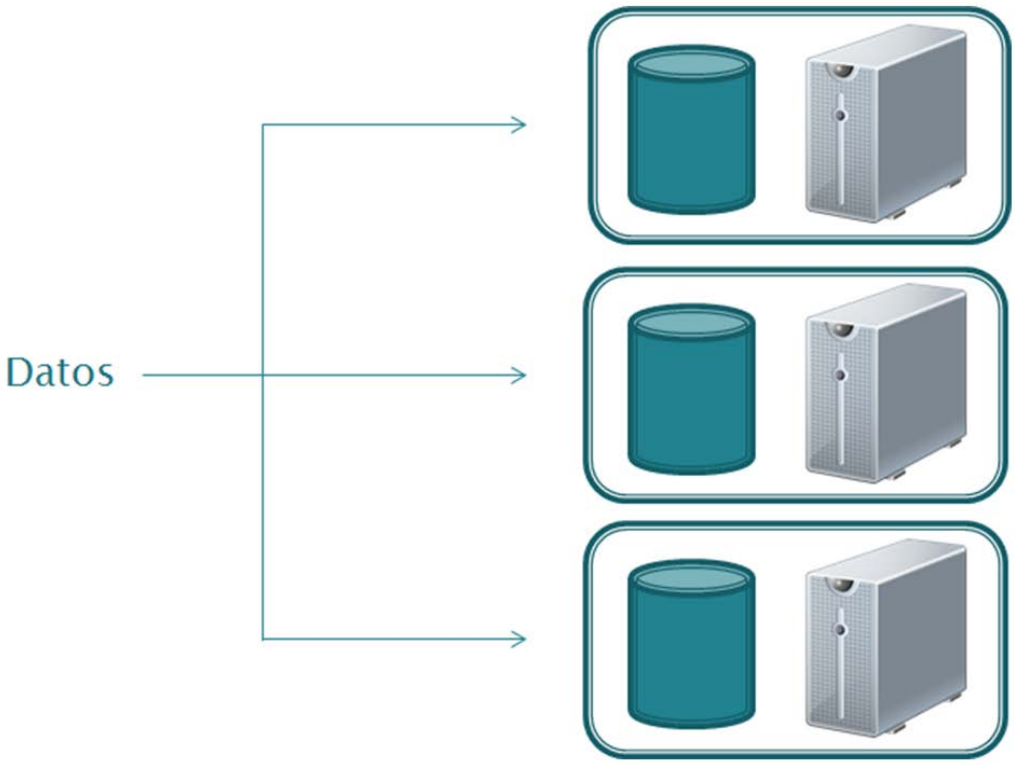

Figura 5. Arquitectura de Hadoop

Cuando los datos se almacenan en Hadoop, estos se dividen en bloques, que es la unidad en la cual se realizan los cálculos (procesos *map*). Un nodo maestro gestiona los cálculos para asegurar que se realizan correctamente. Este proceso se muestra gráficamente en la Figura 6.

De esta forma, se persigue que los nodos se comuniquen entre ellos lo menos posible. Como los datos se distribuyen cuando se almacenan, evitamos el cuello de botella que supone leer los datos de la base de datos. Además, los cálculos se llevan a los datos, no al revés, es decir, que los cálculos se ejecutan en los ordenadores que almacenan los datos sobre los que se deben realizar tales cálculos.

Finalmente, Hadoop replica los bloques en varios nodos por razones de prestaciones y de tolerancia a fallos.

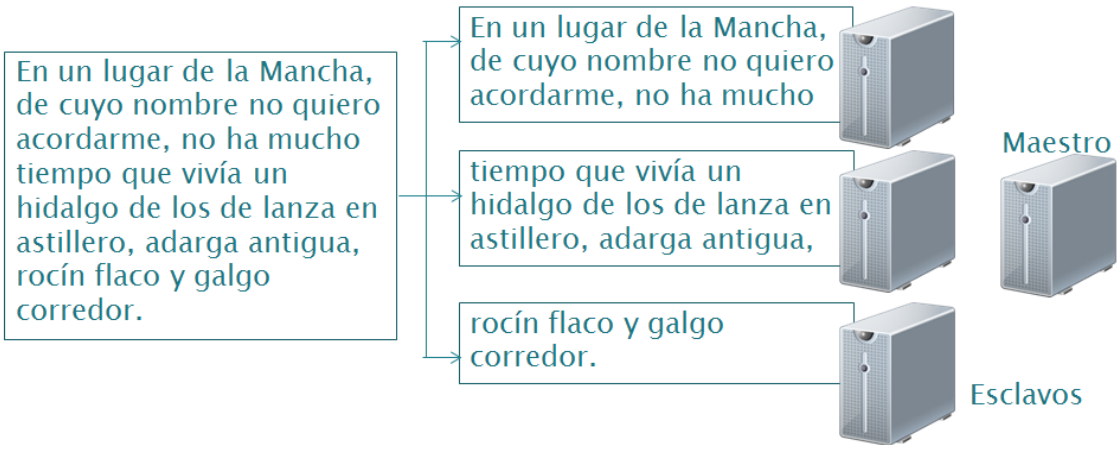

Figura 6. Datos divididos en bloques

3 Hadoop y su estructura

Hadoop forma parte de un ecosistema con múltiples componentes, algunos de los cuales se muestran en la Figura 7. Los componentes principales de Hadoop son el sistema de ficheros distribuido de Hadoop (*Hadoop Distributed File System, HDFS*) y el entorno de programación MapReduce. A continuación veremos brevemente las características principales de cada uno.

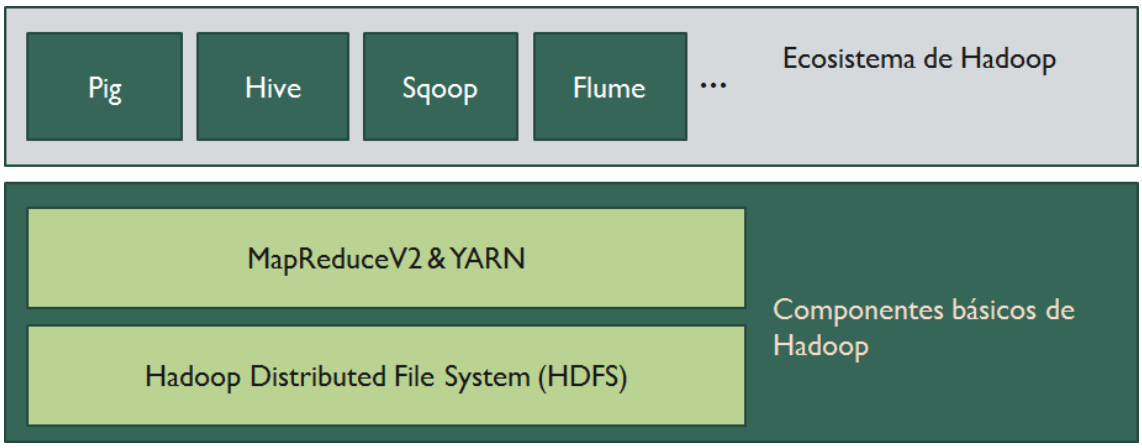

Figura 7. Ecosistema de Hadoop

3.1 Hadoop Distributed File System (HDFS)

HDFS es un sistema de archivos distribuido y tolerante a fallos. Funciona sobre el conjunto de los nodos de un cluster de Hadoop, balanceando la carga de archivos entre las máquinas del cluster, de forma equitativa. Gracias a su naturaleza distribuida, proporciona alta disponibilidad y altas prestaciones que le permiten ser capaz de manejar grandes ficheros.

Para insertar datos en HDFS existen una variedad de formas:

- Copiarlos manualmente utilizando un comando.
- Utilizando la herramienta Flume, que recoge datos de diversas fuentes y los inserta automáticamente.
- Utilizando la herramienta Sqoop, que transfiere datos entre HDFS y bases de datos relacionales.
- …

Ahora vamos a ver con detalle los procesos de escritura y lectura de datos en HDFS.

3.1.1Escrituras en HDFS

La forma en que HDFS gestiona las escrituras de archivos se explica a continuación:

1. Primero, el fichero de datos se divide en bloques de tamaño fijo, normalmente 64 o 128 MB. Esto se muestra en la Figura 8.
2. Tras esto, cada bloque se almacena en varios de los nodos del cluster. Esto se muestra en la Figura 9.

De esta forma, al estar cada bloque de datos replicado en varios nodos del cluster, en caso de fallo de alguno de los nodos, no se pierde información, y el sistema puede seguir funcionando correctamente (exceptuando el decremento de la capacidad del sistema consecuencia del fallo).

Para gestionar HDFS, tenemos un nodo especial en el cluster que se llama *NameNode.* Esta máquina almacena para cada fichero dónde se almacenan los bloques que lo forman. Los nodos donde se almacenan los datos son los *DataNodes*.

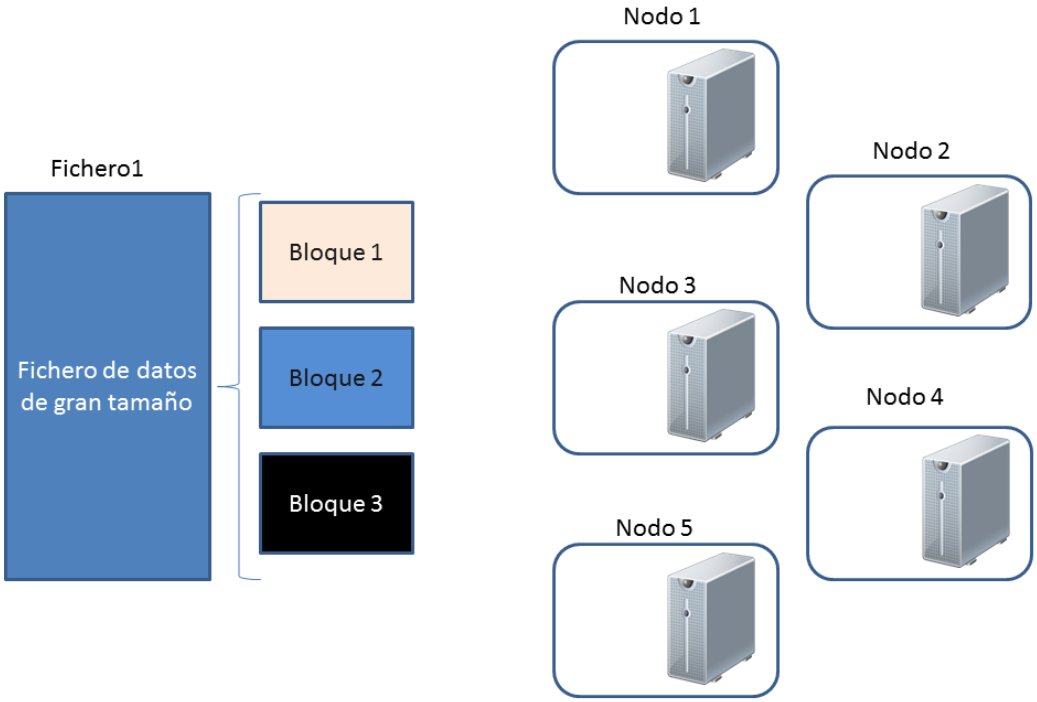

Figura 8. Escrituras en HDFS: Dividir el archivo en bloques

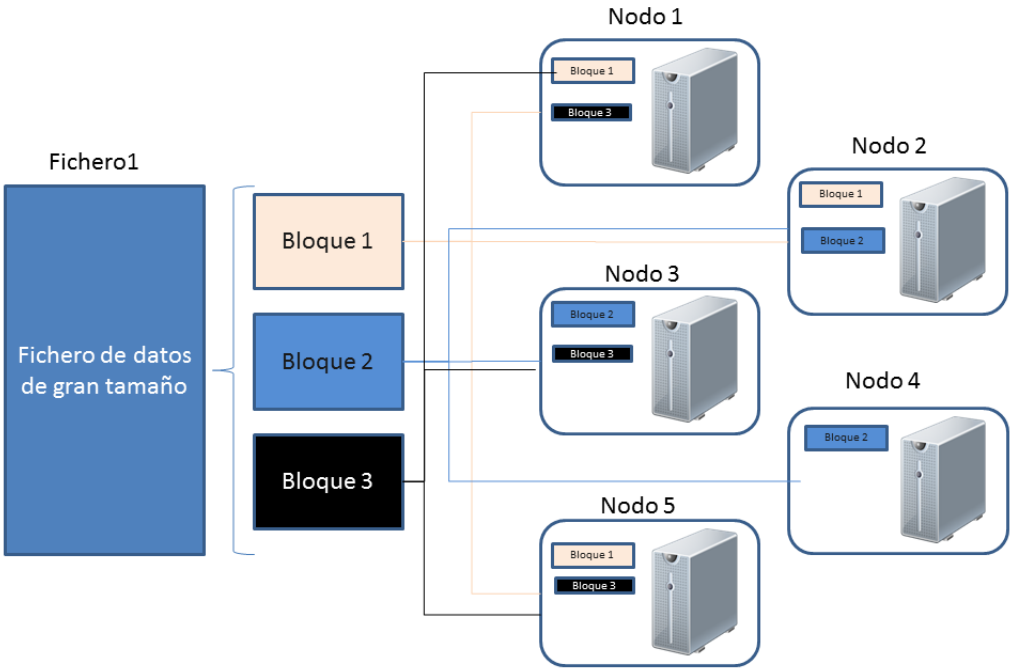

Figura 9. Escritura en HDFS: Almacenar cada bloque en múltiples nodos del cluster

3.1.2 Lecturas en HDFS

El proceso de lecturas de ficheros en HDFS es como sigue:

1. En primer lugar, el ordenador del cliente realiza la petición de lectura de un archivo al NameNode (esto se muestra en la Figura 10).
2. Entonces, el NameNode chequea en sus registros qué bloques pertenecen a dicho archivo así como dónde están almacenados tales bloques, y devuelve esta información al cliente (ver Figura 11)
3. Tras esto, el cliente solicita directamente a los nodos correspondientes que le envíen los bloques de dicho fichero (ver Figura 12). Finalmente, los nodos le envían al cliente los bloques correspondientes directamente, sin pasar por el NameNode.

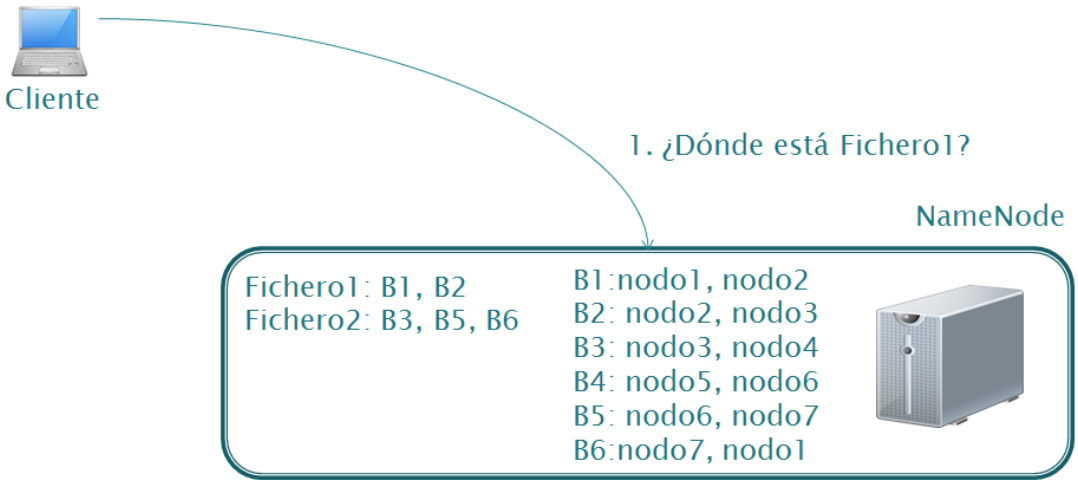

Figura 10. Lectura de HDFS: Petición del cliente

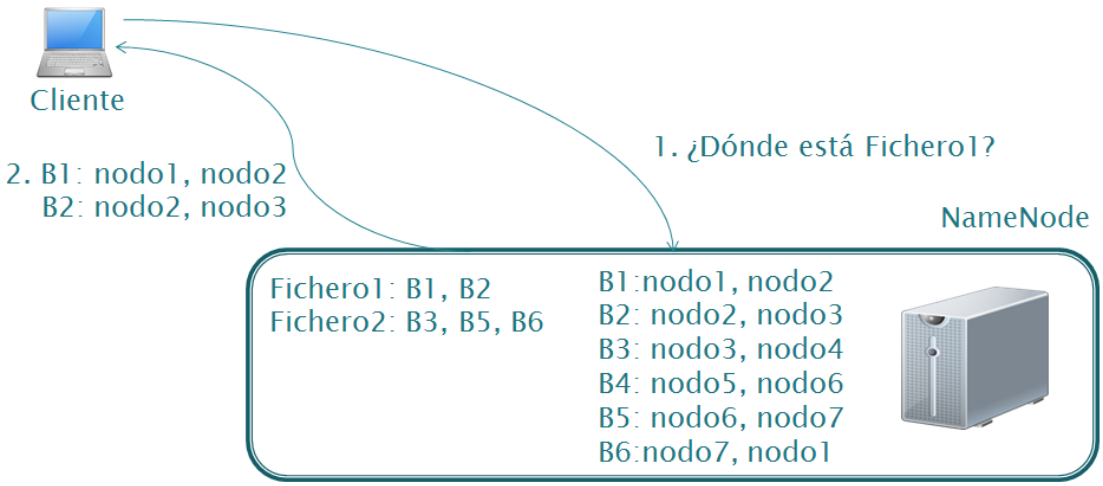

Figura 11. Lectura de HDFS: Respuesta del NameNode

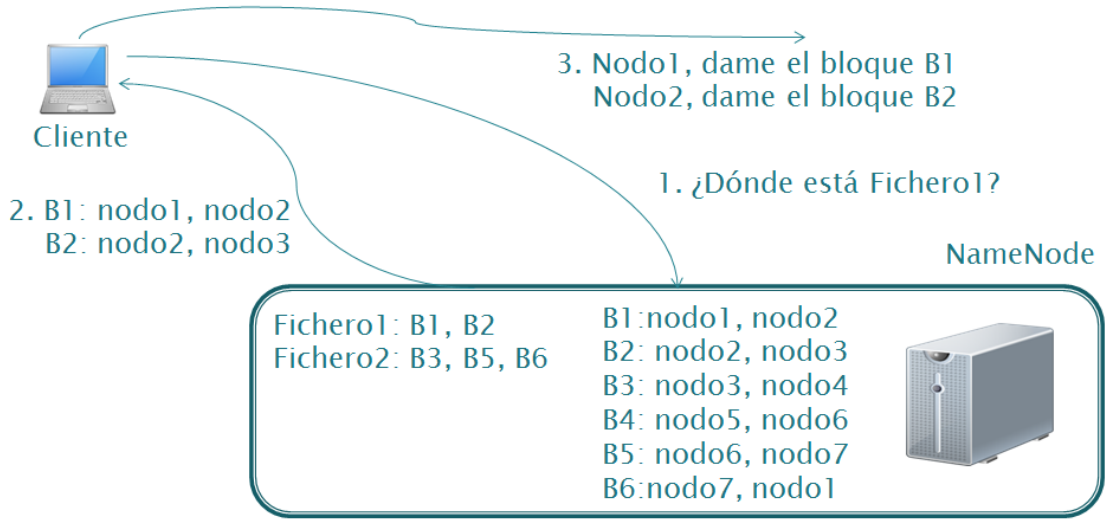

Figura 12. Lectura HDFS: Petición a los nodos

3.2 MapReduce

MapReduce es un modelo de programación paralela distribuida enfocado a grandes conjuntos de datos procesados en un cluster. Automatiza la paralelización de las ejecuciones así como la distribución de las tareas entre los nodos.

MapReduce consta de varias fases:

- *Map*:
  - Opera en un único bloque de un fichero HDFS.
  - Se ejecuta siempre que sea posible en el nodo que almacena dicho bloque, lo que minimiza el tráfico sobre la red.
- *Shuffle & sort* (barajar y ordenar):
  - Ordena y consolida los datos intermedios de todos los maps.
  - Sucede cuando todas las tareas map han acabado y antes de que comiencen las tareas *reduce*.
- *Reduce*:
  - Opera sobre los resultados intermedios ordenados y barajados (la salida de las tareas map).
  - Produce los resultados finales.

Para entender mejor Mapreduce, vamos a ver un ejemplo típico consistente en un contador de palabras. Partimos de un fichero de texto y deseamos obtener un conteo de las palabras que lo forman, y esto se realiza a través de las funciones Map y Reduce, como se muestra en la Figura 13.

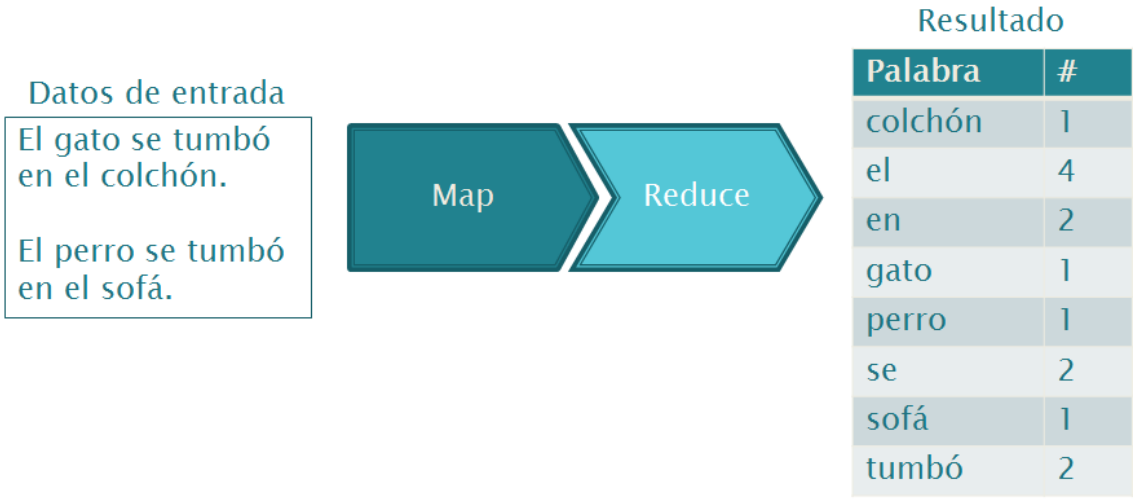

Figura 13. Ejemplo de MapReduce

Los pasos que sigue Hadoop para realizar esta tarea son los que siguen:

1. En primer lugar, se ejecutan procesos *map* sobre cada bloque que forma dicho fichero. Estos procesos se ejecutan en la medida de lo posible en los ordenadores que almacenan dichos bloques de forma que se minimice la información que se transfiere a través de la red de interconexión. Como resultado de esta fase, se obtiene un listado de pares <clave, valor>, que en este caso tiene como clave cada palabra, y como valor 1. Esto se muestra en la Figura 14.

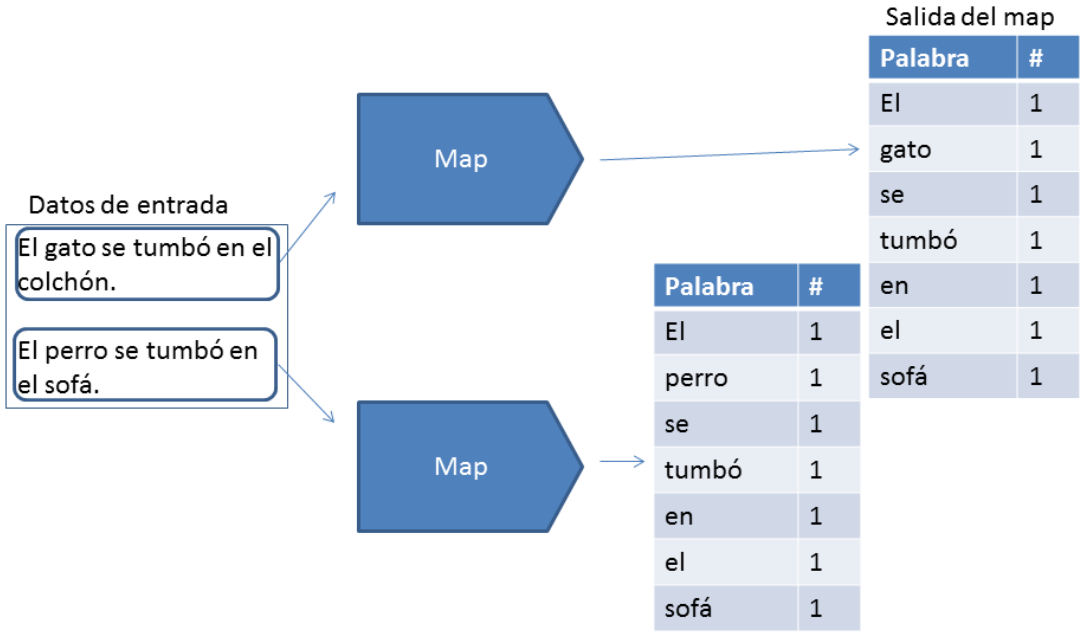

Figura 14. Ejemplo de Mapreduce: Map

2. Tras finalizar los *map*, la salida de todos los *map* se combina dentro de la fase de *shuffle and sort*, de forma que las parejas <clave,valor> que tengan la misma clave se agrupan (es decir, las que tengan la misma palabra), de la forma en que lo muestra la Figura 15.

3. Finalmente, la fase *reduce* toma la salida de la fase anterior y agrega los valores que tengan la misma clave, dando como resultado el sumatorio de ocurrencias de cada palabra (ver Figura 16).

Hadoop se encarga de automatizar la diseminación de las tareas a través de los nodos, la gestión de los fallos, ... de forma que el usuario solamente se tiene que concentrar en implementar las funciones *map* y *reduce*. Además, con el fin de evitar realizar transferencias de información a través de la red, que crearían latencias, Hadoop trata de ejecutar las tareas *map* en los nodos del cluster que almacenan los bloques sobre los que opera dicha función *map*.

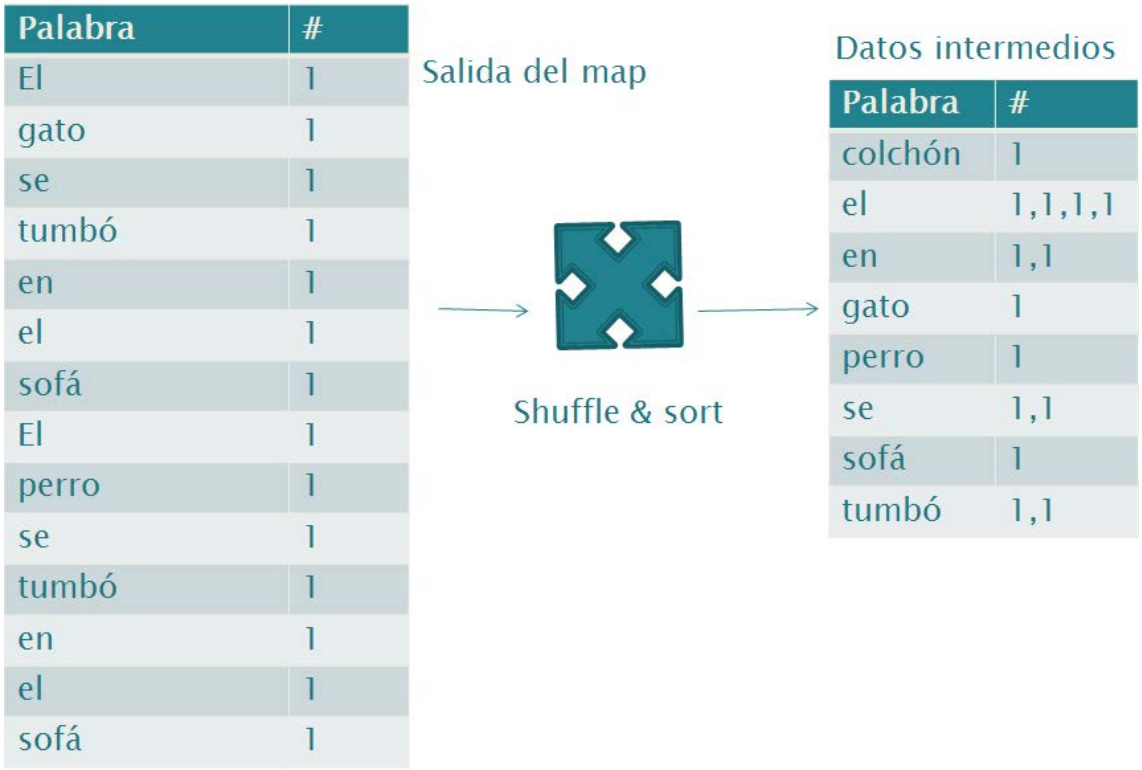

Figura 15. Ejemplo de Mapreduce: Shuffle & sort

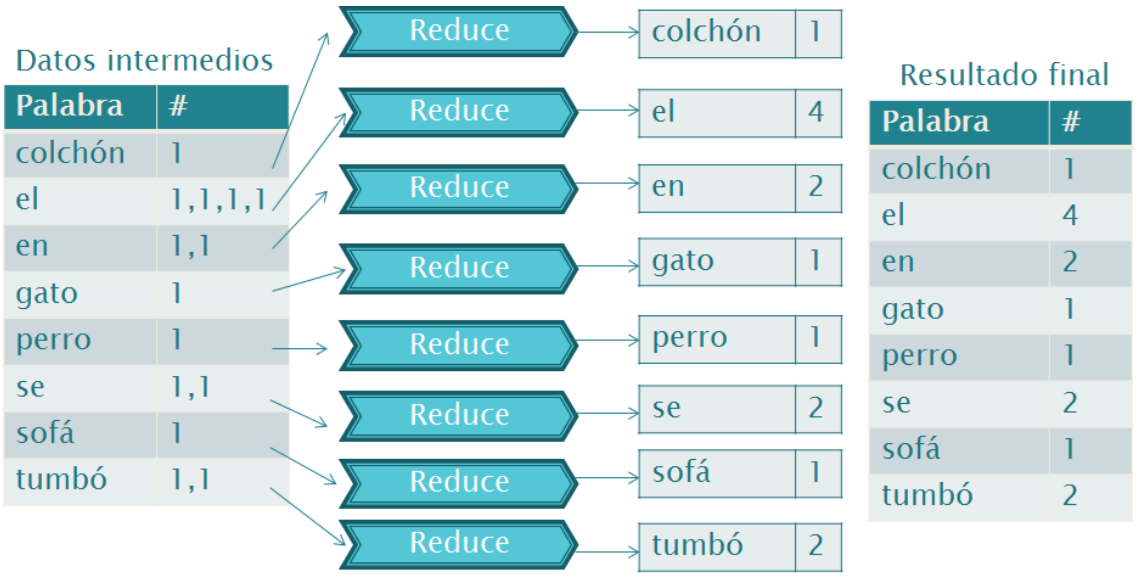

Figura 16. Ejemplo de Mapreduce: Reduce

De forma opcional, también existe la posibilidad de añadir un paso a la salida del map, que es el *combiner*. Este paso se ejecuta a la salida del map en el mismo nodo y suele consistir en el mismo código del reducer. Si tenemos el combiner, los datos que llegan al reducer se encuentran parcialmente reducidos y el coste de transferir por la red y procesar en el reducer esos datos se minimiza.

En este documento se mostrarán más adelante ejemplos de código MapReduce junto con instrucciones para su ejecución.

3.3 Ecosistema de Hadoop

Hadoop es un proyecto vivo con una gran variedad de herramientas que incrementan su funcionalidad y facilidad de uso.

- **Hive**: Interacción con Hadoop utilizando un lenguaje de consultas similar a SQL.
- **Pig**: Interacción con Hadoop utilizando un lenguaje de script.
- **Sqoop**: Comunicación de Hadoop con bases de datos relacionales.
- **Flume**: Herramienta para mover grandes cantidades de datos.
- **HUE**: Interfaz gráfico para Hadoop.
- …

3.4 Distribuciones

Existen una variedad de distribuciones de Hadoop, proporcionadas por diversas empresas, que extienden su funcionalidad y le proporcionan soporte.

- Cloudera, Hortonworks, MapR

Una de las más importantes es **Cloudera**, que será la que utilicemos en este curso.

Entre las ventajas de Cloudera se encuentran …

- Proporciona asistentes que facilitan la instalación.
- Es una de las distribuciones más utilizadas.
- Hay gran cantidad de materiales, tutoriales, libros… que se basan en ella.

4 Herramientas de programación MapReduce inspiradas en SQL

Como hemos visto en el apartado anterior, la programación MapReduce implica el diseño y el desarrollo de, al menos, la función map y la función reduce. Aunque existen patrones de diseño bien conocidos, algunos de los cuales se encuentran en la referencia [10], programar aplicaciones MapReduce es una tarea compleja que requiere una curva de aprendizaje significativa.

Es por esto que se han desarrollado herramientas que facilitan la programación MapReduce, algunas de las cuales proporcionan un lenguaje similar a SQL. De esta forma, redactando consultas en un lenguaje similar a SQL, se genera el código MapReduce.

En concreto, Apache Hive es una de dichas herramientas. Su lenguaje de consultas, HiveQL, ofrece una sintaxis similar a SQL. De esta forma, desarrolladores con experiencia en SQL pueden empezar a programar MapReduce con una curva de aprendizaje reducida. Hay que destacar que lo que hace Hive es tomar como entrada consultas realizadas en HiveQL y generar como salida código MapReduce, como se muestra en la Figura 17. Dicho código MapReduce se ejecuta de la forma descrita anteriormente.

En el próximo apartado veremos ejemplos de trabajo con Hive junto con instrucciones para su ejecución.

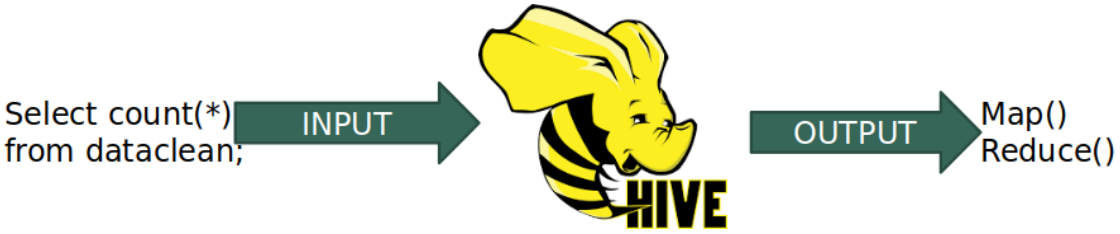

Figura 17. Qué hace Hive.

5 Ejemplos de trabajo con Hadoop

En este apartado vamos a presentar una serie de ejemplos prácticos de trabajo con Hadoop. En ellos trabajaremos con la distribución Cloudera, que ha sido seleccionada por las ventajas arriba mencionadas. Además, se puede descargar de su web una máquina virtual donde podremos practicar y aprender los conceptos de Hadoop.

Presentaremos una serie de ejemplos de programas MapReduce realizados en Python. También veremos cómo trabajar con Hive utilizando tanto el terminal como el interfaz gráfico HUE y jupyter notebooks. Ésta última herramienta, jupyter notebooks, es un entorno de desarrollo basado en web que permite preparar documentos que combinen código ejecutable, imágenes, gráficos, y comentarios, y es ampliamente utilizada en el tratamiento de datos masivos debido a su integración con herramientas de procesamiento paralelo como Hadoop o Spark.

Para la realización de esta práctica, se ha preparado un entorno de trabajo basado en los recursos oficiales de Cloudera, que han sido extendidos con las herramientas necesarias para este ejercicio. La forma de desplegar este entorno de trabajo se detalla en un documento aparte, que se encuentra publicado en el curso virtual.

5.1 Ejemplo inicial: Contador de palabras

En este ejemplo veremos cómo programar un trabajo MapReduce, programando las tareas map y reduce con funciones Python. Veremos la forma de simular la ejecución de un trabajo MapReduce sin necesidad de utilizar el software Hadoop simplemente con órdenes del sistema operativo. El ejemplo que vamos a mostrar es el contador de palabras.

Para empezar veremos la función map, implementada en el fichero *mapper.py*, cuyos contenidos se encuentran comentados:

```
#!/usr/bin/env python
import sys
# entrada de la entrada estandar STDIN
for line in sys.stdin:
    # eliminamos espacios blancos al principio y final
    line = line.strip()
    # dividimos la linea en palabras
    words = line.split()
    # incrementamos los contadores
    for word in words:
        # escribimos los resultados a la salida estandard STDOUT.
        # Esta salida sera la entrada para el reduce, es decir, para reducer01.py
        # delimiado por tab, para cada palabra ponemos 1 ocurrencia
        print '%s\t%s' % (word, 1)
```

A continuación mostramos la función reduce, implementada en el fichero *reducer.py*, igual que antes se encuentra comentado:

```
#!/usr/bin/env python

from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None

# entrada desde STDIN
for line in sys.stdin:
    # eliminamos espacios blancos al principio y final
    line = line.strip()

    # parseamos la entrada que hemos obtenido del mapper01.py
    word, count = line.split('\t', 1)

    # pasamos el contador de string a int
    try:
        count = int(count)
    except ValueError:
        # si el contados no es un numero, descartamos la linea
        continue

    # este if solamente funciona porque Hadoop ordena la salida del map por
la clave (aqui es word) antes de pasarsela al reducer
    if current_word == word:
        current_count += count
    else:
        if current_word:
            # escribir resultado a STDOUT
            print '%s\t%s' % (current_word, current_count)
```

```
        current_count = count
        current_word = word

# escribimos la ultima palabra
if current_word == word:
   print '%s\t%s' % (current_word, current_count)
```

Una vez tenemos estos ficheros preparados, podemos similar la ejecución de MapReduce de la siguiente forma, utilizando órdenes del sistema operativo Linux. Para ello partimos de un fichero de texto llamado *fichero.txt* que se encuentra en la misma carpeta que los ficheros mapper.py y reducer.py. Lo primero debemos abrir un terminal, y desde él, ejecutar lo siguiente:

```
cat fichero.txt | ./mapper.py |sort | ./reducer.py
```

Para ejecutar correctamente esta orden, los ficheros mapper.py y reducer.py deben tener permisos de ejecución.

De esta forma, veremos por pantalla el resultado de contar las palabras existentes en el fichero dado. Una vez comprobamos el funcionamiento del código MapReduce de esta forma, podemos ejecutarlo en nuestro cluster Hadoop con la siguiente orden:

```
hadoop jar  /usr/lib/hadoop-mapreduce/hadoop-streaming.jar
-file /home/cloudera/examplemapreduce/mapper01.py
-mapper /home/cloudera/examplemapreduce/mapper01.py
-file /home/cloudera/examplemapreduce/reducer01.py
-reducer /home/cloudera/examplemapreduce/reducer01.py
-input /user/cloudera/examplemapreduce/*
-output /user/cloudera/examplemapreduce-output
```

En la orden de arriba, indicamos a Hadoop que se ejecute llamando a la librería Hadoop-streaming, en la ruta especificada. Después, le indicamos la ruta a los ficheros que implementan las funciones map y reduce (en este ejemplo, hemos creado una carpeta llamada **/home/cloudera/examplemapreduce en el sistema de archivos local** en la cual hemos dejado los ficheros que implementan las funciones map y reduce). Seguidamente, indicamos dónde están los ficheros de datos sobre los que deseamos ejecutar este trabajo MapReduce, y que se deben encontrar en la carpeta **/user/cloudera/examplemapreduce que hemos creado previamente en HDFS.** Para finalizar, le indicamos a Hadoop que los resultados se deben escribir en la carpeta **/user/cloudera/examplemapreduce-output en HDFS.**

Al ejecutar esta orden en la máquina virtual, obtendremos como resultado dentro de la carpeta carpeta **/user/cloudera/examplemapreduce-output** dos archivos, _SUCCESS y part-00000. El primero tiene tamaño 0 e indica el resultado de la

operación (que ha sido un éxito), mientras que el segundo contiene la cuenta de las palabras.

5.2 Contador de palabras con Jupyter notebooks

Ahora vamos a ver la ejecución del programa del contador de palabras en un Jupyter notebook. Para ello, se ha preparado el notebook llamado *ejemplo-inicial_wordcount.ipynb*, que se puede descargar del curso virtual.

Tras descargar el notebook del ejemplo inicial en nuestra máquina, deberemos copiarlo en la carpeta la carpeta raíz de jupyter según se ha detallado en el documento de preparación del entorno de trabajo (en el entorno basado en contenedores sería lo que hayamos puesto en lugar de <CARPETA LOCAL> en el fichero yml, mientras que en el entorno basado en máquina virtual será el directorio donde hayamos arrancado el servidor jupyter). Para ello se utiliza la orden siguiente:

```
$ sudo cp ejemplo-inicial_wordcount.ipynb /media/notebooks
```

Ahora, tras arrancar el entorno de trabajo, abriremos el notebook clicando sobre él, y se mostrará como indica la Figura 18.

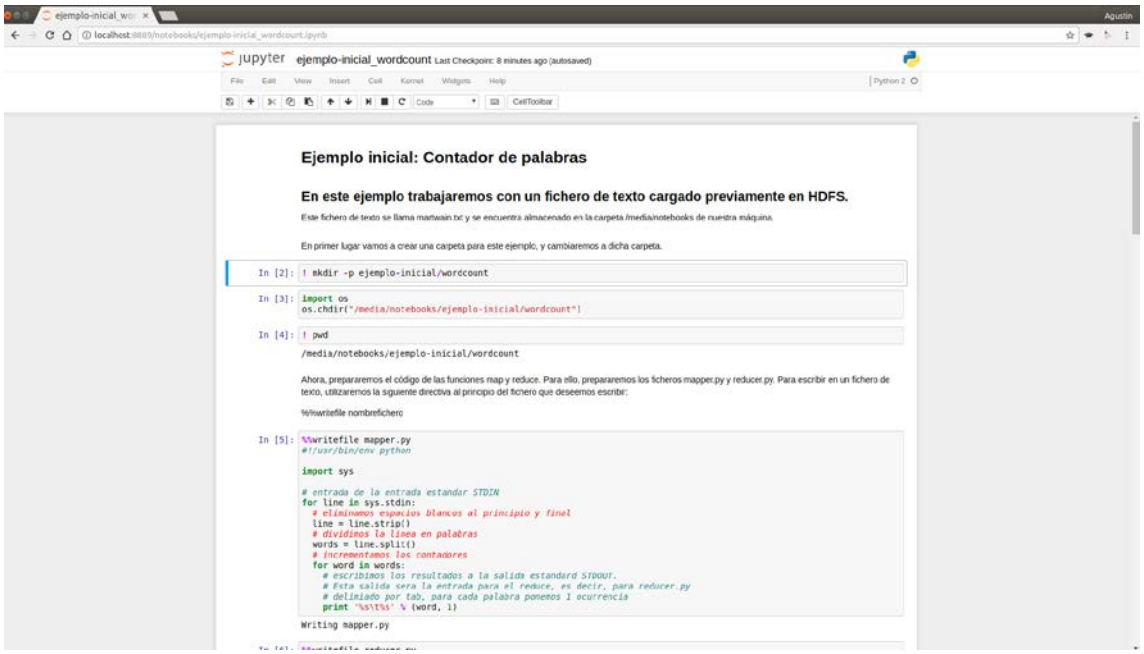

Figura 18. Notebook ejemplo inicial.

Para ejecutar el código, debemos seleccionar la celda que se desee ejecutar y a continuación pulsar las teclas “Mayúsculas Intro” al mismo tiempo. Iremos ejecutando celda a celda comprendiendo lo que hace cada una de ellas.

Para escribir en un fichero nuestro código para las funciones map y reduce, utilizaremos la directiva *%%writefile nombreFichero* al comienzo de la celda. Así, al ejecutar dicha celda, se creará el fichero llamado *nombreFichero* con el contenido definido dentro de la celda.

Una vez hemos preparado los ficheros que contienen el código de las funciones map y reduce, procederemos a ejecutarlo de forma similar a como se ha visto en la sección anterior, solamente hay que añadir el símbolo ! al comienzo de la orden, como se muestra en la Figura 19.

```
In [12]: ! cat /media/notebooks/marktwain.txt | python mapper.py | sort | python reducer.py > salidawordcount
```

Figura 19. Simulación de MapReduce en un notebook.

De la misma forma, podremos ejecutar nuestro código en Hadoop con la misma orden vista anteriormente, tal como se muestra en la Figura 20.

```
! hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar -file mapper.py -mapper mapper.py \
-file reducer.py -reducer reducer.py -input /tmp/marktwain.txt -output /tmp/salida-wordcount
```

Figura 20. Ejecución de código MapReduce desde un notebook sobre un cluster Hadoop.

5.3 Ejemplo avanzado: Yelp Challenge

5.3.1 Preparando los datos

En este ejemplo más avanzado vamos a utilizar datos de Yelp, que se pueden descargar del siguiente enlace:

https://unedo365-my.sharepoint.com/:u:/g/personal/accaminero_scc_uned_es/EYPTYHMlRYZLp06YkNyZWQoBYvTrQmPwOemyr08E6CxYGA?e=72Gnvf

Desde este enlace nos descargaremos un fichero comprimido que contiene, entre otros los siguientes ficheros:

- *yelp_academic_dataset_business.json* → fichero que contiene información de negocios.
- *yelp_academic_dataset_review.json* → fichero que contiene información de opiniones.

Los datos de los negocios que tenemos son los siguientes:

- "city", "review_count", "name", "neighborhoods", "type", "business_id", "full_address", "hours", "state", "longitude", "stars", "latitude", "attributes", "open", "categories".

- Los datos más importantes con los que trabajaremos en este ejemplo son:
  - review_count: contador de opiniones.
  - Latitude, altitude: coordenadas geográficas.
  - Business_id: un identificador para el negocio.
  - Categories: el tipo de negocio (ej. Restaurante, ...).

También tenemos datos de opiniones:

1. "funny", "useful", "cool", "user_id", "review_id", "text", "business_id", "stars", "date", "type"
   - Los datos más importantes con los que trabajaremos en este ejemplo son:
     - Text: el texto de la opinión, que refleja la descripción que el usuario de Yelp ha realizado sobre ese negocio.
     - Cool: un número entero que cuanto más alto, mejor es la valoración de este negocio.

Los datos de Yelp Challenge son datos en formato JSON, es decir, que tienen una estructura como la que sigue, en la que para cada campo tenemos su nombre seguido de su valor:

- Ejemplo de negocios:
  - **{"business_id"**: "vcNAWiLM4dR7D2nwwJ7nCA", **"full_address"**: "4840 E Indian School Rd\nSte 101\nPhoenix, AZ 85018", **"categories"**: ["Doctors", "Health & Medical"], **"city"**: "Phoenix", "review_count": 9, **"name"**: "Eric Goldberg, MD", **"longitude"**: -111.98375799999999, **"stars"**: 3.5, **"latitude"**: 33.499313000000001}
- Ejemplo de opiniones:
  - {**"votes"**: {**"funny"**: 0, **"useful"**: 2, **"cool"**: 1}, **"user_id"**: "Xqd0DzHaiyRqVH3WRG7hzg", **"stars"**: 5, **"date"**: "2007-05-17", **"text"**: "dr. goldberg offers everything i look for in a general practitioner", **"type"**: "review", **"business_id"**: "vcNAWiLM4dR7D2nwwJ7nCA"}

5.3.2 Trabajando con Hive

En este apartado vamos a trabajar con Hive, la herramienta del ecosistema de Hadoop que proporciona un interfaz con una sintaxis similar al lenguaje de consultas de bases de datos relacionales SQL. Utilizaremos también el interfaz de

línea de comandos de Hadoop, así como su interfaz gráfico HUE, que nos proporcionará una forma intuitiva para cargar ficheros en el cluster, redactar y ejecutar programas así como ver los resultados.

Utilizaremos estas herramientas para procesar datos de Yelp. Para ello, seguiremos los siguientes pasos.

1. En primer lugar, antes de nada deberemos preprocesar los datos de Yelp. La estructura en formato JSON de estos datos no es idónea para el trabajo con Hive, debido a la existencia de campos anidados (el campo "votes" contiene los subcampos "funny", "useful" y "cool") así como la presencia de saltos de línea dentro de algunos campos de texto en las estructuras JSON tanto en el fichero de opiniones como el de negocios.

   Para solucionar estos problemas, ejecutaremos el script llamado convert2.py que se ha proporcionado. Esto se realiza desde un terminal, tecleando la siguiente orden:

   ```
   ./convert2.py
   ```

   Tras esto, tendremos los nuevos ficheros siguientes:
   - yelp_academic_dataset_business_*clean2*.json
   - yelp_academic_dataset_review_*clean2*.json

2. Ahora iniciaremos HUE, para lo cual deberemos ejecutar el navegador de Internet. En la página de inicio del navegador de Internet del entorno virtual, haz click en el botón que dice "Launch Hue UI". Para conectarte a HUE deberás utilizar los siguientes datos de acceso:

   Usuario: cloudera

   Clave: cloudera

3. Ahora cargaremos los dos ficheros generados en el paso 1 en nuestro cluster de Hadoop. Este punto se realiza desde la opción "File Browser" situada arriba a la derecha en el interfaz de HUE.

4. Seguidamente, crearemos tablas partiendo de estos ficheros, una llamada "business" y otra llamada "reviews". Este paso se realiza desde "Data Browsers" → "Metastore tables". Debemos prestar atención a que las columnas de las tablas tengan los nombres correctos, dichos nombres son los que se mencionan en el apartado 5.3.1 de este documento. Para nombrar las columnas, clica en "Bulk edit column names" y pega los nombres de las columnas correspondientes a la tabla que estás creando.

5. Una vez las tablas estén creadas con los datos correctos, crea una consulta utilizando el editor de Hive (esto se encuentra en "Query editors"-> "Hive") con el siguiente contenido:

```
SELECT name, review_count
FROM business
ORDER BY review_count DESC
LIMIT 25
```

Esta consulta devuelve el nombre y el contador de opiniones de los 25 negocios que mayor número de opiniones tengan. Para ejecutar esta consulta clicaremos en "Execute".

6. Tras ejecutarla, si vamos a la opción "Chart" podremos ver sus resultados graficados de varias formas diferentes.

7. Ahora ejecutaremos otra consulta algo más compleja que nos devolverá, entre otra información, la localización geográfica de los 25 restaurantes con mejores valoraciones. La consulta a ejecutar es la siguiente:

```
SELECT r.business_id, name, SUM(cool) AS coolness, longitude, latitude
FROM review r JOIN business b
ON (r.business_id = b.business_id)
WHERE categories LIKE '%Restaurants%'
GROUP BY r.business_id, name, longitude, latitude
ORDER BY coolness DESC LIMIT 25
```

Tras clicar en "Execute", veremos sus resultados. En este caso, si vamos a la opción "Chart", y dentro de ella, "Map", podremos visualizar los resultados en forma de mapa. Para ello, deberemos indicar qué campos resultado de la consulta se deben utilizar para realizar la visualización en el mapa, es decir, la longitud y latitud geográficas. Esto es, en el desplegable "Latitude" seleccionamos "Latitude" y en el desplegable "Longitude" seleccionamos "Longitude". El resultado de esta consulta se muestra en Figura 21.

Con el fin de aclarar los conceptos de Hadoop así como para presentar varios ejemplos de funcionamiento, tanto utilizando el interfaz gráfico HUE como con la línea de comandos, hemos preparado una serie de vídeos que se encuentran accesibles desde el curso virtual.

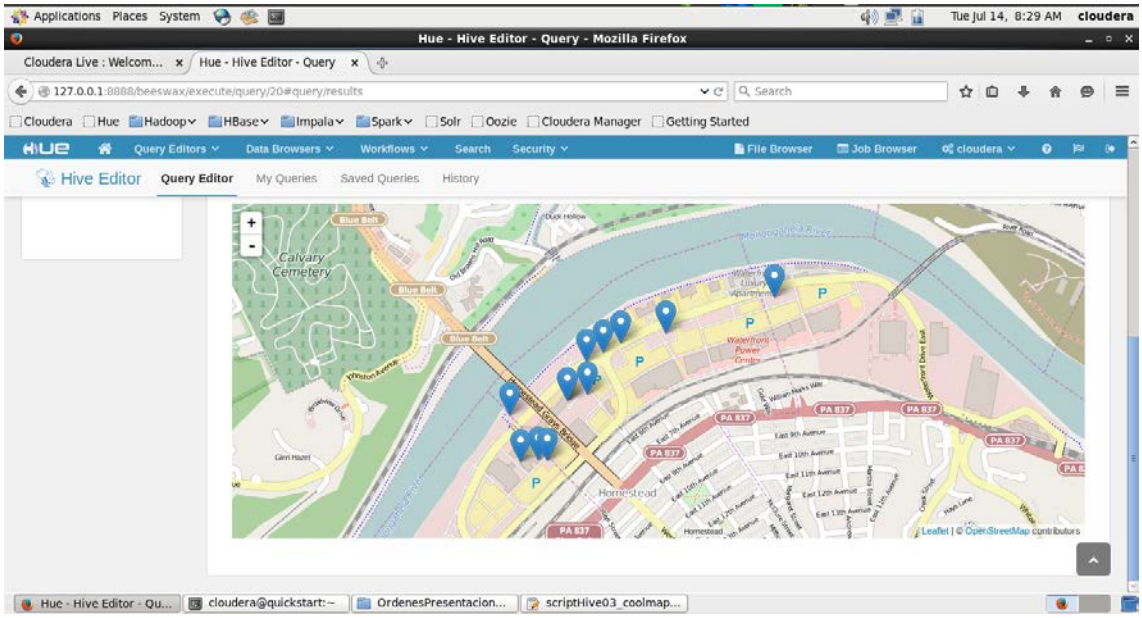

Figura 21. Visualización en el mapa del resultado de una consulta Hive

5.4 Trabajando con la línea de comandos de Hive

Vamos ahora a ver un ejemplo de creación de una base de datos de futbolistas partiendo de un fichero de datos dado. En este ejemplo utilizaremos la línea de comandos. Para trabajar con Hive desde la línea de comandos, en primer lugar abriremos un terminal en el entorno de trabajo, tras lo cual introduciremos las siguientes órdenes:

```
$ hadoop fs –mkdir datoshive
$ hadoop fs -put futbolistas.txt datoshive
```

Las órdenes de arriba sirven para crear en HDFS una carpeta y para cargar en dicha carpeta el fichero de datos de futbolistas. Tras esto, ejecutaremos el programa beeline, que es la interfaz de línea de comandos de Hive, y nos conectaremos con el servidor de Hive local, utilizando las siguientes órdenes:

```
$ beeline

!connect jdbc:hive2://localhost:10000/default
```

Cuando nos pida un usuario y clave, pulsaremos la tecla Intro del teclado, y estaremos listos para comenzar a introducir comandos. Ahora introduciremos las siguientes órdenes para crear la base de datos, la tabla y cargar datos en dicha tabla.

```
create database if not exists futbolistasdb
Comment 'Base de datos de futbolistas'
Location '/user/cloudera/futbolistasdb'
With dbproperties ('Creada por' = 'User', 'Creada el' = 3-Sep-2018');

use futbolistasdb;

CREATE TABLE IF NOT EXISTS futbolistas
(
 nombre string,
 posiciones ARRAY<string>,
 temporada_equipo ARRAY<STRUCT<temporada:string,equipo:string>>,
 habilidad_puntuacion MAP<string,int>
)
COMMENT 'Tabla de futbolistas'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
COLLECTION ITEMS TERMINATED BY ','
MAP KEYS TERMINATED BY ':';

LOAD DATA INPATH '/user/cloudera/datoshive/futbolistas.txt' INTO TABLE futbolistas;
```

5.5 Trabajando con Jupyter notebooks y Hive

De forma similar al ejemplo de MapReduce con Jupyter notebooks, podemos incluir comandos de Hive dentro de celdas de un notebook. En el curso virtual se ha publicado el archivo hive-1erpasos.ipynb que contiene el código visto arriba para la creación de la base de datos de futbolistas.

Hay que tener en cuenta que las órdenes de Hive se deben preceder del símbolo ! y que se pueden ejecutar órdenes de Hive desde un fichero o directamente. En el primer caso, se realiza mediante el parámetro -f como se muestra en la

```
! beeline -u "jdbc:hive2://localhost:10000/default" -f ejerciciohive.hql
```

Figura 22. Ejecución de fichero con comandos Hive.

El segundo caso se realiza con el parámetro -e como se muestra en la Figura 23.

```
! beeline -u "jdbc:hive2://localhost:10000/default" -e "use futbolistasdb; show tables;"
```

Figura 23. Ejecución de comandos Hive.

6 Ejercicio de evaluación

En este trabajo se van a utilizar ficheros (**disponibles en el curso virtual**) descargados de la web del Banco Mundial y contienen información sobre superficie de tierra cultivable por persona y porcentaje de población rural, en cada uno diferenciando por años desde 1960 y por país. Son los archivos siguientes:

- API_AG.LND.ARBL.HA.PC_DS2_es_csv_v2_2593504.csv.
- API_SP.RUR.TOTL.ZS_DS2_es_csv_v2_2601419.csv.

Cada uno de los ficheros de arriba tiene los campos siguientes:

- Country Name: Nombre del país.
- Country Code: Código identificador del país.
- Indicator Name: Nombre del indicador que se muestra en el fichero.
- Indicator Code: Código identificador del indicador.

Abajo se muestra un fragmento del fichero API_SP.RUR.TOTL.ZS_DS2_es_csv_v2_2601419.csv, el otro tiene una estructura similar.

Country Name	Country Code	Indicator Name	Indicator Code	1960	1961
Aruba	ABW	Población rural (% de la población total)	SP.RUR.TOTL.ZS	49.224	49.239

Partiendo de estos datos, desarrolla las órdenes de HiveQL que implementen las siguientes tareas.

1. (2 puntos) Crea las tablas necesarias para almacenar datos. Pueden ser internas o externas en función de los datos que se desee. La decisión de interna o externa debe estar razonada.

2. (1 puntos) Importa los datos en las tablas creadas.

3. (2 puntos) Crea una vista sobre las tablas creadas. Esta vista tendrá para cada país, su nombre, código, porcentaje de población rural en 2018 y superficie de tierra cultivable en 2018.

4. (3 puntos) Crea las consultas de Hive necesarias para responder las siguientes cuestiones:
    - ¿Cuál es el porcentaje de tierra cultivable de Suiza en 2018?
    - ¿Cuál es el porcentaje de tierra cultivable del país que tiene menor población rural en 2010?
    - ¿Cuáles son los cinco países con mayor población rural en 2018?

5. (2 puntos) Desarrolla el código MapReduce en Python que implemente la primera consulta del apartado anterior.

6. (2 puntos extra) Si realizas todos los apartados anteriores, puedes sumar hasta 2 puntos extra de la siguiente forma. Encuentra uno o más ficheros de datos, impórtalos en una o más tablas de Hive, y realiza alguna consulta sobre dichos datos. Para encontrar datasets se pueden utilizar las referencias bibliográficas que se indican al final de este documento. Se valorará la originalidad y la complejidad de los datos y consultas.

6.1 Aclaraciones

Para realizar las implementaciones de esta práctica, se ofrecen los siguientes recursos:

- Entorno de trabajo basado en los recursos oficiales de Cloudera. Para la utilización de este entorno se ha publicado un documento en el curso virtual que detalla las diferentes opciones disponibles junto con su proceso de descarga y despliegue.

En el caso de que el ordenador de que se disponga no tenga potencia para ejecutar el entorno de trabajo de Cloudera, los desarrollos de este ejercicio se pueden implementar utilizando los siguientes recursos:

- Máquina virtual Ubuntu con el sistema gestor de bases de datos PostgreSQL instalado. En PostgreSQL se deberán crear las tablas y consultas que se indican en los puntos 1 al 4. En Python se puede implementar el código para el punto 5. En este caso no es necesario utilizar Jupyter notebooks – se puede presentar el código desarrollado en documentos de texto sin formato listos para su ejecución y las explicaciones en una memoria aparte que incluya pantallazos que demuestren el correcto funcionamiento de los desarrollos.

La práctica se valorará igualmente siempre que se realice utilizando cualquiera de las opciones indicadas en este enunciado.

Para el desarrollo del apartado 5, se recomienda la lectura de la referencia [10], en concreto, el capítulo sobre patrones de filtrado.

El apartado 6 solamente se evaluará si se realizan los apartados 1 al 5, aunque no es esencial que los apartados 1 al 5 estén perfectos para proceder a evaluar el apartado 6. Para encontrar datasets para el apartado 6, se pueden utilizar las referencias bibliográficas que se indican al final de este documento.

7 Detalles de la evaluación

Una vez realizadas las tareas explicadas en la sección anterior, la forma de evaluar el trabajo se hará en base a un Jupyter notebook que contenga tanto el código desarrollado como todas las explicaciones necesarias. El notebook que se envíe para su evaluación puede estar basado en alguno de los notebook que se han presentado como ejemplo. Las celdas de código deben estar ejecutadas, de forma que en al abrir el notebook los docentes puedan ver el resultado de las ejecuciones realizadas por el/la estudiante.

Opcionalmente, junto con el notebook se puede entregar una memoria que explique los desarrollos realizados. En el caso de que se incluya, esta memoria se debe enviar en formato pdf, pero también se deben incluir sus fuentes (fichero doc, docx, odt...). En el caso de utilizar la máquina virtual con PostgreSQL, no es necesario utilizar notebooks, sino que, junto a la memoria explicativa se debe incluir el código en documentos de texto sin formato, listos para su ejecución.

Consideraciones generales:

- Se deberá indicar en qué entorno de trabajo se ha realizado la práctica.
- No es necesario explicar el proceso de instalación o despliegue del entorno de trabajo.
- Si se entrega **memoria**, se deben incluir **pantallazos** que reflejen el correcto funcionamiento de los desarrollos.
- Si se entregan **notebooks,** tienen que estar ejecutados, y se deben incluir las órdenes necesarias para demostrar el correcto funcionamiento de los desarrollos. Por ejemplo, para demostrar que una tabla se ha creado y cargado de datos correctamente, habría que ejecutar un select que devuelva algunos de los contenidos de la tabla (no hacer select * from table, utilizar la cláusula limit). **Los notebooks no deben contener consultas select * from tabla** ya que ralentizan mucho la carga del notebook.
- Se debe utilizar la **vista** en las consultas que sean necesarias.

- Se valorará positivamente que el entregable contenga un apartado final donde se explique la **opinión del/la estudiante sobre esta práctica**, así como puntos fuertes y/o débiles de la práctica. Recuerda que, aparte de esto,al final del curso se publicarán encuestas para evaluar la calidad de la docencia de la asignatura.
- Sobre el apartado 6, se valorará la originalidad y la complejidad de los datos y consultas.

Este material se deberá incluir en un fichero comprimido y enviado a través del curso virtual dentro de los plazos establecidos para su entrega. El nombre de dicho fichero comprimido deberá tener la siguiente estructura: *CentroAsociado-ApellidosNombre.zip*, donde *CentroAsociado*, *Apellidos* y *Nombre* deben sustituirse por los valores correspondientes para cada estudiante. Un nombre correcto es, por ejemplo, el siguiente:

Albacete-CamineroHerraezAgustin.zip

Solamente habrá un intento para entregar la práctica y se valorarán negativamente los defectos de forma al realizar la entrega. En estos casos, se considerará la práctica como no entregada, y no será posible volver a entregarla. Se recomienda no dejar la entrega para última hora, ya que pueden haber imprevistos (por ejemplo, fallos técnicos, ...) que impidan su entrega en los plazos establecidos.

Esta práctica tiene carácter optativo por lo que se puede aprobar la asignatura sin realizarla. Además, tiene un valor de 1.5 puntos en la nota final de la asignatura y solamente se corregirá en el cuatrimestre en el que se imparte la asignatura.

8 Notas y referencias de interés

8.1 Notas de interés

Con el fin de aclarar el funcionamiento de Hadoop, así como para introducir algunos ejemplos más de uso de Hadoop tanto mediante línea de comandos como mediante el interfaz gráfico, hemos preparado una serie de vídeos explicativos. Dichos vídeos se encuentran accesibles desde el curso virtual.

Para el acceso a la biblioteca digital de O'Reilly, es necesario crear una cuenta en https://learning.oreilly.com/home/ utilizando la dirección de correo electrónico de alumno de la UNED, y siguiendo los pasos que se detallan en dicho enlace.

8.2 Referencias de interés

1. Página de la Wikipedia sobre Big Data. Disponible en https://es.wikipedia.org/wiki/Big_data
2. Página web de Apache Hadoop. Disponible en: https://hadoop.apache.org/
3. Página web de Apache Hive. Disponible en: https://hive.apache.org/
4. Página web de HUE. Disponible en: http://gethue.com/
5. Programming Hive. Data Warehouse and Query Language for Hadoop Autores: Edward Capriolo, Dean Wampler, Jason Rutherglen. Editorial O'Reilly Media. Año: 2012. URL (solamente funciona tras autenticarse en O'Reilly):https://learning.oreilly.com/library/view/programming-hive/9781449326944/
6. Manual de Python. Disponible en: http://www.diveintopython.net/
7. Algunos tutoriales de Hadoop. Disponibles en: https://github.com/romainr/hadoop-tutorials-examples
8. Apache Hive Cookbook. Autores: Hanish Bansal; Saurabh Chauhan; Shrey Mehrotra. Editorial: Packt Publishing. Año: 2016. URL (solamente funciona tras autenticarse en O'Reilly): https://learning.oreilly.com/library/view/apache-hive-cookbook/9781782161080/.
9. Practical Hive: A Guide to Hadoop's Data Warehouse System. Autores: Scott Shaw; Andreas François Vermeulen; Ankur Gupta; David Kjerrumgaard.Editorial: Apress. Año: 2016. URL (solamente funciona tras autenticarse en O'Reilly): https://learning.oreilly.com/library/view/practical-hive-a/9781484202715/
10. MapReduce Design Patterns. Autores: Donald Miner; Adam Shook Editorial: O'Reilly Media, Inc. Año, 2012 ISBN-13 en papel: 978-1-4493-2717-0 URL (solamente funciona tras autenticarse en O'Reilly): https://learning.oreilly.com/library/view/mapreduce-design-patterns/9781449341954/
11. Dataset search de Google. Disponible en https://datasetsearch.research.google.com/
12. Datasets de Kaggle. Disponible en https://www.kaggle.com/datasets
13. Datos del Banco Mundial. Disponibles en https://datos.bancomundial.org/