UD 4 - Apache Hadoop - MapReduce

1. Introducción

MapReduce

Hadoop MapReduce es un **framework** para escribir fácilmente aplicaciones que procesan **grandes cantidades de datos** en **paralelo** en **grandes clústeres** (miles de nodos) de **hardware commodity** de manera **confiable** y **tolerante a fallos**.

MapReduce está diseñado para poder procesar grandes cantidades de datos, ya que sigue una filosofía **Divide y Vencerás (DYV)**, que consiste en que para resolver un problema complejo, la mejor forma de hacerlo es dividirlo en fragmentos muy pequeños que pueden ser solucionados de forma independiente, resolverlo por separado e ir construyendo con las soluciones parciales la solución final.

Para el caso del procesamiento de datos de mucho volumen, la aproximación Divide y Vencerás que hace MapReduce consiste en dividir todo el conjunto de datos de entrada en pequeños fragmentos, procesarlos por separado, e ir agrupando los resultados parciales.

Para ello, usa un paradigma de programación funcional en dos fases, la de **mapeo** y la de **reducción**, y define el algoritmo que utiliza Hadoop para paralelizar las tareas. Un algoritmo MapReduce divide los datos, los procesa en paralelo, los reordena, combina y agrega de vuelta los resultados mediante un formato **clave/valor**.

Note

Hadoop Aunque MapReduce ha sido utilizado ampliamente por toda la comunidad de desarrolladores o proyectos Big Data, la realidad es que hoy en día cada vez se usa menos, como puede verse en este gráfico de tendencia de búsquedas en Google:

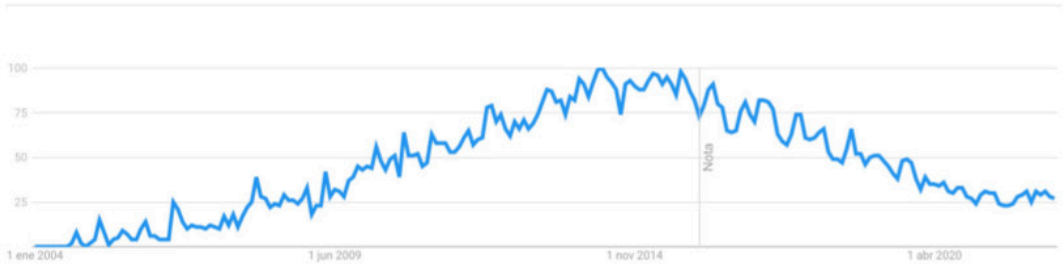

Figura4.1_MapReduce: Historial de Búsquedas en Google. (Fuente: Ministerio de Educación)

Sin embargo, es importante entender y comprender bien MapReduce por dos motivos:

- Utiliza un modelo de programación que es común en otras herramientas o frameworks Big Data.
- Muchas herramientas que se ejecutan sobre Hadoop, como puede ser el caso de Hive, pese a que ofrecen funcionalidad de alto nivel, como puede ser la capacidad de hacer consultas en formato SQL, por debajo ejecutan MapReduce. Entender MapReduce es fundamental para poder depurar o resolver problemas en la ejecución de trabajos con este tipo de herramientas.

Estos subprocesos asociados a la tarea se ejecutan de manera **distribuida**, en diferentes nodos de procesamiento o esclavos. Para controlar y gestionar su ejecución, existe un proceso **Master** o **Job Tracker**. También es el encargado de aceptar los nuevos trabajos enviados al sistema por los clientes.

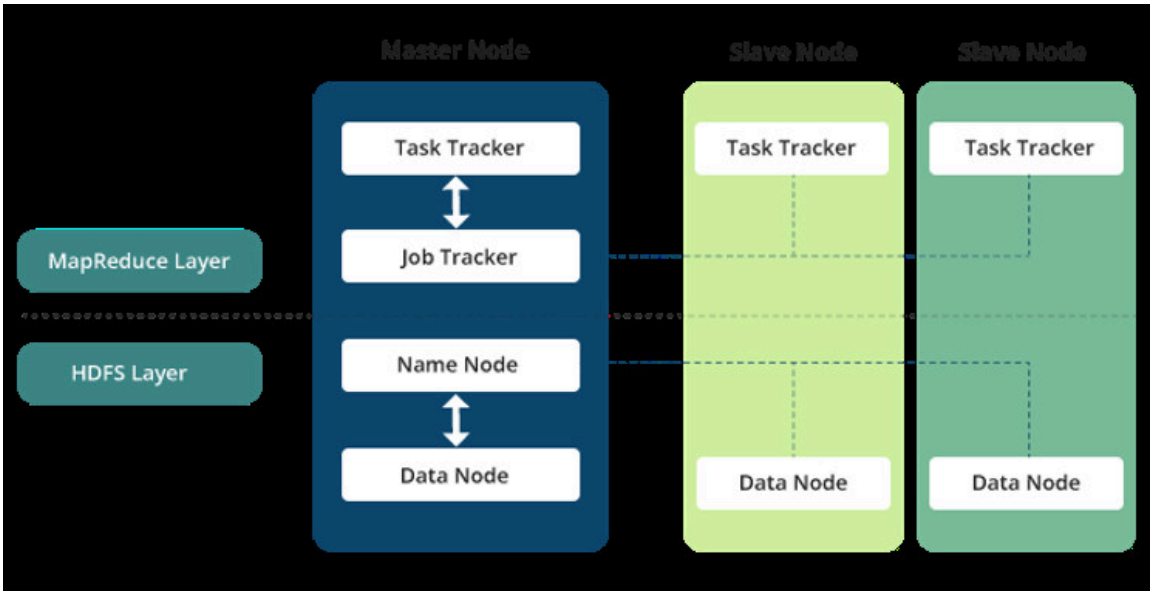

Figura4.2_MapReduce WorkFlow. (Fuente: Apache Hadoop)

Este sistema de procesamiento se apoya en tecnologías de almacenamiento de datos distribuidas, en cuyos nodos se ejecutan estas operaciones de tipo map y reduce. El sistema de ficheros distribuido de Hadoop es HDFS (Hadoop Distributed File System), encargado de almacenar los ficheros divididos en bloques de datos. HDFS proporciona la división previa de los datos en bloques que necesita MapReduce para ejecutar. Los resultados del procesamiento se pueden almacenar en el mismo sistema de almacenamiento o bien en una base de datos o sistema externo.

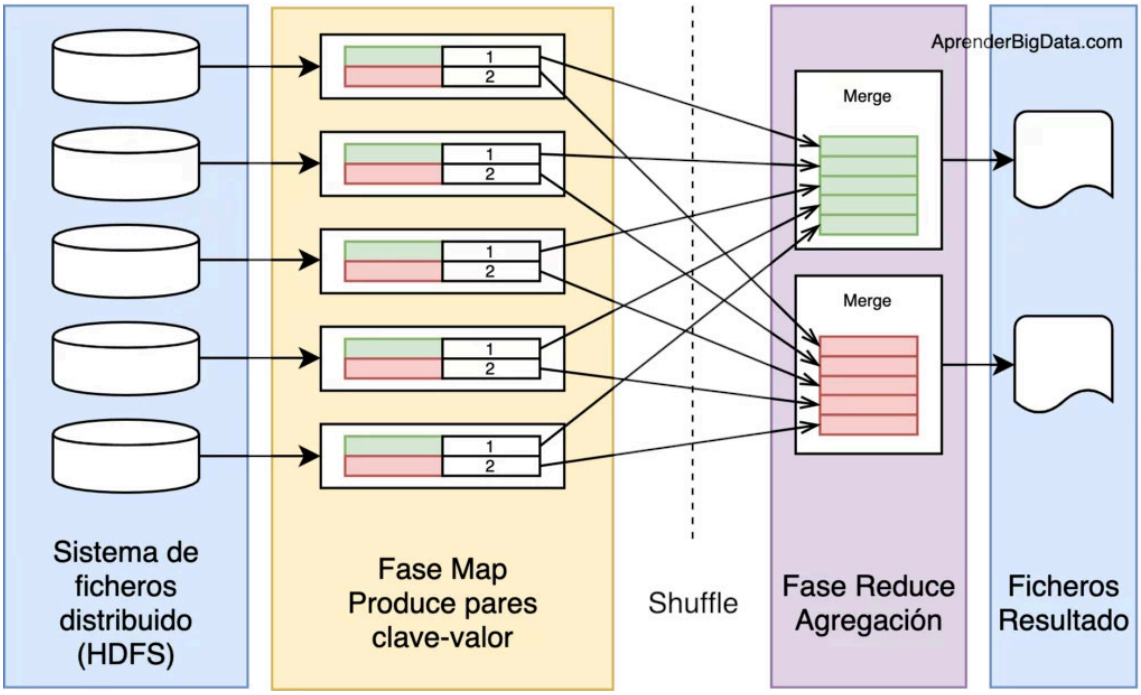

Figura4.3_MapReduce_Esquema de fases de MapReduce. (Fuente: AprenderBigData)

2. Fases y Funcionamiento

Un trabajo de MapReduce se compone de cinco etapas distintas, ejecutadas en orden:

1. Envío del trabajo, aceptación y distribución en el clúster.
2. Ejecución de la fase **map**: Se ejecuta en subtareas llamadas *mappers*. Estos componentes son los responsables de generar pares *clave-valor* ***filtrando, agrupando, ordenando o transformando los datos originales***. Los pares de datos intermedios, no se almacenan en HDFS.
3. Ejecución de la fase **shuffle**: Puede no ser necesaria. Es el paso intermedio entre Map y Reduce que ayuda a recoger los datos y ordenarlos de manera conveniente para el procesamiento. Con esta fase, se pretende agregar las ocurrencias repetidas en cada uno de los mappers.
4. Ejecución de la fase **order**. Ordena y/o baraja los datos a partir de la clave.
5. Ejecución de la fase **reduce**: Gestiona la agregación de los valores producidos por todos los mappers del sistema (o por la fase shuffle) de tipo clave-valor en función de su clave. Por último, cada reducer genera su fichero de salida de forma independiente, generalmente escrito en HDFS.

De todas estas fases debes saber que el programador sólo suele programar la fase ***map*** y ***reduce***, siendo el resto de fases ejecutadas de forma automática por MapReduce en base a los parámetros de configuración.

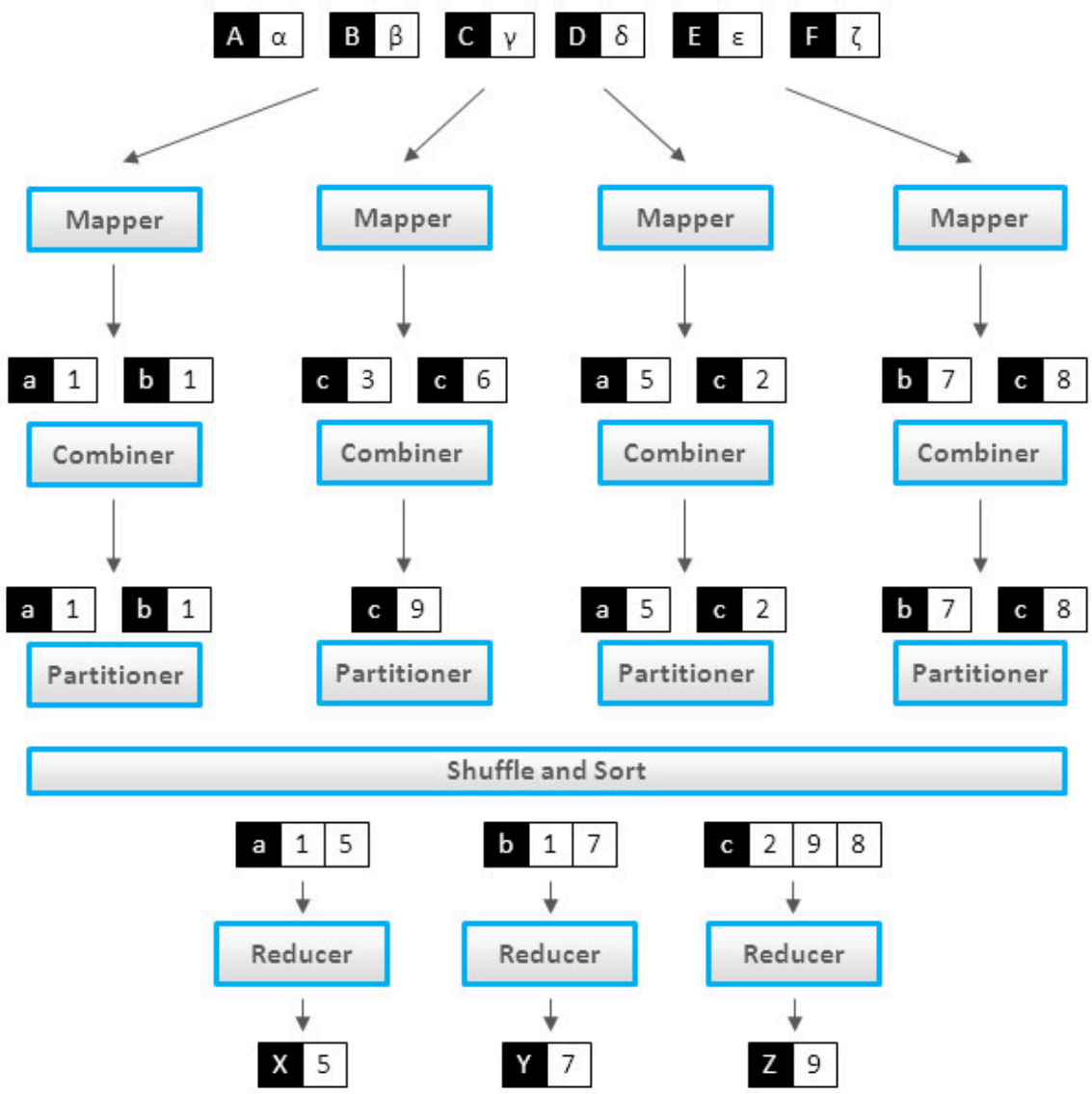

Figura4.4_MapReduce_Framework MapReduce. (Fuente: Ministerio de Educación)

En un trabajo Hadoop MapReduce, se dividen los datos de entrada en fragmentos independientes que son procesados por los mappers en paralelo. A continuación, se ordenan los resultados del map, que son la entrada para los reducers. Generalmente, las entradas y salidas de los trabajos se almacenan en un sistema de ficheros, siendo los nodos de almacenamiento y de cómputo los mismos. También es muy común que la lógica de la aplicación no se pueda descomponer en una única ejecución de MapReduce, por lo que se encadenan varias de estas fases, tratando los resultados de una como entrada para los mappers de la siguiente fase.

Esta característica, permite ejecutar las tareas de cada fragmento en el nodo donde se almacena, reduciendo el tiempo de acceso a los datos y los movimientos entre nodos del clúster.

3. WebUI

Podemos observar los trabajos realizados desde la Interfaz WebUI en el puerto 8088

`http://bda-iesgrancapitan:8088`

> **WebUI**
>
> La configuración y visualización de las WebUIs las veremos en el siguiente punto.

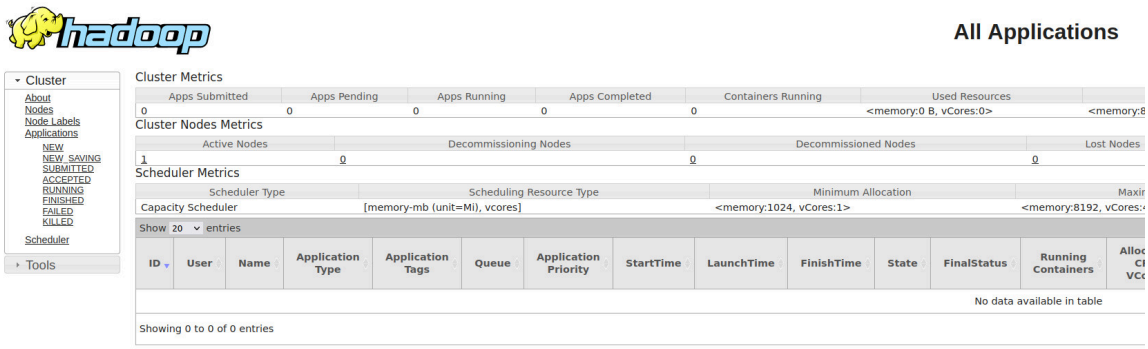

Figura4.5_MapReduce_WebUI. (Fuente: Propia)

4. Ejemplos

Para explicar el funcionamiento de MapReduce, se van a utilizar los siguientes ejemplos:

4.1 Ejemplo 1

El siguiente gráfico muestra un ejemplo de una empresa que fabrica juguetes de colores. Cuando un cliente compra un juguete desde la página web, el pedido se almacena como un fichero en Hadoop con los colores de los juguetes adquiridos. Para averiguar cuantas unidades de cada color debe preparar la fábrica, se emplea un algoritmo MapReduce para contar los colores:

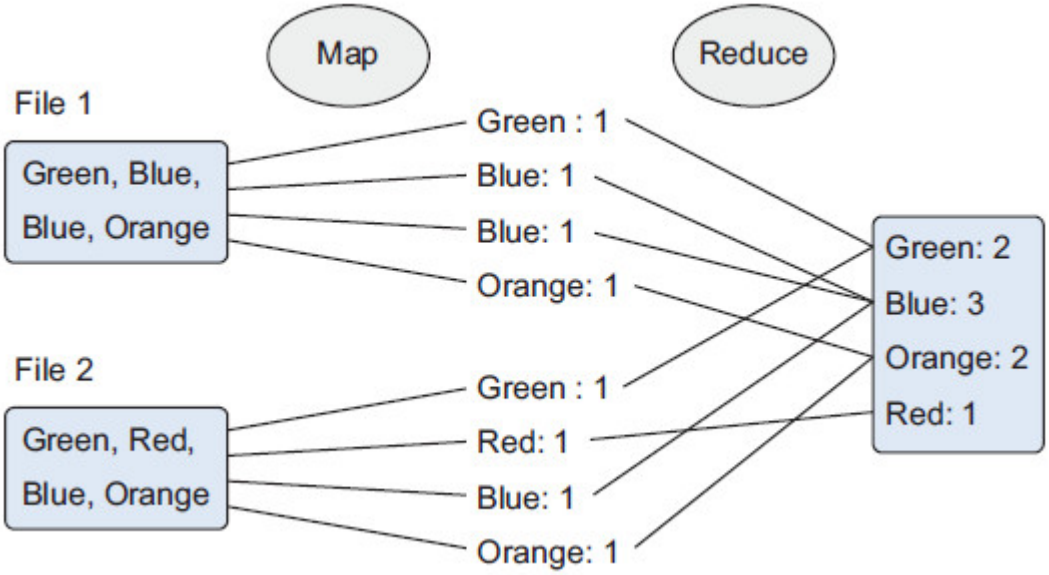

Figura4.6_MapReduce_Ejemplo1_Map y Reduce. (Fuente: Ministerio de Educación)

Siguiendo MapReduce:

- Fase de *mapeo* **(Map)**: Los documentos se parten en pares de clave/valor. Hasta que no se reduzca, podemos tener muchos duplicados.
- Fase de *reducción* **(Reduce)**: Es en cierta medida similar a un "group by" de SQL. Las ocurrencias similares se agrupan, y dependiendo de la función de reducción, se puede crear un resultado diferente. En nuestro ejemplo queremos contar los colores, y eso es lo que devuelve nuestra función.

Es un proceso de procesamiento costoso. Los pasos serían los siguientes:

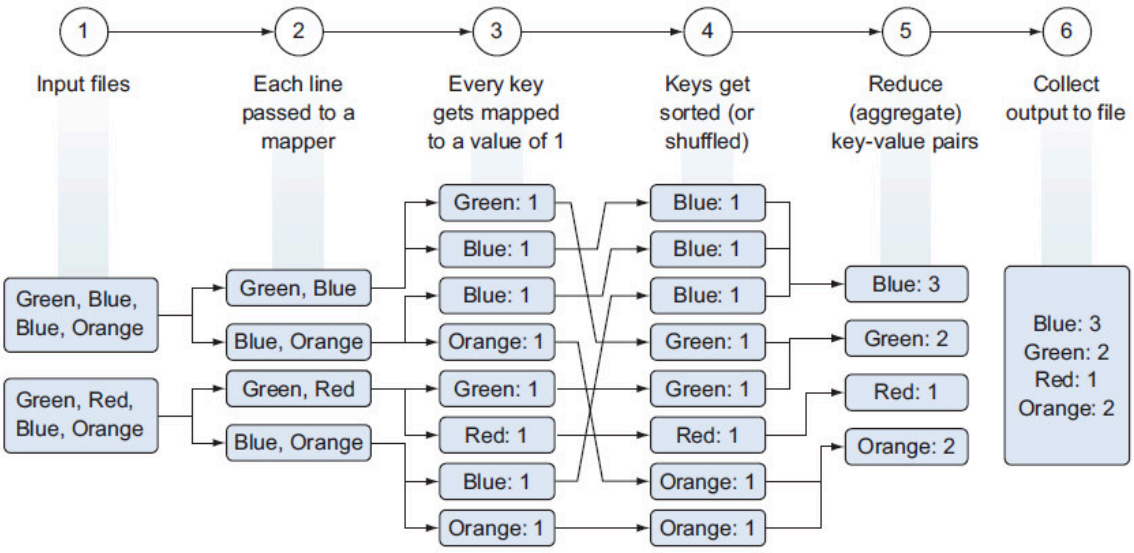

Figura4.7_MapReduce_Ejemplo1 Detallado. (Fuente: Ministerio de Educación)

1. Lectura desde HDFS de los ficheros de entrada.
2. Pasar cada línea de forma separada al mapeador, teniendo tantos mapeadores como bloques de datos que tengamos.
3. El mapeador parsea los colores (claves) de cada fichero y produce un nuevo fichero para cada color con el número de ocurrencias encontradas (valor), es decir, mapea una clave (color) con un valor (número de ocurrencias).
4. Para facilitar la agregación, se ordenan y/o barajan los datos a partir de la clave.
5. La fase de reducción suma las ocurrencias de cada color y genera un fichero por clave con el total de cada color.
6. Las claves se unen en un único fichero de salida que se persiste en HDFS.

4.2 Ejemplo 2

Imagina que tenemos un fichero de muchos terabytes de datos con todas las cotizaciones de todas las empresas de todas las bolsas del mundo desde hace 30 años, con una cotización cada minuto. Cada línea del fichero tiene el siguiente formato:

`Fecha y hora (día/mes/año hora:minutos:segundos);nombre de la empresa;valor de cotización actual;valor de cotización anterior`

Por ejemplo, algunas de las líneas del fichero podrían ser las siguientes:

```
20/01/2021 11:54:34;SANTANDER;4,54;4,49
14/05/1995 09:54;TELEFONICA;11,90;12,01
01/01/1997 08:03:21;SANTANDER;11,24;11,49
19/06/2022 11:54:22;APPLE;111,25;114,89
23/04/2003 16:32:11;ALPHABET;34,49;36,44;
21/12/2020 10:10:56;TELEFONICA;14,31;14;29
26/02/1995 14:09:40;MICROSOFT;132,29;133,95
04/05/1999 11:05:34;WALLMART;34,98;35,05
```

Tomando una media de 35.000 empresas cotizadas en el mundo, es decir, 35.000 cotizaciones por minuto serían 25.200.000 cotizaciones al día, y un total de **275.940.000.000 líneas** en el fichero por los 30 años a 365 días por cada año, que son unos **25 terabytes de datos en un único fichero**.

Nuestro objetivo es averiguar cuántas veces ha tenido cada empresa un incremento en su cotización, es decir, si una cotización es superior a su valor anterior, sumaremos uno, y si la cotización es inferior, no lo sumaremos. Es decir, el resultado sería una lista de la siguiente forma:

```
SANTANDER 3888981
TELEFONICA 3331923
```

Intentar este cálculo leyendo el fichero de forma secuencial y teniendo un contador para cada empresa sería un proceso que llevaría días de procesamiento, así que vamos a utilizar MapReduce para realizar este proceso.

Como se describió anteriormente, el primer paso es crear la aplicación, por ejemplo, utilizando lenguaje Java, y enviar el programa al clúster Hadoop utilizando el API de MapReduce para enviar trabajos.

Una vez arrancada la aplicación, en primer lugar decidirá cómo partir el fichero de entrada en fragmentos para que los datos puedan ser procesados en paralelo. El componente que realiza esta división de los ficheros de entrada se denomina **InputFormat**.

Por cada fragmento del fichero de entrada, se crea una tarea **map** que ejecutará la función **map** desarrollada en diferentes nodos y en paralelo, es decir, cada fragmento será procesado en paralelo por diferentes nodos.

La función **map** toma cada línea, que es separada por el *InputFormat*, la lee, y emite un resultado parcial, que será `[Nombre de la empresa, 1]`, en los casos en los que vea que el valor actual es mayor que el valor anterior. Esta función se ejecutará tantas veces como líneas tenga el fragmento de fichero asignado, y en tantos nodos como fragmentos se haya dividido el fichero.

Es decir, para cada nodo, tendremos, por ejemplo, este resultado:

```
SANTANDER, 1
WALLMART, 1
TELEFONICA, 1
APPLE, 1
ALPHABET, 1
...
```

Y tendremos tantos resultados de éstos como fragmentos del fichero haya, y en tantos nodos/servidores como se haya ejecutado la función.

A continuación se ejecutan las fases de **shuffle** y **sort** de forma automática y transparente para el desarrollador, donde se toman los resultados parciales, se ordenan por una **clave**, que en este caso será el nombre de la empresa, se **combinan** y se **ordenan**, juntando todos los valores de cada empresa, es decir, teniendo la lista de valores con el siguiente formato:

```
SANTANDER, 1
SANTANDER, 2
SANTANDER, 1
...
TELEFONICA, 1
TELEFONICA, 1
...
APPLE, 1
...
ALPHABET, 1
```

```
ALPHABET, 1
ALPHABET, 1
...
```

Por último, se divide la lista ordenada en diferentes particiones, siendo cada partición un conjunto de datos con la misma clave, y se llaman a la función **reduce** desarrollada por el usuario, que tomará los diferentes valores emitidos por la fase **map**, pero ya ordenados y unidos, e irá haciendo la suma de cada empresa, dando como resultado pares `[Nombre de la empresa, número de veces que se ha encontrado una cotización incrementada]`.

MapReduce, por último tomará todos los resultados de las funciones reduce y las unirá, formando el resultado final, que será la lista total de empresas con el número de veces en las que la cotización sube.

El ejemplo puede parecer sencillo, pero permite entender cómo funciona MapReduce para dividir un procesamiento en diferentes bloques de ejecución que se ejecutan en paralelo.

En la siguiente imagen puede verse de forma gráfica el ejemplo anterior:

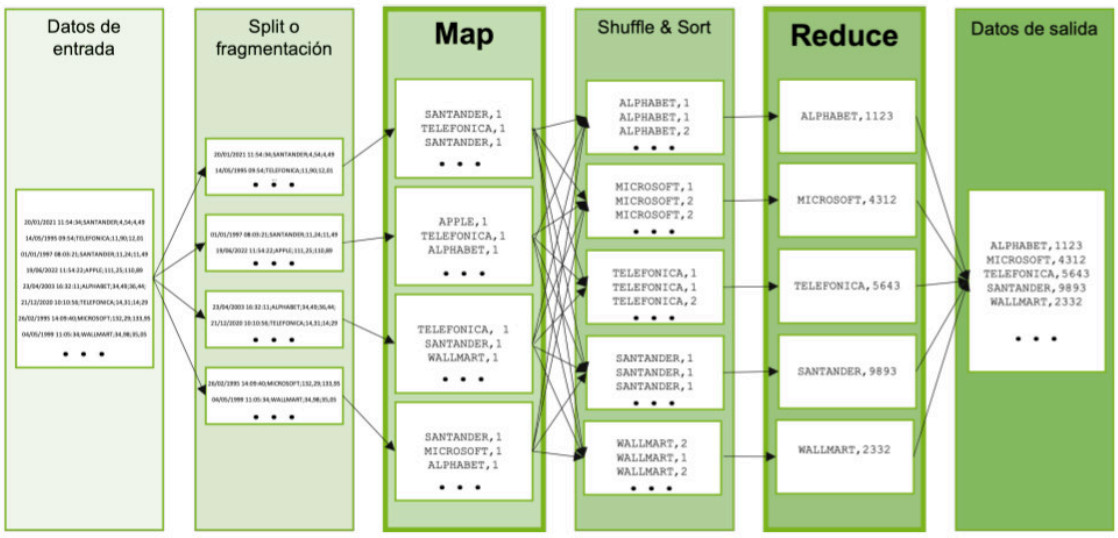

Figura4.8_MapReduce_Ejemplo2 Detallado. (Fuente: Ministerio de Educación)

Sobre los ejemplos

Quizás los ejemplos te parezcan muy sencillos y puedes pensar que MapReduce no resuelve problemas muy complejos. La realidad es que MapReduce permite resolver problemas de procesamiento de datos realmente complejos, pero requiere en primer lugar un estudio de problema y una división en problemas más sencillos que pueden resolverse en paralelo.

Además, cuando tienes que resolver un problema muy complejo, lo habitual es desarrollar un flujo de ejecución en el que se concatenan varios procesos MapReduce, donde el primer proceso hace una parte del trabajo, su resultado lo coge otro proceso MapReduce que realiza otra parte, etc. hasta llegar a tener todo el problema resuelto.

La esencia de MapReduce para resolver un problema es intentar descomponerlo en problemas más sencillos en los que cada problema se resuelve con una primera fase en la que se toman todos los datos de entrada uno a uno, se realiza alguna operación con ellos, y los resultados son combinados y ejecutados por otra fase de ejecución que realiza una operación con la que se devuelve el resultado.

5. Ejercicios

Vamos a ver 2 ejercicios para poner en práctica lo aprendido en MapReduce

5.1 Ejercicio 1

En este ejercicio, vamos a crear nuestro primer programa MapReduce. Asegúrate de que tienes Apache Hadoop instalado y funcionando. Si no, accede al recurso correspondiente.

1. Creamos un directorio en local donde vamos a desarrollar cada uno de los códigos fuente en Java

```
mkdir -p $HOME/bda/MapReduce/ejercicios
cd $HOME/bda/MapReduce/ejercicios
```

2. Descargamos el fichero fuente para el ejercicio en local. Contiene información relacionada con ventas, el nombre del producto, el precio, el modo de pago, la ciudad, el país del cliente, etc; alojado en este gist de github

```
wget
https://gist.githubusercontent.com/jaimerabasco/cb528c32b4c4092e6a0763d8b6bc25c0/
```

El objetivo será descubrir el número de productos vendidos en cada país.

3. Generamos las clases necesarias para nuestro objetivo en MapReduce

VentasMapper.java

```java
package VentasPais;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;

public class VentasMapper extends MapReduceBase implements
Mapper<LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);

    public void map(LongWritable key, Text value, OutputCollector<Text,
IntWritable> output, Reporter reporter) throws IOException {

        String valueString = value.toString();
        String[] SingleData_porPais = valueString.split(",");
        output.collect(new Text(SingleData_porPais[7]), one);
    }
}
```

VentasPaisDriver.java

```java
package VentasPais;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;

public class VentasPaisDriver {
    public static void main(String[] args) {
        JobClient my_client = new JobClient();
        // Creamos el objeto configuraciób para el trabajo
        JobConf job_conf = new JobConf(VentasPaisDriver.class);

        // Configuramos un nombre del trabajo
        job_conf.setJobName("VentaPorPais");

        // Especificamos el tipo de dato de clave y valor de salida
        job_conf.setOutputKeyClass(Text.class);
        job_conf.setOutputValueClass(IntWritable.class);

        // Especificamos nombres de la clase Mapper y Reducer
        job_conf.setMapperClass(VentasPais.VentasMapper.class);
        job_conf.setReducerClass(VentasPais.VentasPaisReducer.class);

        // Especificamos formato del tipo de dato de entrada y salida
        job_conf.setInputFormat(TextInputFormat.class);
        job_conf.setOutputFormat(TextOutputFormat.class);

        // Cambiamos la entrada y salida de directorios usando entrada
como argumentos:
```

```
        //arg[0] = nombre del directorio de entrada en HDFS,
        //arg[1] = nombre del directorio de salida que se va a crear para
almacenar el fichero de salida

        FileInputFormat.setInputPaths(job_conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(job_conf, new Path(args[1]));

        my_client.setConf(job_conf);
        try {
            // Ejecuta el job
            JobClient.runJob(job_conf);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

VentasPaisReducer.java

```
package VentasPais;

import java.io.IOException;
import java.util.*;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;

public class VentasPaisReducer extends MapReduceBase implements
Reducer<Text, IntWritable, Text, IntWritable> {

    public void reduce(Text t_key, Iterator<IntWritable> values,
OutputCollector<Text,IntWritable> output, Reporter reporter) throws
IOException {
        Text key = t_key;
        int frequencyPorPais = 0;
        while (values.hasNext()) {
            // reemplazamos el tipo de valor con el tipo real de nuestro
valor
            IntWritable value = (IntWritable) values.next();
            frequencyPorPais += value.get();

        }
        output.collect(key, new IntWritable(frequencyPorPais));
    }
}
```

4. Copia el fichero del "Ventas_Enero23.csv" a HDFS

```
hdfs dfs -mkdir -p /bda/mapreduce/ejercicios
hdfs dfs -copyFromLocal Ventas_Enero23.csv /bda/mapreduce/ejercicios
```

5. Generamos los .class para crear el paquete jar

```
javac -cp `hadoop classpath` -d . VentasMapper.java VentasPaisReducer.java
VentasPaisDriver.java
```

Se genera un directorio con los .class

6. Creamos un archivo llamado `Manifest.txt` con el siguiente contenido, donde indicamos el nombre de la clase main

```
Main-Class: VentasPais.VentasPaisDriver
```

Incluye también un salto de línea al final

7. Generamos el .jar

```
jar cfm VentasProductosPorPais.jar Manifest.txt VentasPais/*.class
```

8. Ejecutamos la aplicación creada indicando el input de entrada(*Ventas_Enero23.csv*) y la salida

```
hadoop jar VentasProductosPorPais.jar
/bda/mapreduce/ejercicios/Ventas_Enero23.csv
/bda/mapreduce/ejercicios/salida_ventas
```

Puedes usar también el comando `yarn jar`

```
2024-11-26 21:05:02,889 INFO impl.MetricsConfig: Loaded properties from
hadoop-metrics2.properties
2024-11-26 21:05:02,949 INFO impl.MetricsSystemImpl: Scheduled Metric
snapshot period at 10 second(s).
2024-11-26 21:05:02,949 INFO impl.MetricsSystemImpl: JobTracker metrics
system started
2024-11-26 21:05:02,961 WARN impl.MetricsSystemImpl: JobTracker metrics
system already initialized!
2024-11-26 21:05:03,041 WARN mapreduce.JobResourceUploader: Hadoop
command-line option parsing not performed. Implement the Tool interface and
execute your application with ToolRunner to remedy this.
2024-11-26 21:05:03,120 INFO mapred.FileInputFormat: Total input files
to process : 1
2024-11-26 21:05:03,176 INFO mapreduce.JobSubmitter: number of splits:1
2024-11-26 21:05:03,534 INFO mapreduce.JobSubmitter: Submitting tokens
for job: job_local1213752617_0001
2024-11-26 21:05:03,534 INFO mapreduce.JobSubmitter: Executing with
tokens: []
2024-11-26 21:05:03,634 INFO mapreduce.Job: The url to track the job:
http://localhost:8080/
2024-11-26 21:05:03,635 INFO mapreduce.Job: Running job:
job_local1213752617_0001
2024-11-26 21:05:03,635 INFO mapred.LocalJobRunner: OutputCommitter set
in config null
2024-11-26 21:05:03,636 INFO mapred.LocalJobRunner: OutputCommitter is
org.apache.hadoop.mapred.FileOutputCommitter
```

```
2024-11-26 21:05:03,641 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2024-11-26 21:05:03,641 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2024-11-26 21:05:03,716 INFO mapred.LocalJobRunner: Starting task: attempt_local1213752617_0001_m_000000_0
2024-11-26 21:05:03,716 INFO mapred.LocalJobRunner: Waiting for map tasks
2024-11-26 21:05:03,755 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2024-11-26 21:05:03,756 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2024-11-26 21:05:03,786 INFO mapred.Task:  Using ResourceCalculatorProcessTree : [ ]
2024-11-26 21:05:03,808 INFO mapred.MapTask: Processing split: hdfs://bda-iesgrancapitan:9000/bda/mapreduce/ejercicios/Ventas_Enero23.csv:0+123637
2024-11-26 21:05:03,850 INFO mapred.MapTask: numReduceTasks: 1
2024-11-26 21:05:03,865 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
2024-11-26 21:05:03,865 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
2024-11-26 21:05:03,865 INFO mapred.MapTask: soft limit at 83886080
2024-11-26 21:05:03,865 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
2024-11-26 21:05:03,865 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
2024-11-26 21:05:03,867 INFO mapred.MapTask: Map output collector class = org.apache.hadoop.mapred.MapTask$MapOutputBuffer
2024-11-26 21:05:03,981 INFO mapred.LocalJobRunner:
2024-11-26 21:05:03,981 INFO mapred.MapTask: Starting flush of map output
2024-11-26 21:05:03,981 INFO mapred.MapTask: Spilling map output
2024-11-26 21:05:03,981 INFO mapred.MapTask: bufstart = 0; bufend = 15743; bufvoid = 104857600
2024-11-26 21:05:03,981 INFO mapred.MapTask: kvstart = 26214396(104857584); kvend = 26210404(104841616); length = 3993/6553600
2024-11-26 21:05:03,994 INFO mapred.MapTask: Finished spill 0
2024-11-26 21:05:04,003 INFO mapred.Task: Task:attempt_local1213752617_0001_m_000000_0 is done. And is in the process of committing
2024-11-26 21:05:04,005 INFO mapred.LocalJobRunner: hdfs://bda-iesgrancapitan:9000/bda/mapreduce/ejercicios/Ventas_Enero23.csv:0+123637
2024-11-26 21:05:04,006 INFO mapred.Task: Task 'attempt_local1213752617_0001_m_000000_0' done.
2024-11-26 21:05:04,010 INFO mapred.Task: Final Counters for attempt_local1213752617_0001_m_000000_0: Counters: 23
      File System Counters
          FILE: Number of bytes read=3122
          FILE: Number of bytes written=737413
          FILE: Number of read operations=0
          FILE: Number of large read operations=0
          FILE: Number of write operations=0
          HDFS: Number of bytes read=123637
          HDFS: Number of bytes written=0
```

```
47          HDFS: Number of read operations=5
48          HDFS: Number of large read operations=0
49          HDFS: Number of write operations=1
50          HDFS: Number of bytes read erasure-coded=0
51      Map-Reduce Framework
52          Map input records=999
53          Map output records=999
54          Map output bytes=15743
55          Map output materialized bytes=17747
56          Input split bytes=126
57          Combine input records=0
58          Spilled Records=999
59          Failed Shuffles=0
60          Merged Map outputs=0
61          GC time elapsed (ms)=9
62          Total committed heap usage (bytes)=249036800
63      File Input Format Counters
64          Bytes Read=123637
65  2024-11-26 21:05:04,012 INFO mapred.LocalJobRunner: Finishing task:
attempt_local1213752617_0001_m_000000_0
66  2024-11-26 21:05:04,012 INFO mapred.LocalJobRunner: map task executor
complete.
67  2024-11-26 21:05:04,014 INFO mapred.LocalJobRunner: Starting task:
attempt_local1213752617_0001_r_000000_0
68  2024-11-26 21:05:04,018 INFO mapred.LocalJobRunner: Waiting for reduce
tasks
69  2024-11-26 21:05:04,019 INFO output.FileOutputCommitter: File Output
Committer Algorithm version is 2
70  2024-11-26 21:05:04,019 INFO output.FileOutputCommitter:
FileOutputCommitter skip cleanup _temporary folders under output
directory:false, ignore cleanup failures: false
71  2024-11-26 21:05:04,019 INFO mapred.Task:  Using
ResourceCalculatorProcessTree : [ ]
72  2024-11-26 21:05:04,047 INFO mapred.ReduceTask: Using
ShuffleConsumerPlugin:
org.apache.hadoop.mapreduce.task.reduce.Shuffle@3dc5c9a6
73  2024-11-26 21:05:04,048 WARN impl.MetricsSystemImpl: JobTracker metrics
system already initialized!
74  2024-11-26 21:05:04,059 INFO reduce.MergeManagerImpl: MergerManager:
memoryLimit=639683776, maxSingleShuffleLimit=159920944,
mergeThreshold=422191296, ioSortFactor=10, memToMemMergeOutputsThreshold=10
75  2024-11-26 21:05:04,061 INFO reduce.EventFetcher:
attempt_local1213752617_0001_r_000000_0 Thread started: EventFetcher for
fetching Map Completion Events
76  2024-11-26 21:05:04,084 INFO reduce.LocalFetcher: localfetcher#1 about
to shuffle output of map attempt_local1213752617_0001_m_000000_0 decomp: 17743
len: 17747 to MEMORY
77  2024-11-26 21:05:04,086 INFO reduce.InMemoryMapOutput: Read 17743 bytes
from map-output for attempt_local1213752617_0001_m_000000_0
78  2024-11-26 21:05:04,087 INFO reduce.MergeManagerImpl: closeInMemoryFile
-> map-output of size: 17743, inMemoryMapOutputs.size() -> 1, commitMemory ->
0, usedMemory ->17743
79  2024-11-26 21:05:04,088 INFO reduce.EventFetcher: EventFetcher is
interrupted.. Returning
80  2024-11-26 21:05:04,088 INFO mapred.LocalJobRunner: 1 / 1 copied.
81  2024-11-26 21:05:04,088 INFO reduce.MergeManagerImpl: finalMerge called
with 1 in-memory map-outputs and 0 on-disk map-outputs
```

```
 82  2024-11-26 21:05:04,093 INFO mapred.Merger: Merging 1 sorted segments
 83  2024-11-26 21:05:04,093 INFO mapred.Merger: Down to the last merge-pass,
with 1 segments left of total size: 17731 bytes
 84  2024-11-26 21:05:04,097 INFO reduce.MergeManagerImpl: Merged 1 segments,
17743 bytes to disk to satisfy reduce memory limit
 85  2024-11-26 21:05:04,097 INFO reduce.MergeManagerImpl: Merging 1 files,
17747 bytes from disk
 86  2024-11-26 21:05:04,098 INFO reduce.MergeManagerImpl: Merging 0
segments, 0 bytes from memory into reduce
 87  2024-11-26 21:05:04,098 INFO mapred.Merger: Merging 1 sorted segments
 88  2024-11-26 21:05:04,099 INFO mapred.Merger: Down to the last merge-pass,
with 1 segments left of total size: 17731 bytes
 89  2024-11-26 21:05:04,099 INFO mapred.LocalJobRunner: 1 / 1 copied.
 90  2024-11-26 21:05:04,433 INFO mapred.Task:
Task:attempt_local1213752617_0001_r_000000_0 is done. And is in the process of
committing
 91  2024-11-26 21:05:04,441 INFO mapred.LocalJobRunner: 1 / 1 copied.
 92  2024-11-26 21:05:04,442 INFO mapred.Task: Task
attempt_local1213752617_0001_r_000000_0 is allowed to commit now
 93  2024-11-26 21:05:04,542 INFO output.FileOutputCommitter: Saved output of
task 'attempt_local1213752617_0001_r_000000_0' to hdfs://bda-
iesgrancapitan:9000/bda/mapreduce/ejercicios/salida_ventas
 94  2024-11-26 21:05:04,546 INFO mapred.LocalJobRunner: reduce > reduce
 95  2024-11-26 21:05:04,547 INFO mapred.Task: Task
'attempt_local1213752617_0001_r_000000_0' done.
 96  2024-11-26 21:05:04,549 INFO mapred.Task: Final Counters for
attempt_local1213752617_0001_r_000000_0: Counters: 30
 97      File System Counters
 98          FILE: Number of bytes read=38648
 99          FILE: Number of bytes written=755160
100          FILE: Number of read operations=0
101          FILE: Number of large read operations=0
102          FILE: Number of write operations=0
103          HDFS: Number of bytes read=123637
104          HDFS: Number of bytes written=661
105          HDFS: Number of read operations=10
106          HDFS: Number of large read operations=0
107          HDFS: Number of write operations=3
108          HDFS: Number of bytes read erasure-coded=0
109      Map-Reduce Framework
110          Combine input records=0
111          Combine output records=0
112          Reduce input groups=58
113          Reduce shuffle bytes=17747
114          Reduce input records=999
115          Reduce output records=58
116          Spilled Records=999
117          Shuffled Maps =1
118          Failed Shuffles=0
119          Merged Map outputs=1
120          GC time elapsed (ms)=0
121          Total committed heap usage (bytes)=249036800
122      Shuffle Errors
123          BAD_ID=0
124          CONNECTION=0
125          IO_ERROR=0
126          WRONG_LENGTH=0
```

```
            WRONG_MAP=0
            WRONG_REDUCE=0
        File Output Format Counters
            Bytes Written=661
    2024-11-26 21:05:04,552 INFO mapred.LocalJobRunner: Finishing task:
attempt_local1213752617_0001_r_000000_0
    2024-11-26 21:05:04,552 INFO mapred.LocalJobRunner: reduce task executor
complete.
    2024-11-26 21:05:04,640 INFO mapreduce.Job: Job job_local1213752617_0001
running in uber mode : false
    2024-11-26 21:05:04,642 INFO mapreduce.Job:  map 100% reduce 100%
    2024-11-26 21:05:05,646 INFO mapreduce.Job: Job job_local1213752617_0001
completed successfully
    2024-11-26 21:05:05,674 INFO mapreduce.Job: Counters: 36
        File System Counters
            FILE: Number of bytes read=41770
            FILE: Number of bytes written=1492573
            FILE: Number of read operations=0
            FILE: Number of large read operations=0
            FILE: Number of write operations=0
            HDFS: Number of bytes read=247274
            HDFS: Number of bytes written=661
            HDFS: Number of read operations=15
            HDFS: Number of large read operations=0
            HDFS: Number of write operations=4
            HDFS: Number of bytes read erasure-coded=0
        Map-Reduce Framework
            Map input records=999
            Map output records=999
            Map output bytes=15743
            Map output materialized bytes=17747
            Input split bytes=126
            Combine input records=0
            Combine output records=0
            Reduce input groups=58
            Reduce shuffle bytes=17747
            Reduce input records=999
            Reduce output records=58
            Spilled Records=1998
            Shuffled Maps =1
            Failed Shuffles=0
            Merged Map outputs=1
            GC time elapsed (ms)=9
            Total committed heap usage (bytes)=498073600
        Shuffle Errors
            BAD_ID=0
            CONNECTION=0
            IO_ERROR=0
            WRONG_LENGTH=0
            WRONG_MAP=0
            WRONG_REDUCE=0
        File Input Format Counters
            Bytes Read=123637
        File Output Format Counters
            Bytes Written=661
```

9. Vemos el fichero resultante y comprobamos si hemos obtenido los datos que teníamos como objetivo

```
hdfs dfs -cat /bda/mapreduce/ejercicios/salida_ventas/part-00000
```

```
Argentina   1
Australia   38
Austria 7
Bahrain 1
Belgium 8
Bermuda 1
Brazil  5
Bulgaria    1
CO  1
Canada  76
Cayman Isls 1
China   1
Costa Rica  1
Country 1
Czech Republic  3
Denmark 15
Dominican Republic  1
Finland 2
France  27
Germany 25
Greece  1
Guatemala   1
Hong Kong   1
Hungary 3
Iceland 1
India   2
Ireland 49
Israel  1
Italy   15
Japan   2
Jersey  1
Kuwait  1
Latvia  1
Luxembourg  1
Malaysia    1
Malta   2
Mauritius   1
Moldova 1
Monaco  2
Netherlands 22
New Zealand 6
Norway  16
Philippines 2
Poland  2
Romania 1
Russia  1
South Africa    5
South Korea 1
Spain   12
Sweden  13
```

```
Switzerland 36
Thailand    2
The Bahamas 2
Turkey  6
Ukraine 1
United Arab Emirates    6
United Kingdom  100
United States   462
```

5.2 Ejercicio 2

Vamos a realizar un ejercicio de ejemplo de uso de MapReduce. En este caso vamos a usar una de las aplicaciones que ya vienen dentro de MapReduce, que es el contador de palabras

1. Primero descargamos un fichero en local de este gist alojado en github. En este caso vamos a contar las palabras de "**El quijote**".

```
wget
https://gist.githubusercontent.com/jaimerabasco/cb528c32b4c4092e6a0763d8b6bc25c0/
```

2. Crea en HDFS la carpeta donde vas a alojar el fichero

```
// No es necesario si ya los has creado en el ejercicio anterior
hdfs dfs -mkdir /bda/mapreduce
hdfs dfs -mkdir /bda/mapreduce/ejercicios
```

3. Copia el fichero del "El quijote" a HDFS

```
hdfs dfs -copyFromLocal El_Quijote.txt /bda/mapreduce/ejercicios
```

4. Listamos la lista de ejemplos que nos dispone MapReduce

```
hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-
3.4.1.jar
```

5. Vemos muchos de ellos. Para este ejercicio nos interesa ***wordcount***

```
An example program must be given as the first argument.
Valid program names are:
  aggregatewordcount: An Aggregate based map/reduce program that counts
the words in the input files.
  aggregatewordhist: An Aggregate based map/reduce program that computes
the histogram of the words in the input files.
  bbp: A map/reduce program that uses Bailey-Borwein-Plouffe to compute
exact digits of Pi.
  dbcount: An example job that count the pageview counts from a database.
  distbbp: A map/reduce program that uses a BBP-type formula to compute
exact bits of Pi.
```

```
    grep: A map/reduce program that counts the matches of a regex in the
input.
    join: A job that effects a join over sorted, equally partitioned
datasets
    multifilewc: A job that counts words from several files.
    pentomino: A map/reduce tile laying program to find solutions to
pentomino problems.
    pi: A map/reduce program that estimates Pi using a quasi-Monte Carlo
method.
    randomtextwriter: A map/reduce program that writes 10GB of random
textual data per node.
    randomwriter: A map/reduce program that writes 10GB of random data per
node.
    secondarysort: An example defining a secondary sort to the reduce.
    sort: A map/reduce program that sorts the data written by the random
writer.
    sudoku: A sudoku solver.
    teragen: Generate data for the terasort
    terasort: Run the terasort
    teravalidate: Checking results of terasort
    wordcount: A map/reduce program that counts the words in the input
files.
    wordmean: A map/reduce program that counts the average length of the
words in the input files.
    wordmedian: A map/reduce program that counts the median length of the
words in the input files.
    wordstandarddeviation: A map/reduce program that counts the standard
deviation of the length of the words in the input files.
```

7. Vemos los parámetros que necesitamos para wordcount

```
hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-
3.4.1.jar wordcount
Usage: wordcount <in> [<in>...] <out>
```

8. Ejecuta el ejemplo de contar palabras sobre el "El quijote". Hay que indicar el fichero de entrada y de salida

```
hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-
3.4.1.jar wordcount /bda/mapreduce/ejercicios/El_Quijote.txt
/bda/mapreduce/ejercicios/salida_quijote
```

```
2024-11-26 21:09:21,134 INFO impl.MetricsConfig: Loaded properties from
hadoop-metrics2.properties
2024-11-26 21:09:21,195 INFO impl.MetricsSystemImpl: Scheduled Metric
snapshot period at 10 second(s).
2024-11-26 21:09:21,195 INFO impl.MetricsSystemImpl: JobTracker metrics
system started
2024-11-26 21:09:21,341 INFO input.FileInputFormat: Total input files to
process : 1
2024-11-26 21:09:21,381 INFO mapreduce.JobSubmitter: number of splits:1
2024-11-26 21:09:21,458 INFO mapreduce.JobSubmitter: Submitting tokens
for job: job_local1608140050_0001
```

```
2024-11-26 21:09:21,458 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-11-26 21:09:21,543 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2024-11-26 21:09:21,543 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2024-11-26 21:09:21,543 INFO mapreduce.Job: Running job: job_local608140050_0001
2024-11-26 21:09:21,561 INFO output.PathOutputCommitterFactory: No output committer factory defined, defaulting to FileOutputCommitterFactory
2024-11-26 21:09:21,562 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2024-11-26 21:09:21,562 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2024-11-26 21:09:21,563 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
2024-11-26 21:09:21,658 INFO mapred.LocalJobRunner: Starting task: attempt_local608140050_0001_m_000000_0
2024-11-26 21:09:21,658 INFO mapred.LocalJobRunner: Waiting for map tasks
2024-11-26 21:09:21,674 INFO output.PathOutputCommitterFactory: No output committer factory defined, defaulting to FileOutputCommitterFactory
2024-11-26 21:09:21,674 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2024-11-26 21:09:21,674 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2024-11-26 21:09:21,684 INFO mapred.Task:  Using ResourceCalculatorProcessTree : [ ]
2024-11-26 21:09:21,687 INFO mapred.MapTask: Processing split: hdfs://bda-iesgrancapitan:9000/bda/mapreduce/ejercicios/El_Quijote.txt:0+2161175
2024-11-26 21:09:21,711 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
2024-11-26 21:09:21,712 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
2024-11-26 21:09:21,712 INFO mapred.MapTask: soft limit at 83886080
2024-11-26 21:09:21,712 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
2024-11-26 21:09:21,712 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
2024-11-26 21:09:21,714 INFO mapred.MapTask: Map output collector class = org.apache.hadoop.mapred.MapTask$MapOutputBuffer
2024-11-26 21:09:22,062 INFO mapred.LocalJobRunner:
2024-11-26 21:09:22,066 INFO mapred.MapTask: Starting flush of map output
2024-11-26 21:09:22,066 INFO mapred.MapTask: Spilling map output
2024-11-26 21:09:22,066 INFO mapred.MapTask: bufstart = 0; bufend = 3688770; bufvoid = 104857600
2024-11-26 21:09:22,066 INFO mapred.MapTask: kvstart = 26214396(104857584); kvend = 24677300(98709200); length = 1537097/6553600
2024-11-26 21:09:22,390 INFO mapred.MapTask: Finished spill 0
2024-11-26 21:09:22,396 INFO mapred.Task: Task:attempt_local608140050_0001_m_000000_0 is done. And is in the process of committing
2024-11-26 21:09:22,399 INFO mapred.LocalJobRunner: map
```

```
 36   2024-11-26 21:09:22,399 INFO mapred.Task: Task
'attempt_local608140050_0001_m_000000_0' done.
 37   2024-11-26 21:09:22,404 INFO mapred.Task: Final Counters for
attempt_local608140050_0001_m_000000_0: Counters: 24
 38       File System Counters
 39           FILE: Number of bytes read=281814
 40           FILE: Number of bytes written=1602813
 41           FILE: Number of read operations=0
 42           FILE: Number of large read operations=0
 43           FILE: Number of write operations=0
 44           HDFS: Number of bytes read=2161175
 45           HDFS: Number of bytes written=0
 46           HDFS: Number of read operations=5
 47           HDFS: Number of large read operations=0
 48           HDFS: Number of write operations=1
 49           HDFS: Number of bytes read erasure-coded=0
 50       Map-Reduce Framework
 51           Map input records=37863
 52           Map output records=384275
 53           Map output bytes=3688770
 54           Map output materialized bytes=605631
 55           Input split bytes=135
 56           Combine input records=384275
 57           Combine output records=40067
 58           Spilled Records=40067
 59           Failed Shuffles=0
 60           Merged Map outputs=0
 61           GC time elapsed (ms)=82
 62           Total committed heap usage (bytes)=499122176
 63       File Input Format Counters
 64           Bytes Read=2161175
 65   2024-11-26 21:09:22,404 INFO mapred.LocalJobRunner: Finishing task:
attempt_local608140050_0001_m_000000_0
 66   2024-11-26 21:09:22,404 INFO mapred.LocalJobRunner: map task executor
complete.
 67   2024-11-26 21:09:22,406 INFO mapred.LocalJobRunner: Waiting for reduce
tasks
 68   2024-11-26 21:09:22,406 INFO mapred.LocalJobRunner: Starting task:
attempt_local608140050_0001_r_000000_0
 69   2024-11-26 21:09:22,411 INFO output.PathOutputCommitterFactory: No
output committer factory defined, defaulting to FileOutputCommitterFactory
 70   2024-11-26 21:09:22,411 INFO output.FileOutputCommitter: File Output
Committer Algorithm version is 2
 71   2024-11-26 21:09:22,411 INFO output.FileOutputCommitter:
FileOutputCommitter skip cleanup _temporary folders under output
directory:false, ignore cleanup failures: false
 72   2024-11-26 21:09:22,411 INFO mapred.Task:  Using
ResourceCalculatorProcessTree : [ ]
 73   2024-11-26 21:09:22,413 INFO mapred.ReduceTask: Using
ShuffleConsumerPlugin: org.apache.hadoop.mapreduce.task.reduce.Shuffle@b429ab7
 74   2024-11-26 21:09:22,414 WARN impl.MetricsSystemImpl: JobTracker metrics
system already initialized!
 75   2024-11-26 21:09:22,425 INFO reduce.MergeManagerImpl: MergerManager:
memoryLimit=639683776, maxSingleShuffleLimit=159920944,
mergeThreshold=422191296, ioSortFactor=10, memToMemMergeOutputsThreshold=10
 76   2024-11-26 21:09:22,427 INFO reduce.EventFetcher:
attempt_local608140050_0001_r_000000_0 Thread started: EventFetcher for
```

```
fetching Map Completion Events
2024-11-26 21:09:22,442 INFO reduce.LocalFetcher: localfetcher#1 about to shuffle output of map attempt_local608140050_0001_m_000000_0 decomp: 605627 len: 605631 to MEMORY
2024-11-26 21:09:22,445 INFO reduce.InMemoryMapOutput: Read 605627 bytes from map-output for attempt_local608140050_0001_m_000000_0
2024-11-26 21:09:22,446 INFO reduce.MergeManagerImpl: closeInMemoryFile -> map-output of size: 605627, inMemoryMapOutputs.size() -> 1, commitMemory -> 0, usedMemory ->605627
2024-11-26 21:09:22,446 INFO reduce.EventFetcher: EventFetcher is interrupted.. Returning
2024-11-26 21:09:22,447 INFO mapred.LocalJobRunner: 1 / 1 copied.
2024-11-26 21:09:22,447 INFO reduce.MergeManagerImpl: finalMerge called with 1 in-memory map-outputs and 0 on-disk map-outputs
2024-11-26 21:09:22,452 INFO mapred.Merger: Merging 1 sorted segments
2024-11-26 21:09:22,453 INFO mapred.Merger: Down to the last merge-pass, with 1 segments left of total size: 605620 bytes
2024-11-26 21:09:22,483 INFO reduce.MergeManagerImpl: Merged 1 segments, 605627 bytes to disk to satisfy reduce memory limit
2024-11-26 21:09:22,484 INFO reduce.MergeManagerImpl: Merging 1 files, 605631 bytes from disk
2024-11-26 21:09:22,484 INFO reduce.MergeManagerImpl: Merging 0 segments, 0 bytes from memory into reduce
2024-11-26 21:09:22,484 INFO mapred.Merger: Merging 1 sorted segments
2024-11-26 21:09:22,484 INFO mapred.Merger: Down to the last merge-pass, with 1 segments left of total size: 605620 bytes
2024-11-26 21:09:22,485 INFO mapred.LocalJobRunner: 1 / 1 copied.
2024-11-26 21:09:22,546 INFO mapreduce.Job: Job job_local608140050_0001 running in uber mode : false
2024-11-26 21:09:22,547 INFO mapreduce.Job:  map 100% reduce 0%
2024-11-26 21:09:22,557 INFO Configuration.deprecation: mapred.skip.on is deprecated. Instead, use mapreduce.job.skiprecords
2024-11-26 21:09:22,793 INFO mapred.Task: Task:attempt_local608140050_0001_r_000000_0 is done. And is in the process of committing
2024-11-26 21:09:22,795 INFO mapred.LocalJobRunner: 1 / 1 copied.
2024-11-26 21:09:22,795 INFO mapred.Task: Task attempt_local608140050_0001_r_000000_0 is allowed to commit now
2024-11-26 21:09:22,859 INFO output.FileOutputCommitter: Saved output of task 'attempt_local608140050_0001_r_000000_0' to hdfs://bda-iesgrancapitan:9000/bda/mapreduce/ejercicios/salida_quijote
2024-11-26 21:09:22,860 INFO mapred.LocalJobRunner: reduce > reduce
2024-11-26 21:09:22,860 INFO mapred.Task: Task 'attempt_local608140050_0001_r_000000_0' done.
2024-11-26 21:09:22,860 INFO mapred.Task: Final Counters for attempt_local608140050_0001_r_000000_0: Counters: 30
      File System Counters
          FILE: Number of bytes read=1493108
          FILE: Number of bytes written=2208444
          FILE: Number of read operations=0
          FILE: Number of large read operations=0
          FILE: Number of write operations=0
          HDFS: Number of bytes read=2161175
          HDFS: Number of bytes written=448985
          HDFS: Number of read operations=10
          HDFS: Number of large read operations=0
          HDFS: Number of write operations=3
```

```
112             HDFS: Number of bytes read erasure-coded=0
113         Map-Reduce Framework
114             Combine input records=0
115             Combine output records=0
116             Reduce input groups=40067
117             Reduce shuffle bytes=605631
118             Reduce input records=40067
119             Reduce output records=40067
120             Spilled Records=40067
121             Shuffled Maps =1
122             Failed Shuffles=0
123             Merged Map outputs=1
124             GC time elapsed (ms)=0
125             Total committed heap usage (bytes)=499122176
126         Shuffle Errors
127             BAD_ID=0
128             CONNECTION=0
129             IO_ERROR=0
130             WRONG_LENGTH=0
131             WRONG_MAP=0
132             WRONG_REDUCE=0
133         File Output Format Counters
134             Bytes Written=448985
135     2024-11-26 21:09:22,860 INFO mapred.LocalJobRunner: Finishing task:
attempt_local608140050_0001_r_000000_0
136     2024-11-26 21:09:22,860 INFO mapred.LocalJobRunner: reduce task executor
complete.
137     2024-11-26 21:09:23,549 INFO mapreduce.Job:  map 100% reduce 100%
138     2024-11-26 21:09:23,549 INFO mapreduce.Job: Job job_local608140050_0001
completed successfully
139     2024-11-26 21:09:23,573 INFO mapreduce.Job: Counters: 36
140         File System Counters
141             FILE: Number of bytes read=1774922
142             FILE: Number of bytes written=3811257
143             FILE: Number of read operations=0
144             FILE: Number of large read operations=0
145             FILE: Number of write operations=0
146             HDFS: Number of bytes read=4322350
147             HDFS: Number of bytes written=448985
148             HDFS: Number of read operations=15
149             HDFS: Number of large read operations=0
150             HDFS: Number of write operations=4
151             HDFS: Number of bytes read erasure-coded=0
152         Map-Reduce Framework
153             Map input records=37863
154             Map output records=384275
155             Map output bytes=3688770
156             Map output materialized bytes=605631
157             Input split bytes=135
158             Combine input records=384275
159             Combine output records=40067
160             Reduce input groups=40067
161             Reduce shuffle bytes=605631
162             Reduce input records=40067
163             Reduce output records=40067
164             Spilled Records=80134
165             Shuffled Maps =1
```

```
        Failed Shuffles=0
        Merged Map outputs=1
        GC time elapsed (ms)=82
        Total committed heap usage (bytes)=998244352
    Shuffle Errors
        BAD_ID=0
        CONNECTION=0
        IO_ERROR=0
        WRONG_LENGTH=0
        WRONG_MAP=0
        WRONG_REDUCE=0
    File Input Format Counters
        Bytes Read=2161175
    File Output Format Counters
        Bytes Written=448985
```

9. Leemos el fichero de salida. Aquí están listados todas las palabras de El Quijote y cuantas veces aparece cada palabra

```
hdfs dfs -cat /bda/mapreduce/ejercicios/salida_quijote/part-r-00000
```