MapReduce

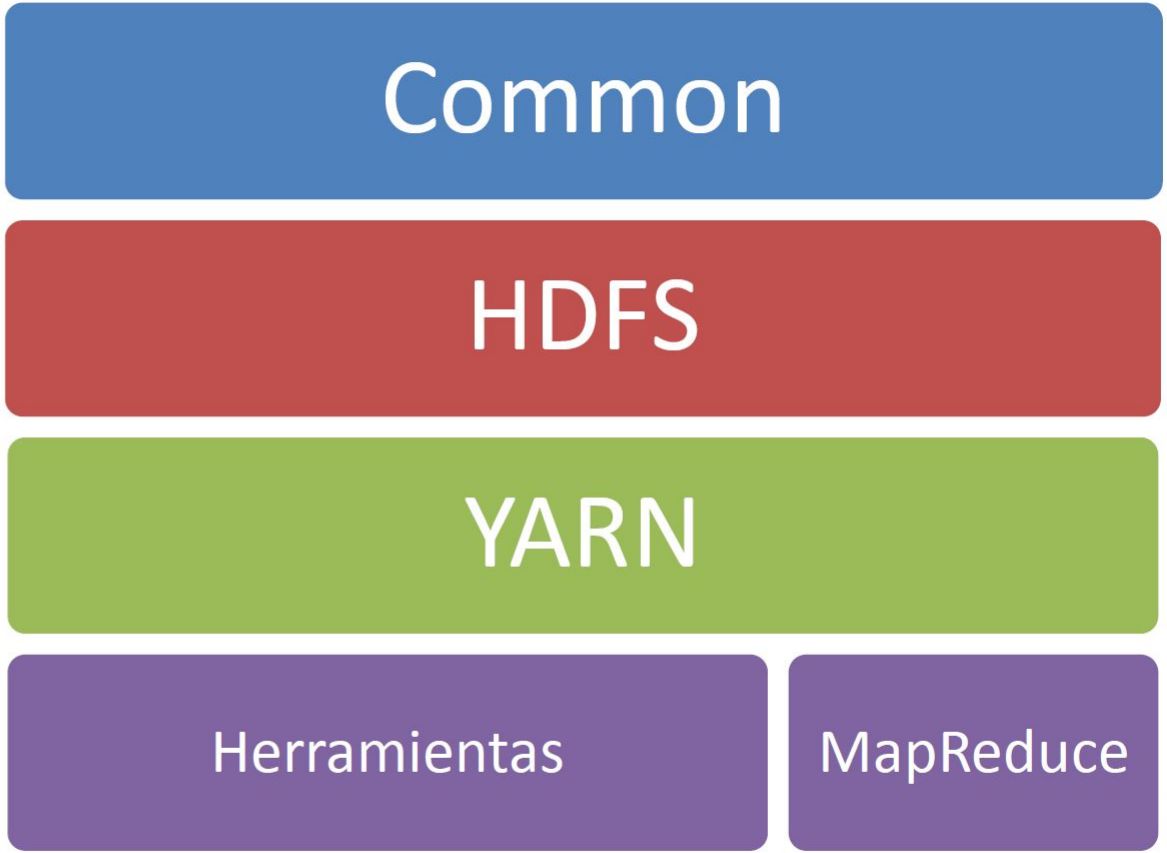
Sistemas de Big Data

Sistemas de
Big_Data

Curso Especialización
Inteligencia_Artificial y
Big_Data

Hadoop

Componentes



MapReduce

MapReduce

- Es un modelo genérico de programación distribuida y en paralelo.
- La herramienta se encarga de distribuir el programa en los nodos
- Es la tecnología por excelencia dentro del procesamiento en lotes.
- Está enmarcada dentro del entorno Hadoop, y aprovecha el sistema de ficheros distribuido para ser altamente escalable.
- Si bien su utilización ha ido decayendo en los últimos años en favor de otras opciones, como Apache Spark, es conveniente conocer sus características y ver sus posibles aplicaciones.

FUNCIONAMIENTO GENERAL

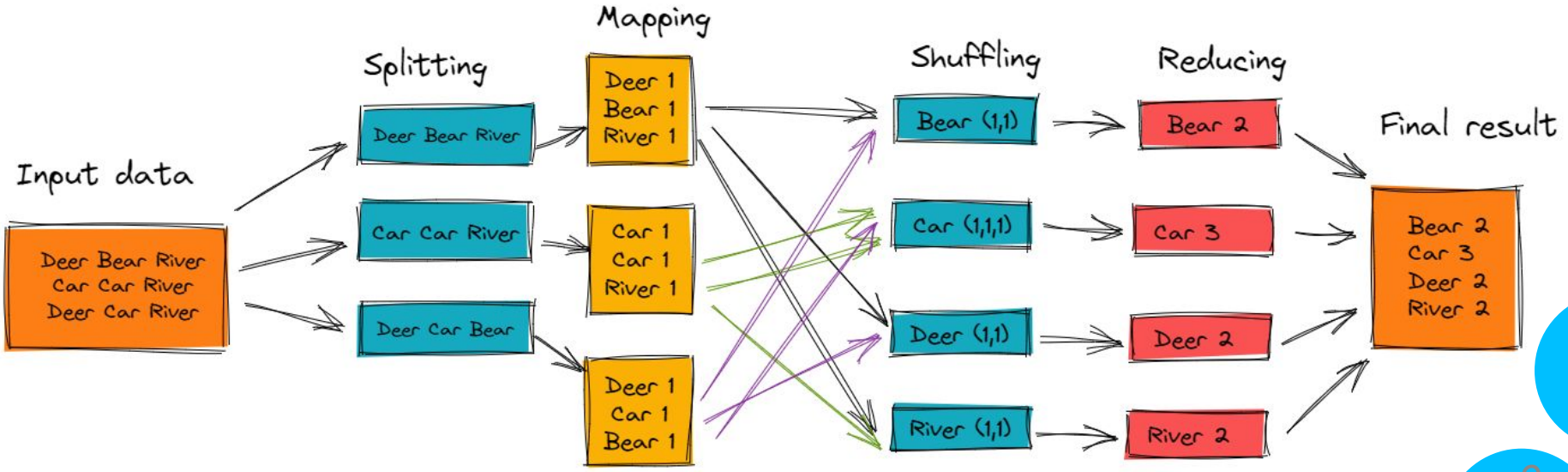
- Una tarea MapReduce (*job*) está enfocada al procesamiento por lotes, y se compone de unos datos de entrada que queremos procesar, una primera función de transformación (*map*), y otra segunda de agregación (*reduce*) y unos parámetros de configuración de la tarea.
- Entre etapa y etapa se transmite información en formato de pares Clave (k), Valor (v).
- Además de las etapas principales, tenemos dos etapas más, que se ejecutarán de forma automática, y siempre de la misma forma.

FUNCIONAMIENTO GENERAL

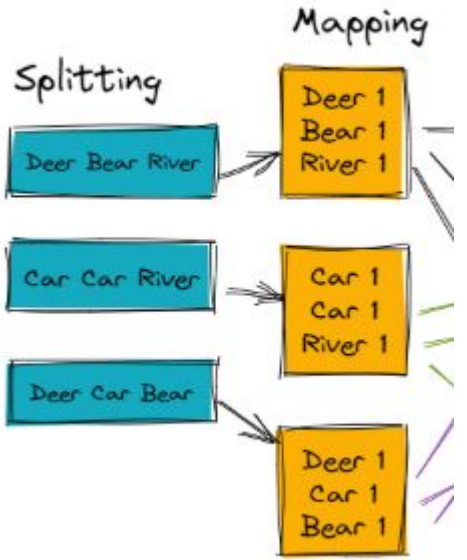
- Como MapReduce trabaja sobre Hadoop, los datos van a estar repartidos en el cluster, por lo que MapReduce lanza de forma automática varias tareas Map, una para cada nodo sobre la que va a ejecutarse.
- En cada ordenador o nodo se analizan principalmente los datos que tiene almacenados ese mismo nodo. Es lo que se llama *Data Locality*.

MapReduce

- Ejemplo del proceso
  - Podemos ver que hay varios pasos. Primero se reparten los datos entre los distintos mappers, la etapa map, la etapa shuffling y la etapa Reducing



Etapa MAP



- MapReduce crea un número de tareas MAP, que depende del tamaño del bloque en HDFS y del número de datos de entrada que haya. Las tareas se distribuyen en el cluster.
- Cada trabajo Map recibe como entrada los datos que queremos analizar, y que se encontrarán almacenados en HDFS.
- En la etapa del MAP se filtran y preparan los datos de entrada para ser procesados en la etapa REDUCE.
- Si quisiéramos contar el número de palabras de un texto, la entrada del map será las líneas del texto, y lo que hará será devolver a la siguiente etapa las palabras separadas de forma individual, una palabra por cada línea con el 1, que indica que esa palabra aparece 1 vez (si una palabra aparece más de una vez, aparecerá más veces con un 1 cada vez)

Etapa MAP

- La salida del map es un conjunto de pares con una clave y un valor. `map(k,v)`
- Recibe los datos de entrada y devuelve como salida el par de clave valor que necesita la etapa REDUCE para calcular los datos.
- Esta salida intermedia puede almacenarse en memoria o en el sistema de fichero local de la máquina en el caso de que la memoria se llene, pero nunca en HDFS.
- En el caso del ejemplo de contar las palabras de un texto, el par clave valor será:
  a. La clave será una palabra individual
  b. El valor será el número 1, ya que esa clave es 1 palabra.
- En función del lenguaje de programación elegida, la entrada podrá ser una línea o bien todos los datos que tiene asignada esa tarea map.

Etapas SHUFFLE y SORT

- Son dos etapas automáticas que se ejecutan entre la tarea Map y la tarea Reduce.
- **SHUFFLE**:
  a. En el mismo nodo en el que se ejecuta la tarea Map, se decide a qué tarea Reduce se va a enviar cada par **<clave , valor>**.
  b. Todos los pares clave valor que compartan la misma clave se reparten a la misma tarea Reduce.
  c. Para ello se realiza un hash de la clave y se le hace el módulo del número de tareas Reduce que se van a utilizar.

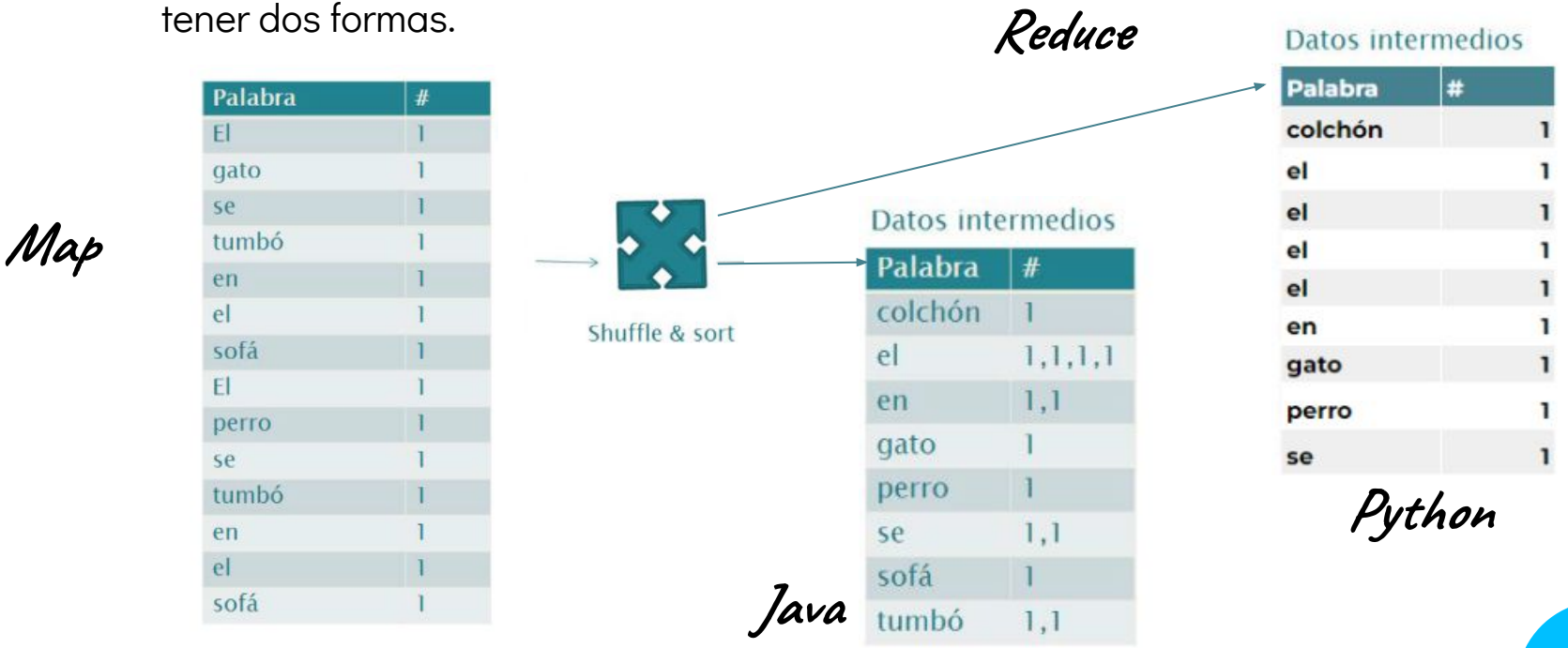
Nº de Nodo al que se envía la clave = **hash** *(clave)* **mod** *(Número Tareas Reduce)*

Etapas SHUFFLE y SORT

- SORT:
  a. Cuando los pares **<clave , valor>** llegan al nodo donde se va a ejecutar el Reduce, y justo antes de que se ejecute el REDUCE, se realiza una ordenación y puede que una agrupación de los pares por su clave.
  b. Es decir, todos los pares van a estar ordenados por su clave, por lo que los pares con la misma clave van a llegar al reduce juntos..
  c. En algunas implementaciones de MapReduce, los valores se van a agrupar por la clave, apareciendo todos los valores en una lista asociada a esa clave.

Etapa SHUFFLE & SORT

- En función del lenguaje de programación que utilicemos, la salida de esta fase podrá tener dos formas.



Palabra	#
El	1
gato	1
se	1
tumbó	1
en	1
el	1
sofá	1
El	1
perro	1
se	1
tumbó	1
en	1
el	1
sofá	1

Palabra	#
colchón	1
el	1,1,1,1
en	1,1
gato	1
perro	1
se	1,1
sofá	1
tumbó	1,1

Palabra	#
colchón	1
el	1
el	1
el	1
el	1
en	1
gato	1
perro	1
se	1

Etapa REDUCE

- Recibe como entrada todos los pares clave valor ordenados que generó la etapa MAP y preparó la etapa intermedia SHUFFLE & SORT.

```
reduce(k,v)
```

- Todos los pares clave valor con la misma clave se mandarán al mismo nodo reduce, que puede ser un nodo distinto al que se generaron.
- En esta etapa lo que normalmente hacemos es realizar agregaciones o cálculos en los valores que vienen de la etapa MAP.
- Finalmente se obtiene un fichero de salida único que se almacena en el HDFS.

MapReduce

Ejemplo:

- Disponemos de un documento muy grande.
- Necesitamos contar el número de veces que una palabra aparece en el fichero.
- Aplicación: Analizar el log de un servidor web para buscar las direcciones IP que más se repiten.

Ejemplo:

- Vamos a pensar que el fichero de log tiene la siguiente estructura.

```
192.168.130.34,10/10/2023,1,2,0,12,3
10.12.1.4,10/10/2023,1,2,0,12,3
…..
192.168.130.34,11/10/2023,1,2,0,12,3
```

- Queremos obtener el número de veces que aparece una ip.
- Del fichero nos interesa coger la dirección IP, el primer campo.

Ejemplo fase MAP:

- El fichero de log es muy grande y se divide en bloques a lo largo de varios ordenadores dentro del cluster. Esto es una característica de Hadoop.
- En cada ordenador o nodo se va a analizar sólo los datos que tiene almacenados ese mismo nodo (*Data Locality*).
- Se lanza una tarea map en varios nodos, teniendo en cuenta que el dato que se va a procesar esté en ese nodo,
- En este caso, en cada nodo, se va recorriendo el fichero log, y coloca como salida por cada línea, la dirección IP y un 1 que corresponde al número de veces que aparece esa IP (en esa línea sólo 1 vez).

MapReduce

Ejemplo Fase Map:

Input

```
192.168.130.34,10/10/2023,1,2,0,12,3
10.12.1.4,10/10/2023,1,2,0,12,3
…..
192.168.130.34,11/10/2023,1,2,0,12,3
```

Output

```
192.168.130.34,1
10.12.1.4,1
…..
192.168.130.34,1
```

Ejemplo fase SHUFFLE & SORT:

- Después de la etapa Map, y en el mismo nodo del MAP, se pasa por una etapa llamada Shuffle, en la que se asigna cada par de claves a una tarea Reduce. Esto se hará a través de una función HASH y del módulo sobre el número total de tareas Reduce que se van a ejecutar, de tal forma que la misma clave, en este caso la IP, siempre vaya al mismo nodo ejecutor de la tarea reduce.
- A continuación se envían por la red los pares clave valor a los nodos ejecutores de las tareas reduce.

Ejemplo fase SHUFFLE & SORT:

- Cuando el nodo que va a ejecutar la tarea Reduce recibe los datos, y antes del Reduce, ejecuta la tarea SORT, en la que se ordenan los datos por la clave para que la etapa Reduce los reciba ordenados.
- En este caso se ordenan los datos por la clave, dirección IP, y por lo tanto aparecen seguidas todas las líneas en las que aparece una misma dirección IP.
- En algunos tipos de ejecuciones, como veremos más adelante, además de ordenar los datos, los agrupará.

Ejemplo fase REDUCE:

- Por último el Reduce. En este caso irá recorriendo el fichero buscando líneas consecutivas con las misma IP, contando el número de veces que se repite, y anotándolo después.
- La salida es la IP, una coma, y el número de veces que aparece esa IP.
- En el ejemplo, hay una IP que aparece dos veces y otra que aparece sólo 1, y por eso queda la salida así.

Input

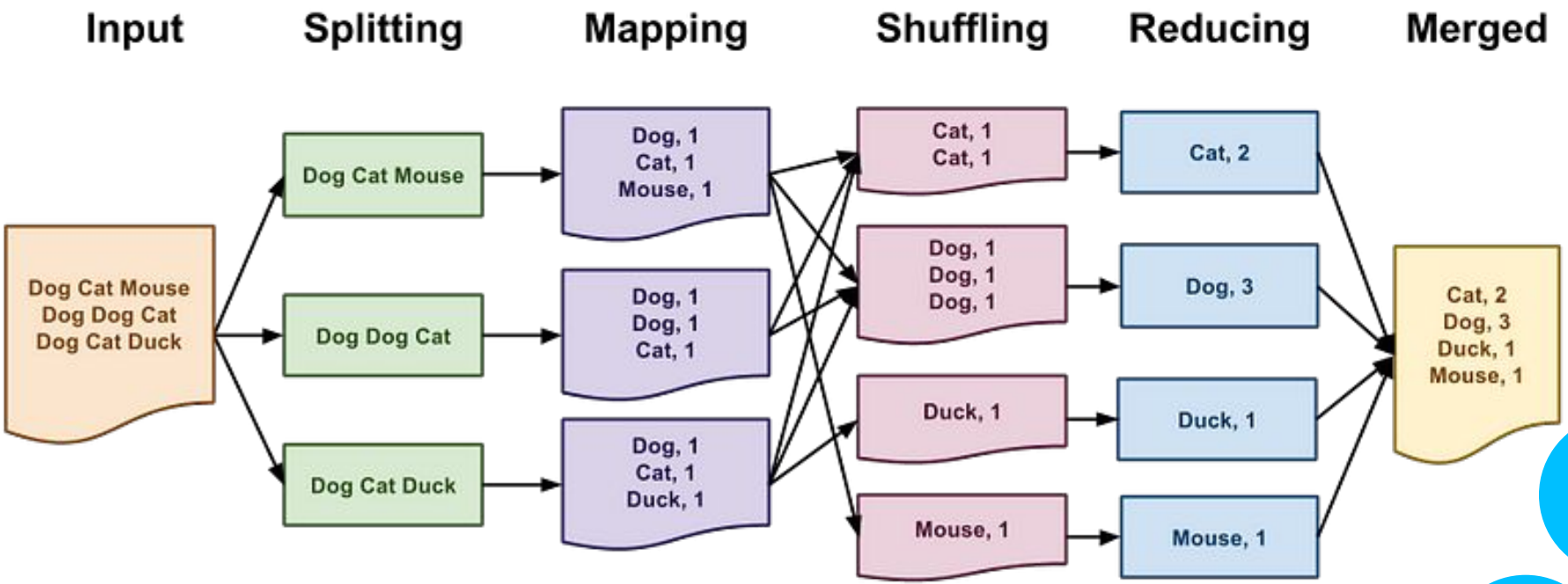
```
10.12.1.4,1
…..
192.168.130.34,1
192.168.130.34,1
```

Output

```
10.12.1.4,1
…..
192.168.130.34,2
```

MapReduce

Ejemplo: contador de palabras



- Ejemplo MapReduce

contador de palabras

Datos de entrada

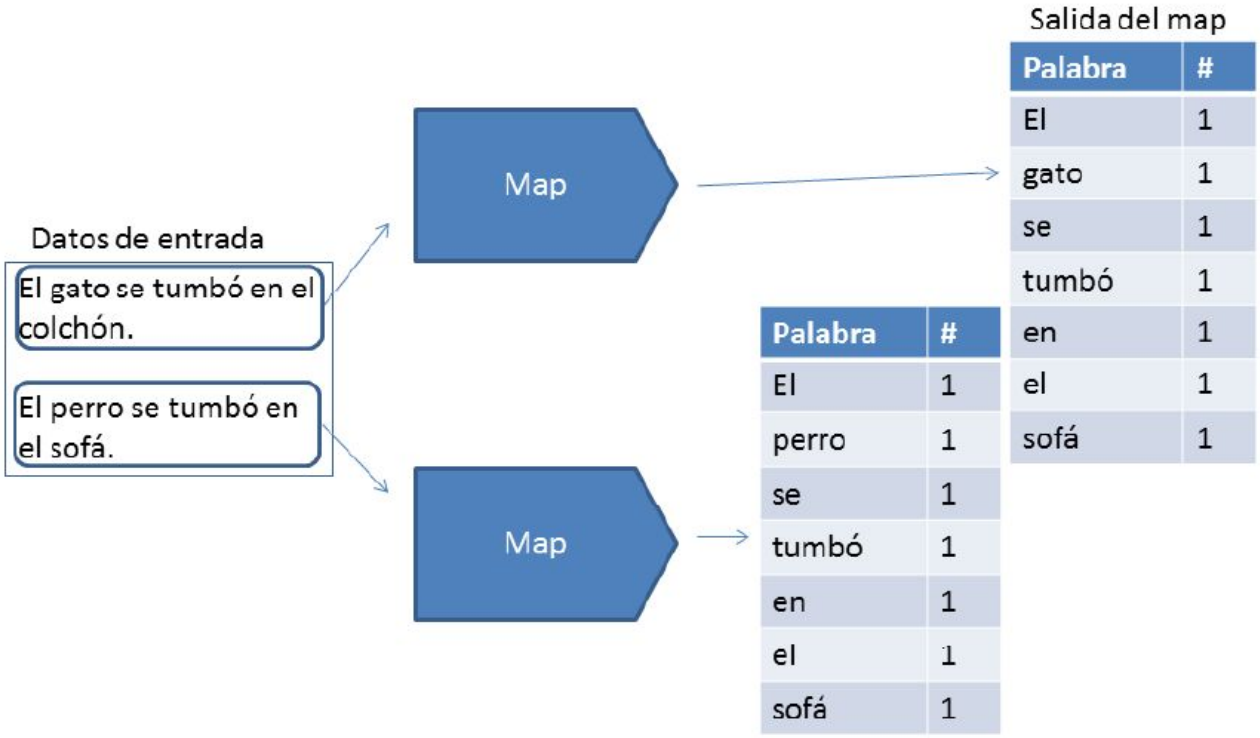
El gato se tumbó en el colchón.

El perro se tumbó en el sofá.

Resultado

Palabra	#
colchón	1
el	4
en	2
gato	1
perro	1
se	2
sofá	1
tumbó	2

PASO 1: MAP


Datos de entrada
El gato se tumbó en el colchón.
El perro se tumbó en el sofá.
Map
Map
Salida del map
Palabra
#
El 1
gato 1
se 1
tumbó 1
en 1
el 1
sofá 1
Palabra
#
El 1
perro 1
se 1
tumbó 1
en 1
el 1
sofá 1

PASO 2: SHUFFLE AND SORT (JAVA)

Palabra	#
El	1
gato	1
se	1
tumbó	1
en	1
el	1
sofá	1
El	1
perro	1
se	1
tumbó	1
en	1
el	1
sofá	1

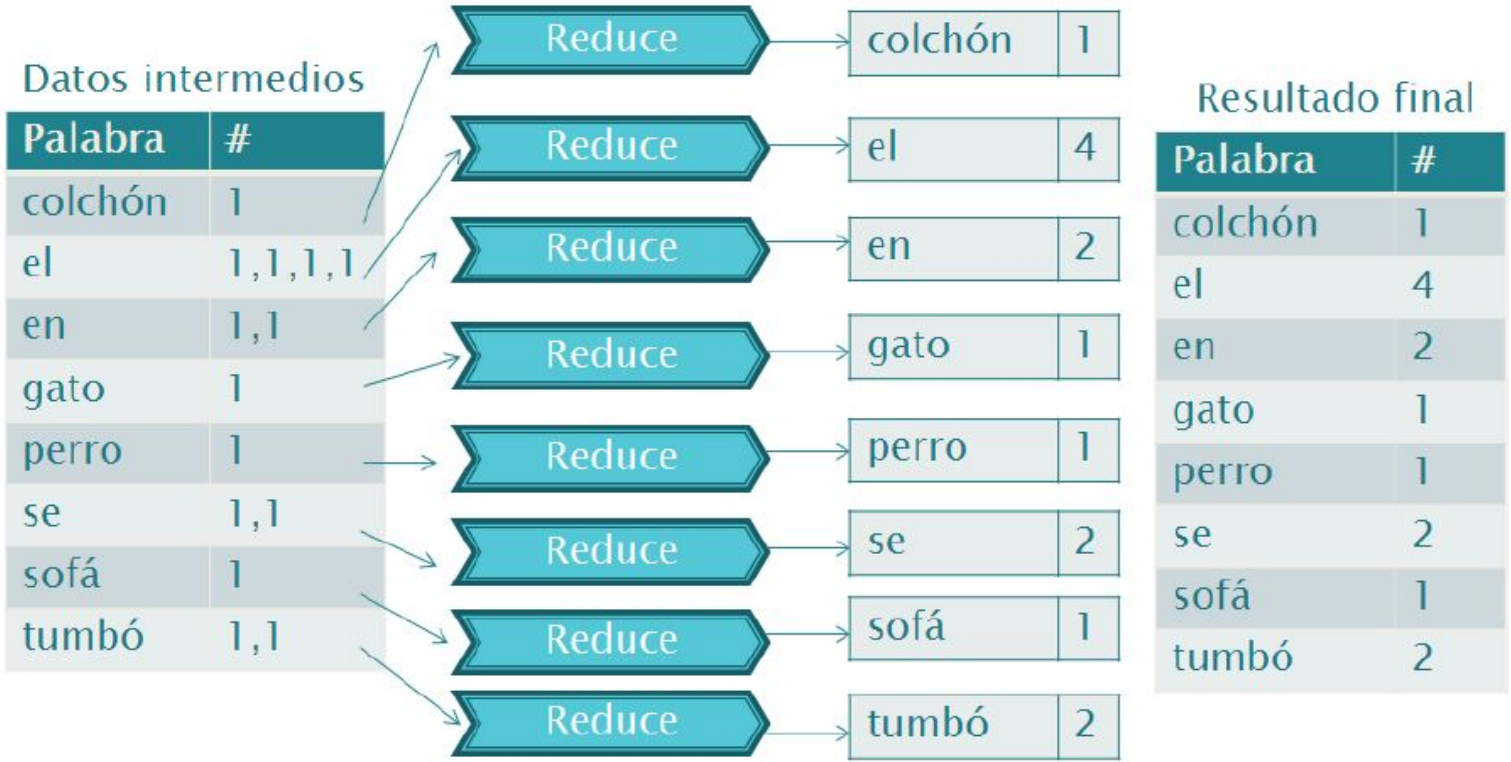
Salida del map

Shuffle & sort

Datos intermedios

Palabra	#
colchón	1
el	1,1,1,1
en	1,1
gato	1
perro	1
se	1,1
sofá	1
tumbó	1,1

PASO 3: REDUCE (JAVA)



- PASO 2: SHUFFLE AND SORT (PYTHON)

Palabra	#
El	1
gato	1
se	1
tumbó	1
en	1
el	1
sofá	1
El	1
perro	1
se	1
tumbó	1
en	1
el	1
sofá	1

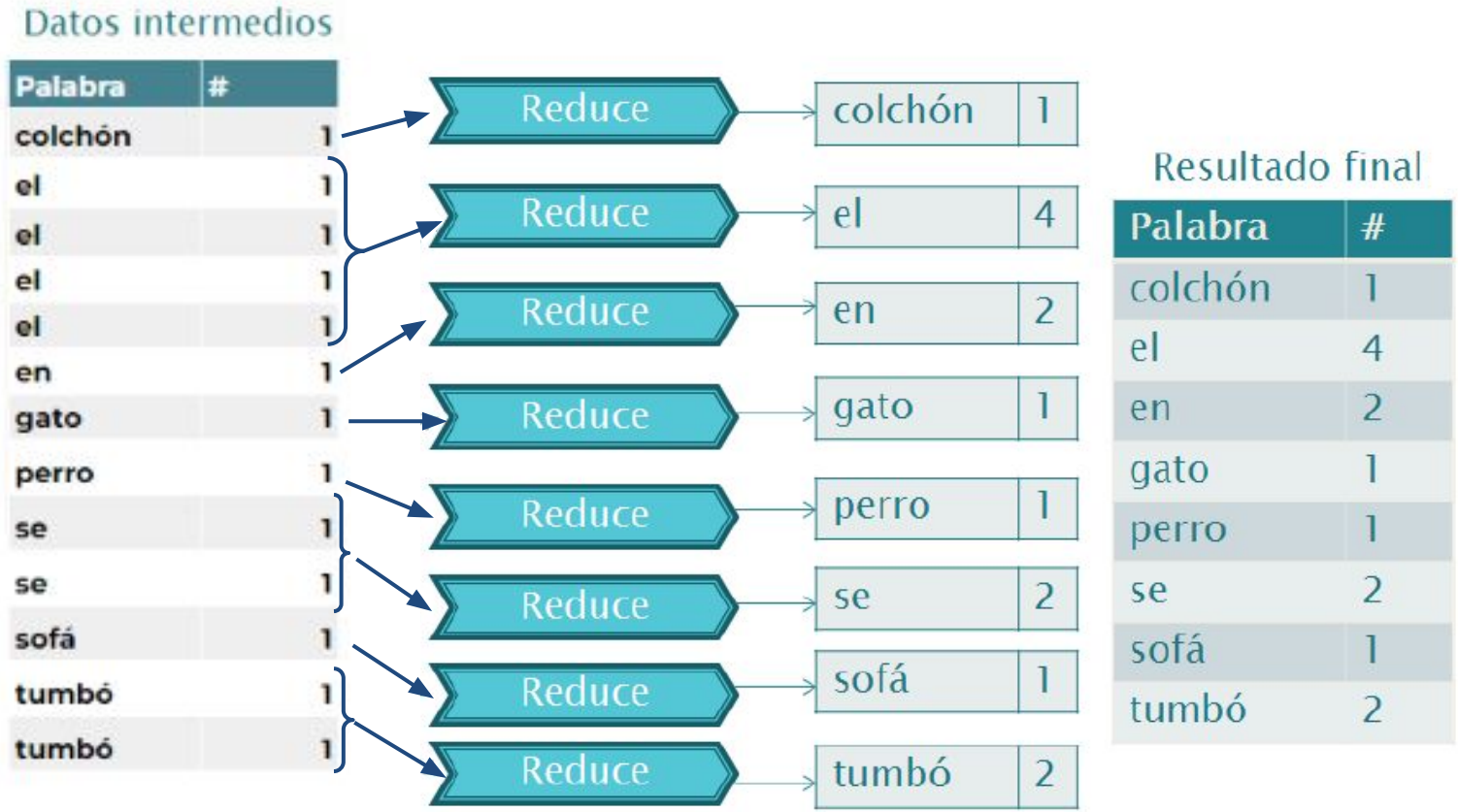
Salida del map

Shuffle & sort

Datos intermedios

Palabra	#
colchón	1
el	1
el	1
el	1
el	1
en	1
gato	1
perro	1
se	1
se	1
sofá	1
tumbó	1
tumbó	1

- PASO 3: REDUCE (PYTHON)



MapReduce

Sistemas de Big Data

Sistemas de
Big_Data

Curso Especialización
Inteligencia_Artificial y
Big_Data