Laura.mestres@salle.url.edu

Presentación asignatura 26/02

Hadoop ya no se usa.
RDMS – Base de datos relacional
HDFS – Hadoop Distribution File System

¿Qué es big data? 1. Introd al Big Data

Volum → Moltes dades
Varietat → De molts tipus i fonts diferents
Velocidad → Analitzades ràpidament
Veracidad → Les dades son verdaderes
Valor → Aporten un valor (informació)

Exemple Volumetria:
- Telescopis
- Sequenciador genètic
- Netflix: Analisi de recomanacions a través de visualitzacions
- Google: Anàlisi de cerques, de mapa (trànsit, recorreguts, estades...)
- Amazon: Compres
- Facebook:
- Ebay: Anàlisi de comprres
- Spotify:

Exemple Volum + Velocitat:
- Manteniment predictiu: Predir el manteniment d'algun component.
- Gestió d'alarmes: Analitzar el procés d'algun component industrial i fer saltar l'alarma si falla. Per exemple si es filtren pèrdua de seguretat de arxius.
- Publicitat dirigida: Després de que una persona estigui comentant o buscant un cert producte, posar-li de seguida la publicitat encaminada.
- Detecció de frau: Analitzar quan un assegurador surt rentable o no. Analitzar gent que gasta diners a on no toca perquè li han robat el compte o perquè està fent un comportament il·legal.

Variabilidad de datos
Estrucutrado. De bases de datos relacionales. Estructurados en tablas.
Desestructurado: Imágenes, Vídeos, Texto
Semi estructurado: Logs de un servidor. Formato JSON

BIG DATA: Búsqueda de soluciones para almacenar y analizar datos (estructurados y no estructurados) de una manera económica, escalable y tolerante a fallos con el objetivo de descubrir información nueva y valiosa.

SIRVE PARA:

• **Recopilar:** Big data facilita el paso para recopilar datos variados sin procesar (como registros, dispositivos móbiles…) en tiempo real o por lotes.

• **Almacenar:** Big data proporciona un repositorio seguro, escalable y duradero donde almacenar datos antes y después de procesarlos.

• **Procesar y Analizar:** Procesar (classificar, acumular, agregar…) datos para el consumo.

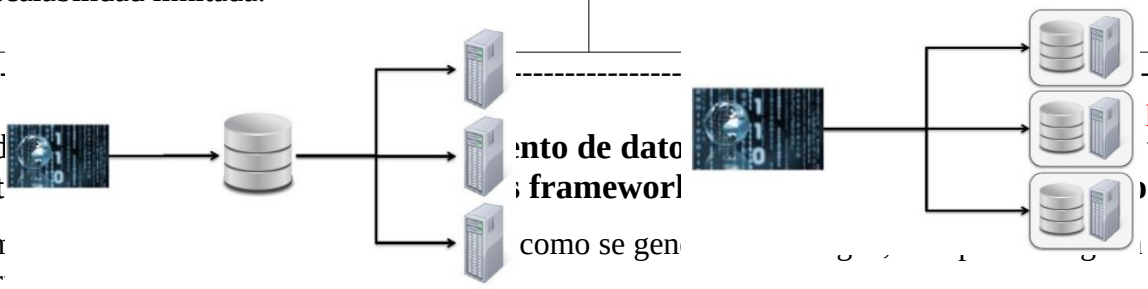
• **Consumir y Visualizar:** Consumir información visualizandola o inteligencia de negocio.

Ejemplos de tipos de procesos:
• Transformación técnica: Cambio formato, adaptación de los datos a una tecnología de uso.
• Transformación funcional. Aplicar una función a los datos: Filtrar, Agregar, Enriquezer

Ejemplos de tipos de análisis:
• Descriptivo. *¿Qué ha pasado y por qué?* → Cuadro de mando
• Predictivo. *¿Probabilidad de que ocurra un evento?* → Alertas, Fraude, Mantenimento preventivo
• Prescriptivo. *¿Qué debería hacer si pasa x?* → Sistemas de recomendación.

Sistemas tradicionales:
Escalado vertical.

Sistemas tradicionales:	Sistema Distribuido Big Data
Escalado vertical → Escalabilidad limitada - Ordenadores más grandes (más: rápidos, memoria, potentes...)	**Escalado horizontal** - Más ordenadores. Múltiples máquinas para un mismo trabajo. Sistema distribuido. - Reto de complejidad de programación. Mantenimiento de dato y procesamiento en sincronía. ...
Datos centralizados - Procesar datos en durante una copia causa aumento del tiempo de copia. - Aumento de complejidad del mantenimiento. - Escalabilidad limitada.	**Datos distribuidos** en su almacenaje - Computación realizada en la subdivisión del dato.



Had **nto de dato** **una**
plat **framewor** **)**

Perm como se gen
restr

pág 35

• Datos distribuidos en su almacenado/ingesta.
• Procesos de computación en el lugar de su almacenaje.
• Hadoop se despliega sobre un conjunto de ordenadores (**clúster)** que trabajan de manera conjunta.
Se despliega en servidores profesionales montados en un rack, ordenadores domésticos, raspberrys...
• Cada una de las máquinas que forman el clúster se conoce como **nodo**.
• Hadoop proporciona fiabilidad, disponibilidad, escalabilidad y tolerancia a fallos (si falla un nodo, el sistema sigue funcionando. Se reasignan las tareas a otros clústeres).

Frameworks principales:
• Map reduce (para batch)
• Apache Spark (batch y real time)
Características Batch:
• La información se vuelca periódicamente (cada min, hora, semana, mes...) en un clúster.
• El procesamiento no se realiza en el momento en el que el dato se genera en la fuente.
• La respuesta al procesamiento puede demorarse un tiempo.
Aplicaciones: Generación de reportes y cuadros de mandos, análisis de datos históricos...

Características Real time:

• La información se vuelca en un clúster en el tiempo en el que se genera en la fuente.
• El dato es procesado en el momento en que llega al sistema.
• La respuesta al procesamiento es practicante inmediata.
Aplicaciones: Sistemas de alertas, publicidad dirigida, IoT...

Sistemas SQL y NOSQL 1. Introd al Big Data pag 55

SQL:

• Permite integrar Bases de datos Relacionales con muchos sistemas a través de **driver** (elemento de comunicación a entre aplicaciones y bases de datos).

Transaccional

- La transaccionalidad es una transformación que sigue las propiedades ACID
  - **Atomic**: "Todo o nada"
  - **Consistent**: Movimiento de datos de un estado a otro sin posibilidad de estados intermedios
  - **Isolated**: Cada transacción se ejecuta de manera individual sin posibilidad de mezclarse con las demás
  - **Durable**: Una vez la transacción se ejecuta con éxito, los cambios no se perderán y estarán disponibles para futuras transacciones

Schema 1. Introd al Big Data pag 65

New Vocabulary

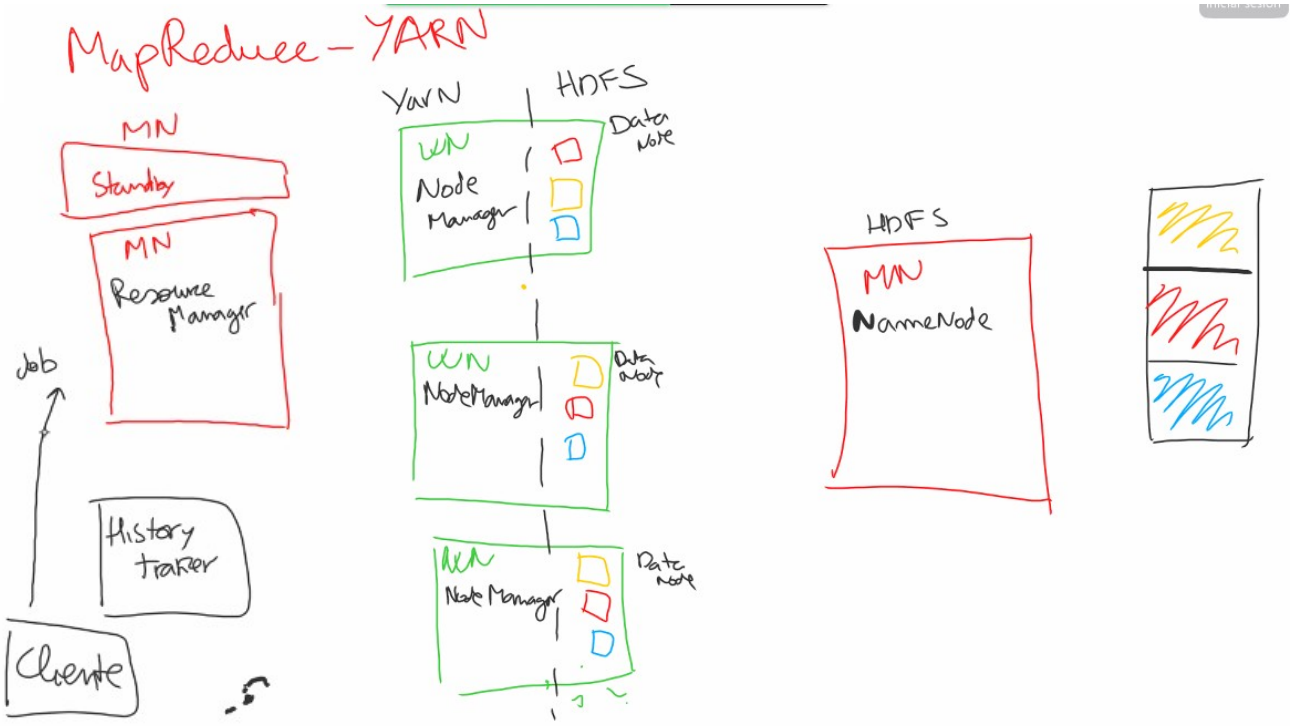
Framework (marco de trabajo): Un framework es una estructura o conjunto de herramientas que proporciona una base para el desarrollo de software. En el contexto de big data, un framework puede ser un conjunto de bibliotecas y herramientas que facilitan el procesamiento y análisis de grandes cantidades de datos, como Apache Hadoop o Apache Spark. Estos frameworks proporcionan una estructura y funcionalidades específicas para realizar tareas relacionadas con el big data.

API (Interfaz de Programación de Aplicaciones): Una API es un conjunto de reglas y herramientas que permite a diferentes aplicaciones comunicarse entre sí. En el contexto de big data, una API podría ser un conjunto de funciones y procedimientos que permiten a los desarrolladores interactuar con un servicio o plataforma de almacenamiento de datos. Por ejemplo, una API de bases de datos podría permitir a una aplicación consultar y manipular datos almacenados.

Driver: Es un programa o componente que controla la coordinación en la ejecución de un programa distribuido. Por ejemplo, en Apache Spark, el driver es responsable de coordinar las tareas y de interactuar con el clúster para distribuir el trabajo entre los nodos.

Nodo: Es una máquina individual en un clúster. Puede haber nodos de almacenamiento, nodos de procesamiento y otros tipos de nodos. Los nodos trabajan en conjunto para realizar tareas de procesamiento y almacenamiento distribuido.

Clúster: Un clúster es un conjunto de nodos interconectados que trabajan juntos para procesar y almacenar grandes conjuntos de datos. Cada nodo en el clúster tiene sus propios recursos y capacidades, y el clúster en su conjunto ofrece una mayor potencia de procesamiento y almacenamiento que un sistema individual. En big data, los clústeres son esenciales para gestionar grandes volúmenes de datos de manera eficiente y escalable.



YARN (Yet Another Resourse Negotiation)
MN = Manager.

MN Standby = Resource Manager que está apagado a la espera por si el Resource Manager principal cayese.

History Tracker: Registra los datos históricos de los procesos.

Client: Se conecta al Resouce manager para mandarle un Job. Por ejemplo: Cuenta cuantas palabras hay en cada línea.

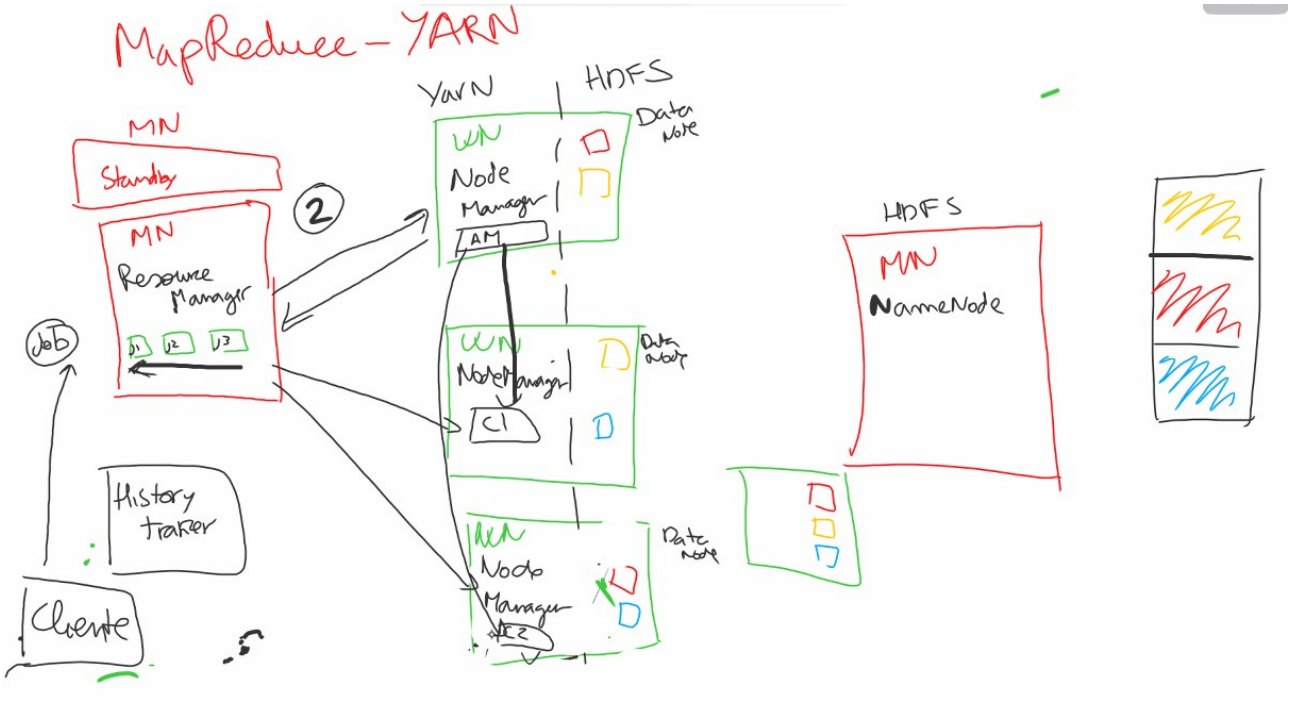
Ese Job está dividido por distintas tareas, Tasks. Hay dos tipos de Tasks: Map Task (siempre existe. Ejemplo: pasa una matriz de 1 y 0 para contar), Reduce Task (a veces no es necesario, aplica y suma el contaje).

MN Resource Manager: Administra los procesos de los distintos Worker Nodes dándoles las tareas concretas a realizar y administrando en caso de caída de alguno.

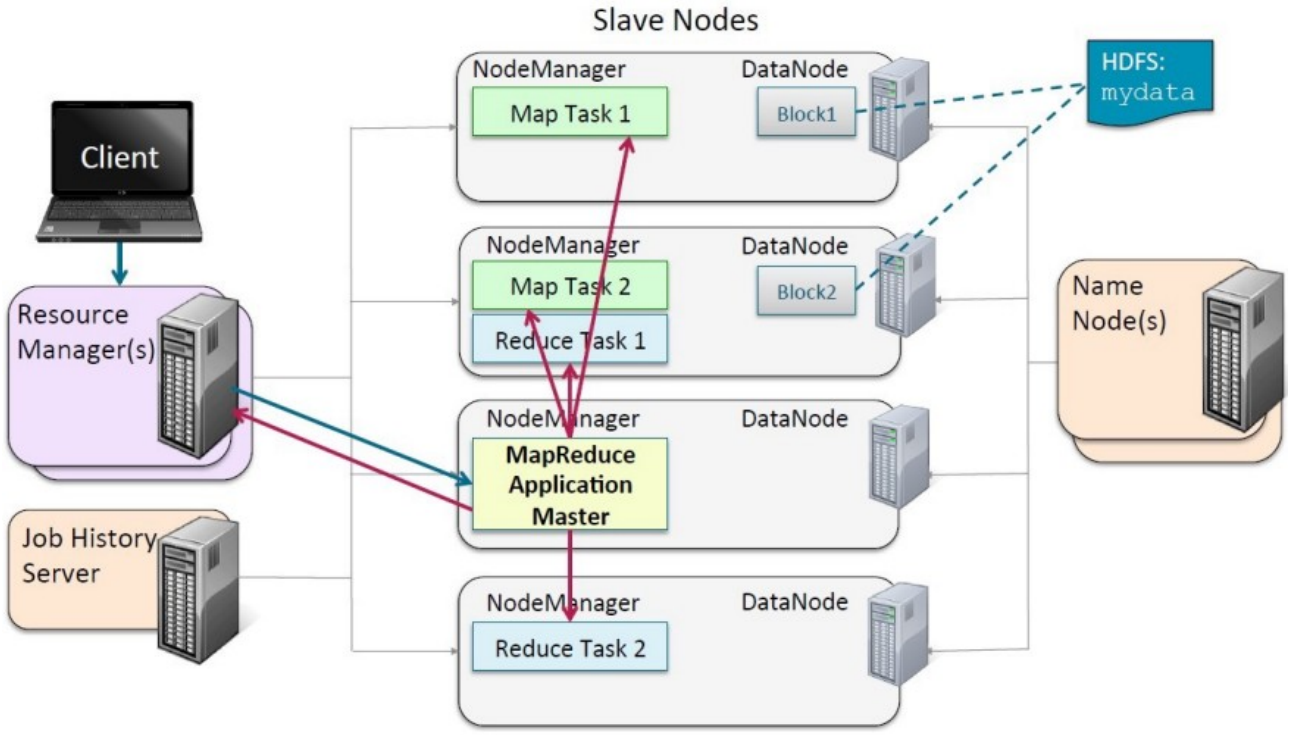
WN = Worker Nodes Slaves

Node Manager: Nodo de conexión con el manager (Resource manager)
Data Node: Nodo de conexión con los datos (Name Node. Es el nodo master de HDSF.)

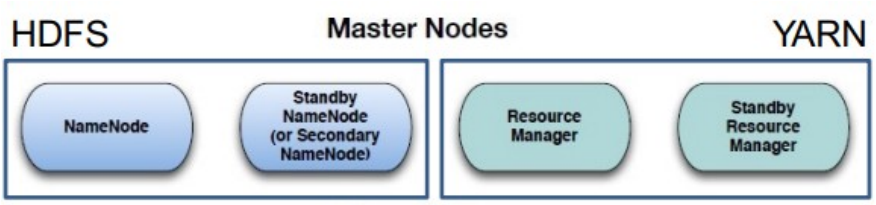
HDFS (Almaceamiento)



AM = Aplication Master: Hay un Worker Node Slave que actúa de comunicador principal (líder) de las órdenes del Resource Manager. Es un contenedor que está en el Node Manager del Worker Node Slave.
El Resource Manager se comunica con el Application Master diciéndole qué tipo de job es necesaria de lo que se deduce qué tipos de recursos que necesitarán. El Application Master consulta si tiene los recursos necesarios y le devuelve mensaje al Resourse Manager y reparte las tareas.



Roles que pueden llegar a tener los Master Nodes.



2. Apache Hadoop. Diapo 80 6/3

Son “write once” porque así no hay que actualizar demasiadas veces las réplicas.

Diapo 90 a 101 11/3

Formatos habituales.

Narrow → Dataset con pocas columnas.
Al escribir tenemos que saber si vamos a acceder mucho a los datos o no y saber las cantidades de datos que tenemos.

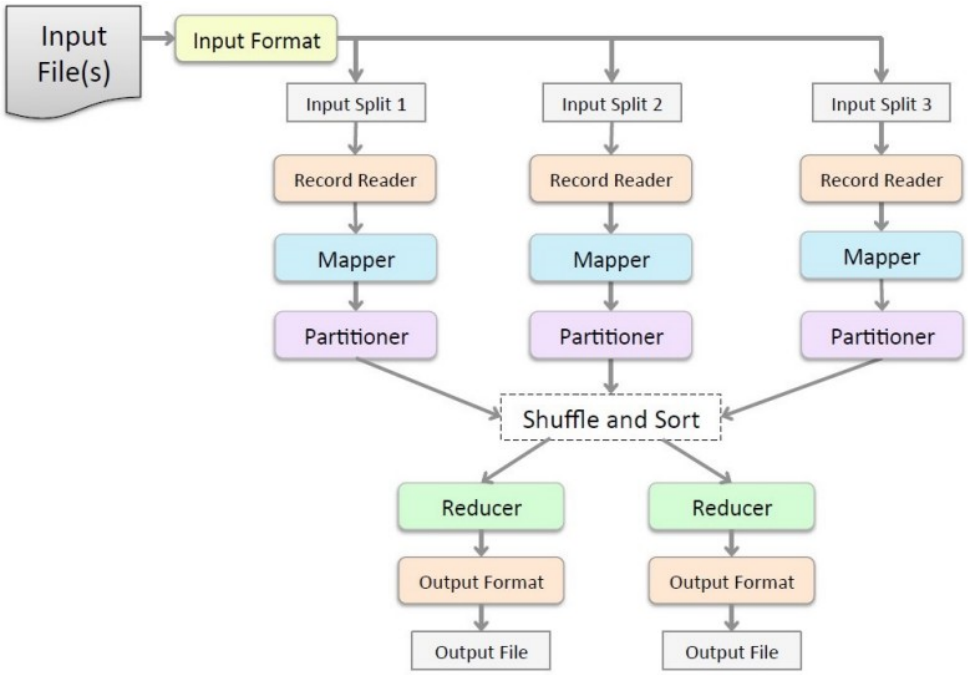
Diapo 102. Mapreduce 11/3

MapReduce es Framework de Java (HDSF para procesar datos).

Características:
• Master Node administrará las tareas de manera automática.
• Todo hadoop de core y MapReduce está escrito en Java.
• El desarrollador solo se concentra en programar y se abstrae de las tareas de mantenimiento y administración del clúster.

Terminología
• Job → Ejecución completa de Mappers y Reducers.
• Task → Ejecución de Mapper o Reduccer.
•

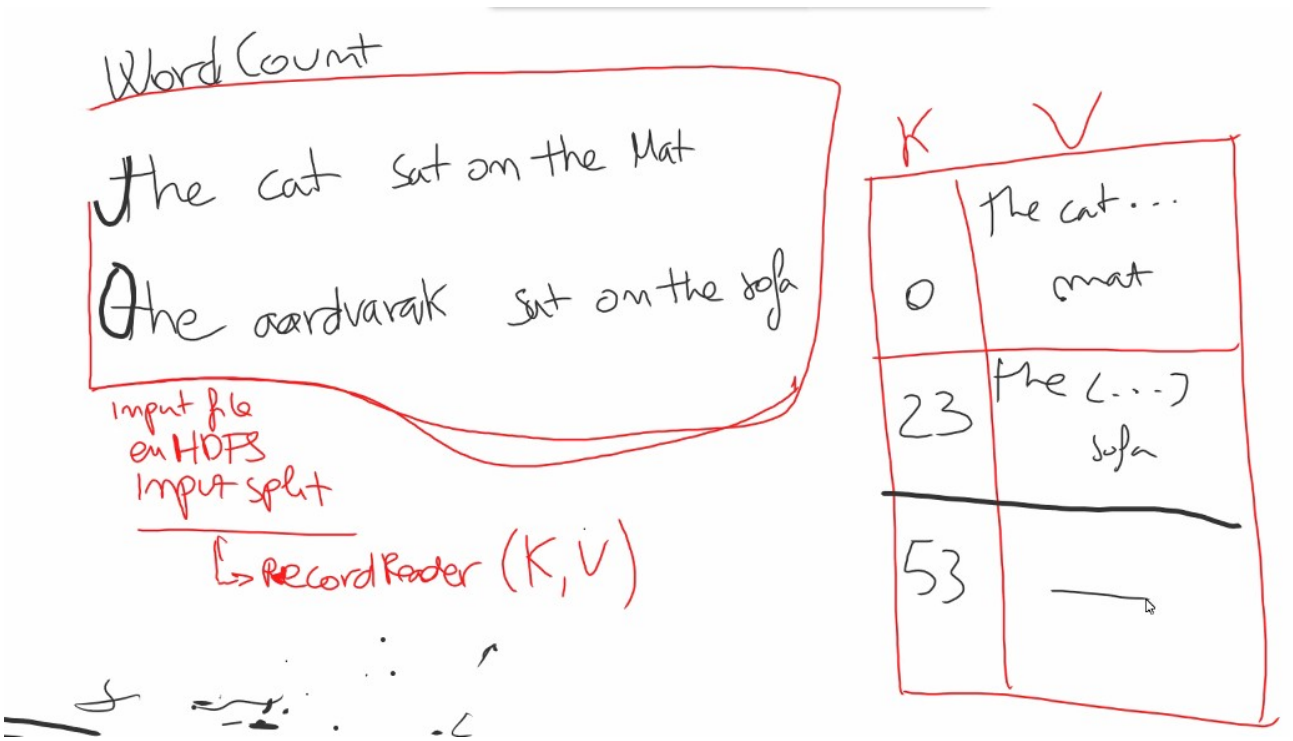
Fases:
1) Map
• Cada Map opera en un solo bloque de HDFS
• Se ejecuta en el nodo donde se almacena el bloque HDFS
2) Shuffle and Sort
• Consolida y ordena los datos intermedios
• Se ejecuta después de terminar todos los Maps y justo antes de la fase de Reduce
3) Reduce
• Opera contra los datos que vienen de la fase Shuffle and Sort
• Proporciona la salida final



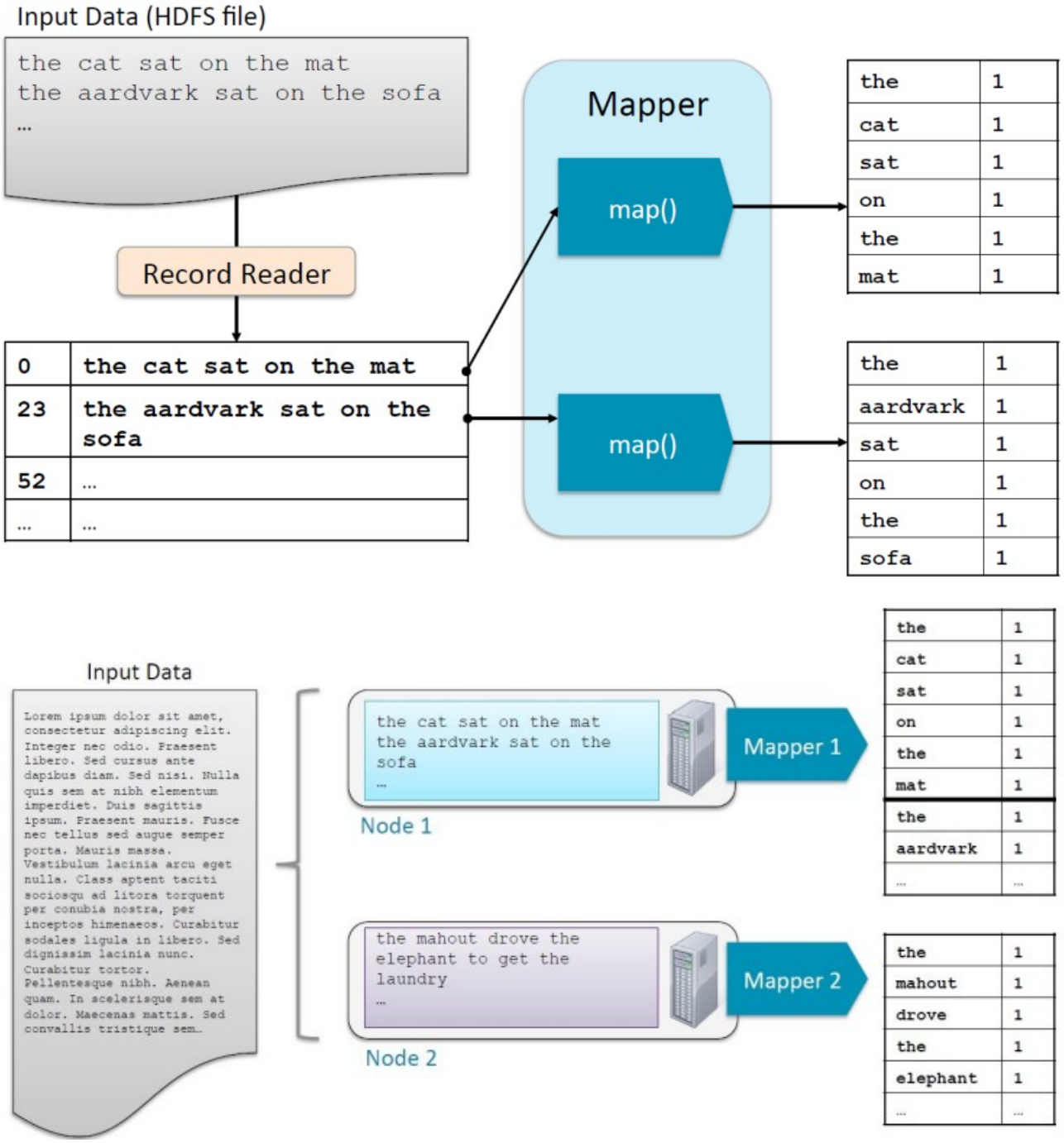
Per exemple: Compta d'aquest llistat de paraules quantes vegades es repeteix cada paraula. Els Mappers analitzen una part del text. Per cada paraula que troben li afageix un 1 al costat.
Per exemple "hello11111" "you11" … "hello1" "you111" … "hello11" "you1"

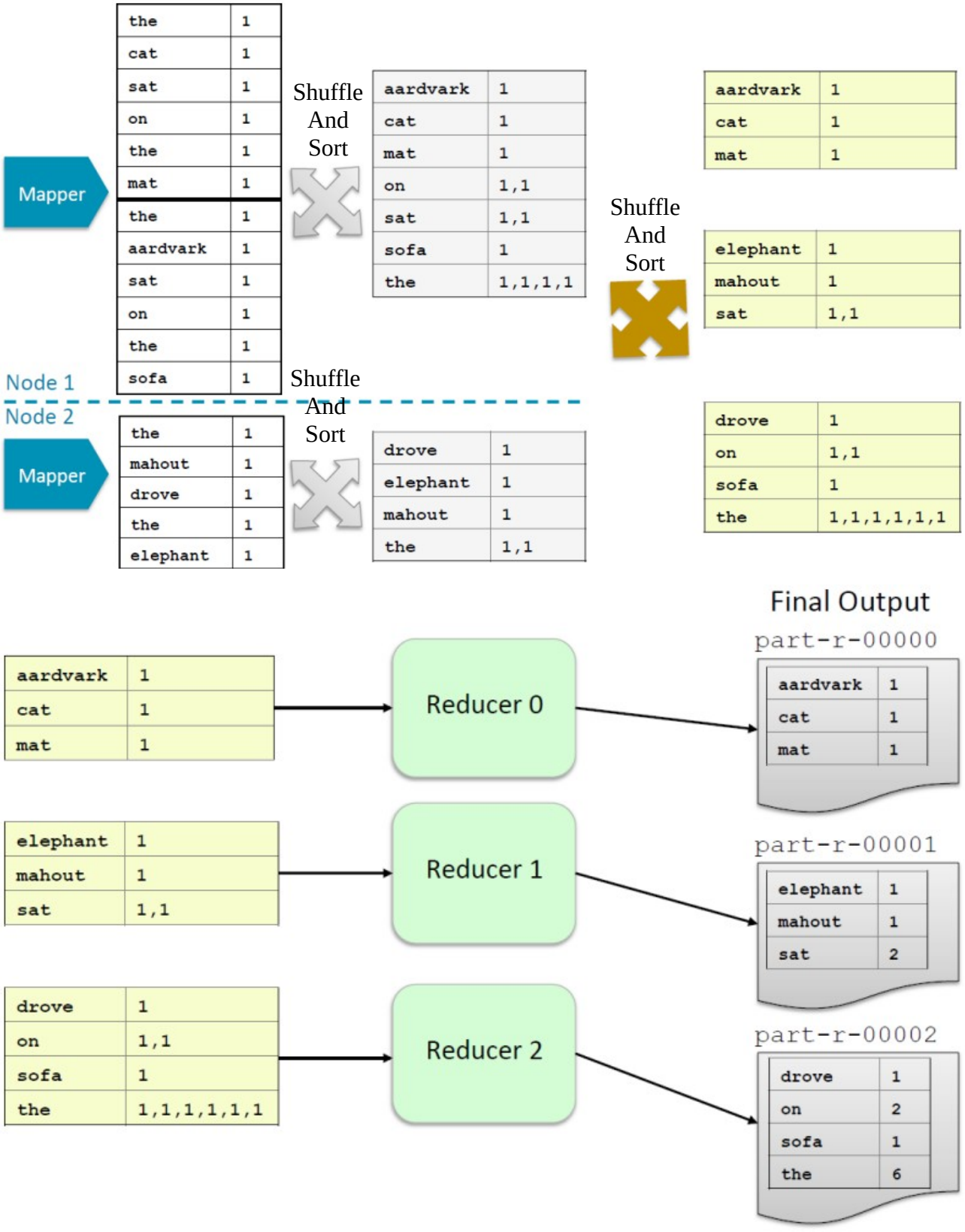
Shuffle and Sort agafa les paraules i reparteix cada paraulaal Reducer.

Després la Reducer ajunta tots els resultats de cada paraula i els suma.



El key value es la cantidad de caracteres que contamos. 0 porque empezamos a leer en la "T" de "the cat sat on the mat". Y seguimos con el 23 porque seguimos en la "T" de "the aarvarak sat on the sofa".


Input Data (HDFS file)
the cat sat on the mat
the aardvark sat on the sofa
…
Record Reader
0
the cat sat on the mat
23
the aardvark sat on the sofa
52
…
…
…
Mapper
map()
map()
the 1
cat 1
sat 1
on 1
the 1
mat 1
the 1
aardvark 1
sat 1
on 1
the 1
sofa 1
Input Data
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. Praesent libero. Sed cursus ante dapibus diam. Sed nisi. Nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur sodales ligula in libero. Sed dignissim lacinia nunc. Curabitur tortor. Pellentesque nibh. Aenean quam. In scelerisque sem at dolor. Maecenas mattis. Sed convallis tristique sem…
the cat sat on the mat
the aardvark sat on the sofa
…
Node 1
Mapper 1
the mahout drove the elephant to get the laundry
…
Node 2
Mapper 2
the 1
cat 1
sat 1
on 1
the 1
mat 1
the 1
aardvark 1
…
the 1
mahout 1
drove 1
the 1
elephant 1
…


Mapper
Mapper
Node 1
Node 2
Shuffle And Sort
Shuffle And Sort
Shuffle And Sort
Reducer 0
Reducer 1
Reducer 2
Final Output
part-r-00000
part-r-00001
part-r-00002

Hadoop Streaming API

Permite ejecutar jobs de map reduce con lenguajes diferentes a Java (Ruby, Python, Perl…)

• Ventajas:

- No hace falta saber aprender Java.
- Se pueden usar librerías propias de cada lenguaje.

• Desventajas:

- Peor rendimiento.
- Solo se puede tratar texto en el imput de los jobs.

(diapo 58 a 63)

Tool runner

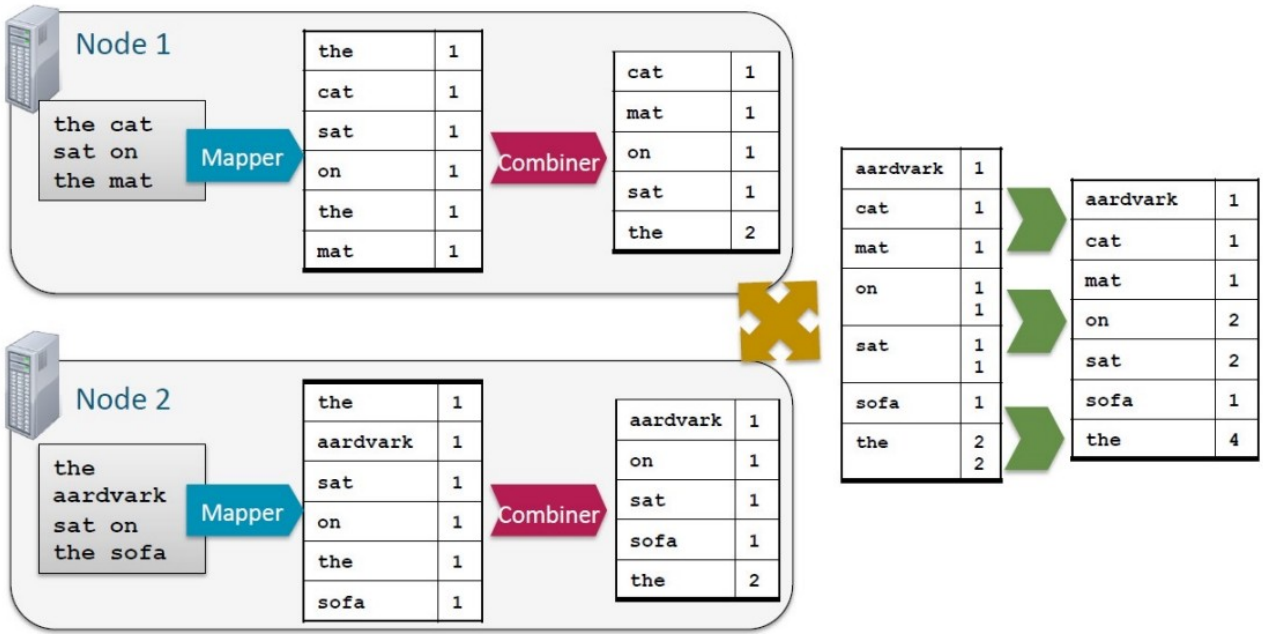
Permite especificar opciones en la línea de comandos.

(diapo 67 a 74)

Combiners

Hace un paso intermedio entre el maper y el reducer. Es como una reducción antes de hacer el reducer. Eso ayuda a que la cantidad de datos del maper no sean tan grandes y produzcan demasiado trafico de red.

Ejemplo:

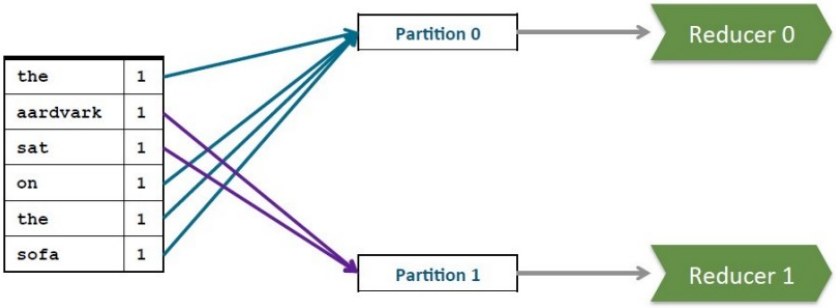


Vigilar porque los combiners no se pueden usar si las propiedades no son asociativas. Como por ejemplo hacer la media después de cada maper no es lo mismo que hacer la media al final.

(diapo 80 a 89)

Partitioner:

Determina en que reducer va cada key. Distribuye las key en los reducers.

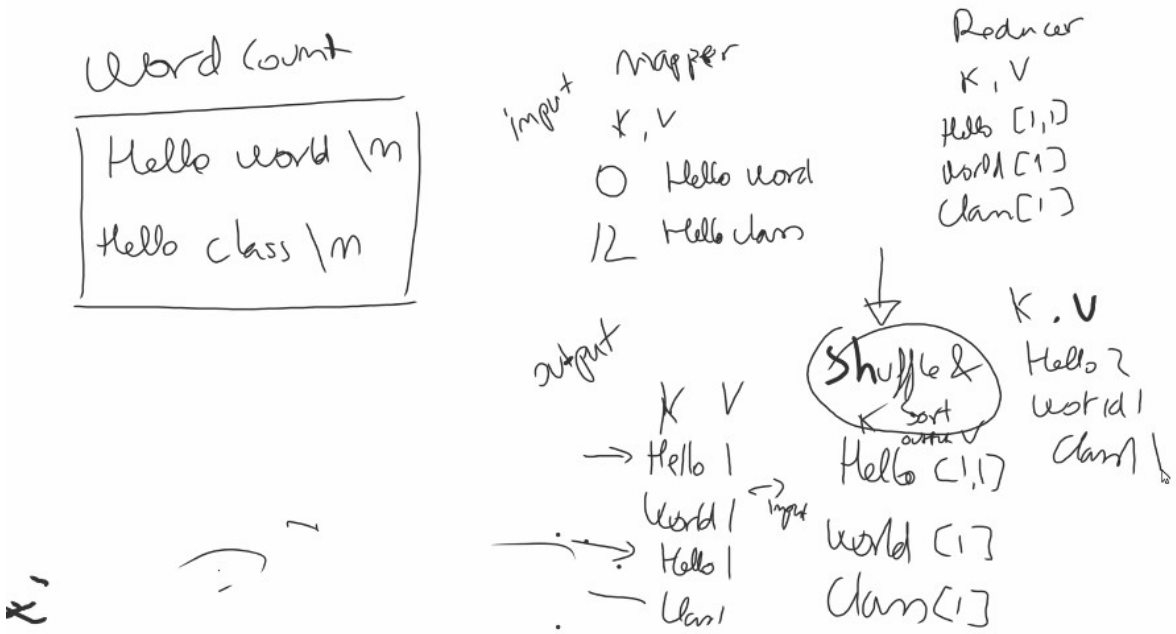


Es importante que las mismas key vayan al mismo reducer para que lo cuente bien. Todos los “the” al particioner 0 por ejemplo.

10/4/24

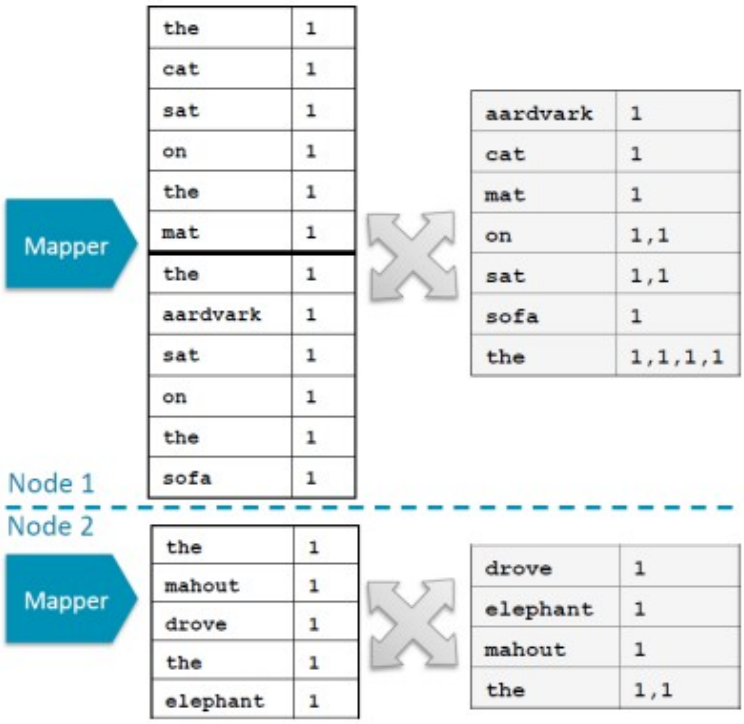
Map reduce se encarga del procesado.
Yarn es el resource allocation.
HDSF se encarga del almacenamiento.

Word Count	*Mapper*		*Shuffle and Sort*		*Reducer*
Hello world \n Hello class \n	→ **Input**: K , V 0 , Hello world 12 , Hello class **Output**: K, V Hello, 1 world, 1 Hello, 1 class, 1	→	**Input**: K, V Hello, 1 world, 1 Hello, 1 class, 1 **Output**: K, V Hello [1,1] world [1]	→	**Input**: K, V Hello [1,1] world [1] class[1] **Output**: K, V Hello, 2 world, 1



class[1] class, 1

El combiner seria:



Combiner 1:
Aardvark 1
Cat 1
Mat 1
On 2
Sat 2
Sofa 1
The 4

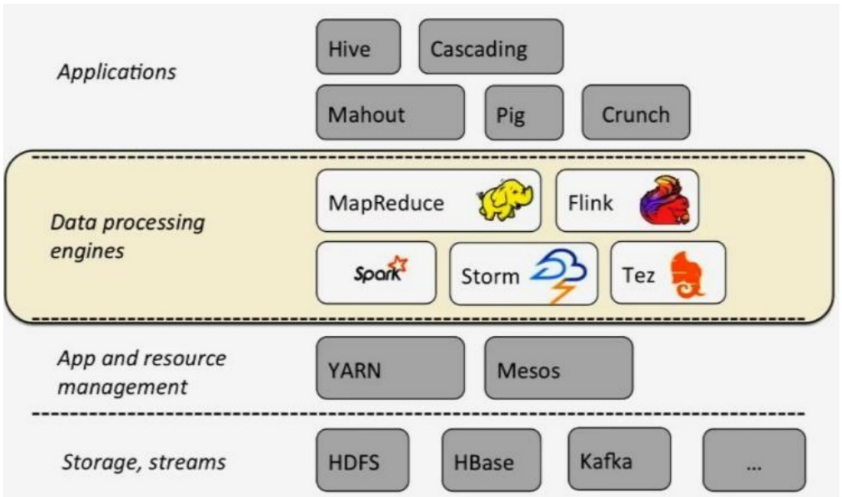
Shuffle and Sort :
Aardvark [1,0]
Cat [1,0]
Mat [1,0]
On [2,0]
Sat [2,0]
Sofa [1,0]
The [4,2]
Drove [0,1]
Elephant [0,1]
Mahout [0,1]

Combiner 2:
Drove 1
Elephant 1
Mahout 1
The 2

Spark puede usar su propio resorce manager (resource allocation) o puede usar otros que sea del archivo de tipo que está usando.

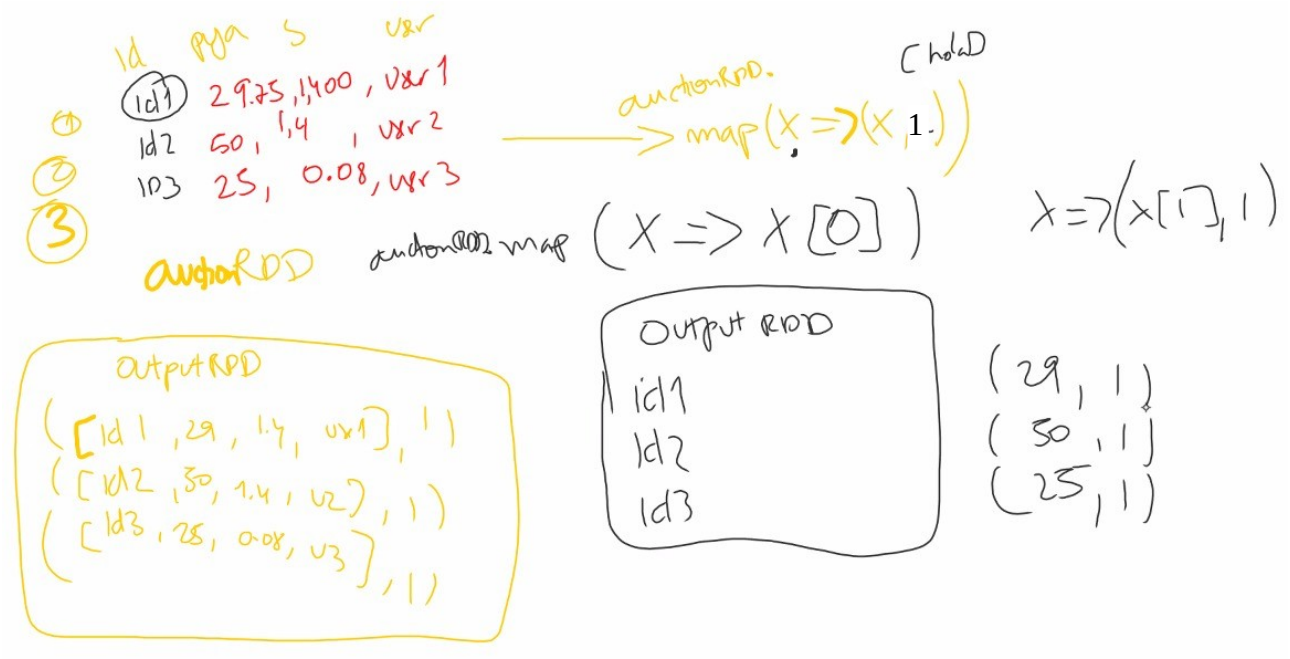
Apache Spark

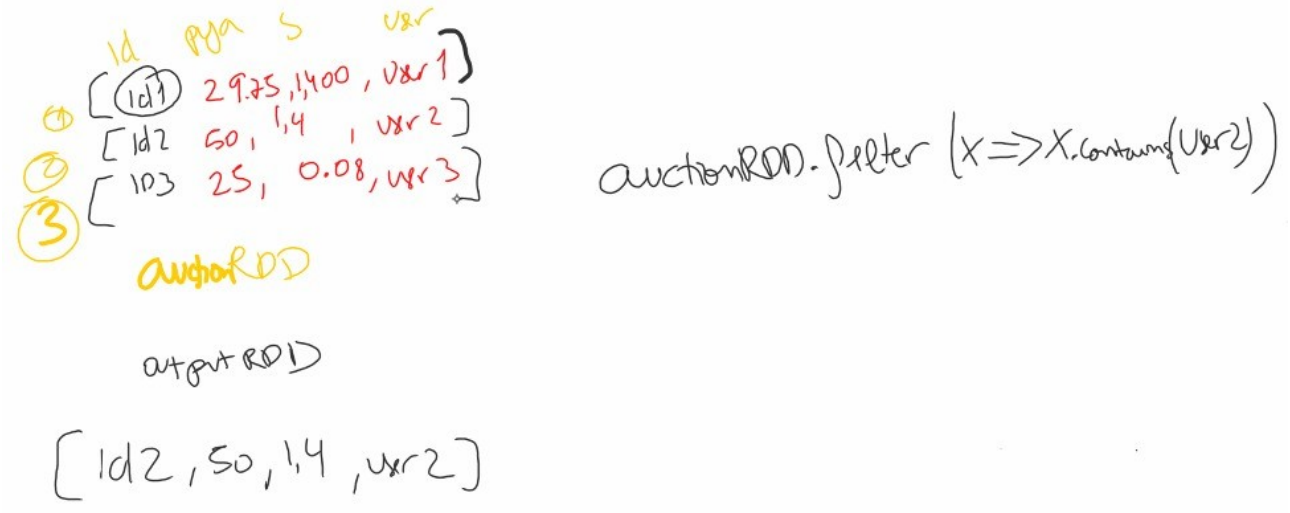
Plataforma para la computación distribuida en clúster, construida por encima de la capa de almacenamiento que extiende las funcionalidades de MapReduce con mas componentes: Streaming, Analítica Interactiva, SQL.



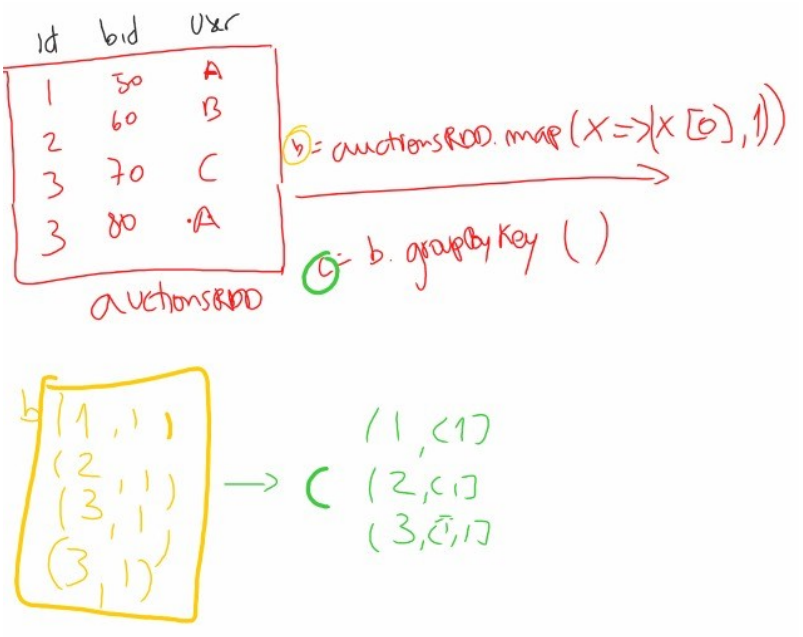
Contra disco → se guarda en HDSF
Contra memoria → se guarda en el nodo. Si se cuelga la info no se pierde porque lo ha guardado el "Graph" "DAG" que tiene los pasos de transformación de los datos guardados como una receta de cocina y al colgarse se sabe a que punto se estaba.

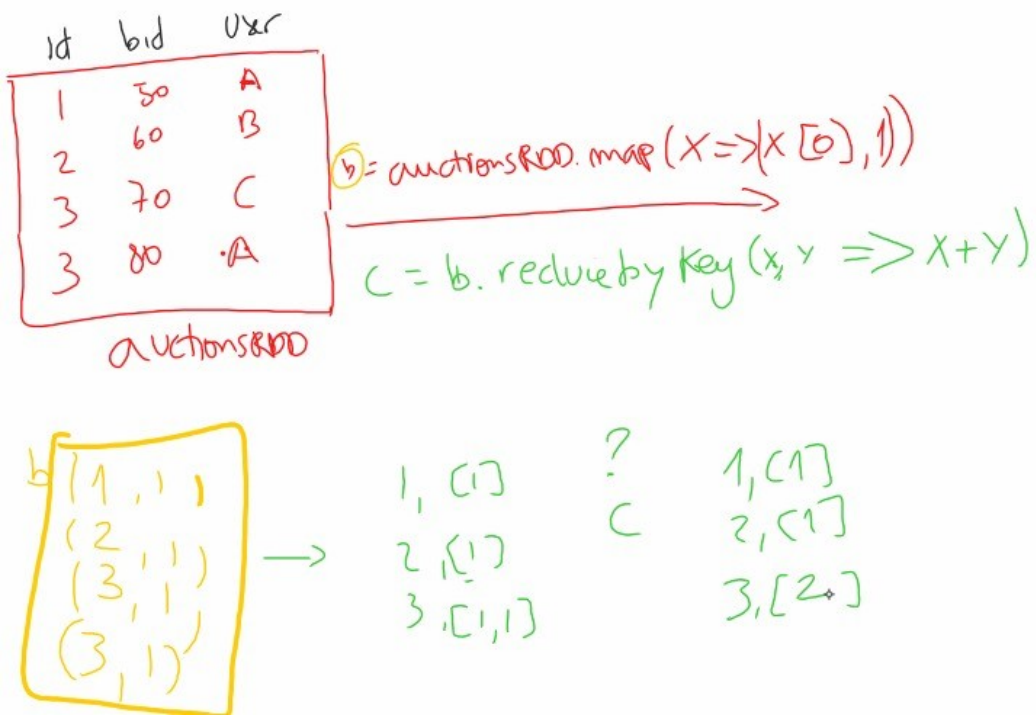
RDD (Resilient Distributed Dataset)**. Transformaciones.** Devuelven un RDD



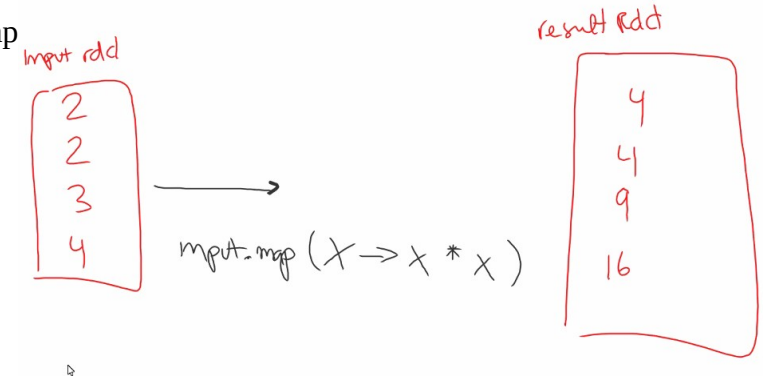



id pga s usr
[id1 29.75, 1,400, usr1]
[id2 50, 1,4 , usr2]
[id3 25, 0.08, usr3]
auctionRDD
outputRDD
auctionRDD.filter(x => x.functa)
x => x.funct a
funct a ()
{ if user 1 in
x[3] return x}
funct a() {
if usr 1 in
X[3]
return x }


id bid usr
1 50 A
2 60 B
3 70 C
3 80 A
auctionsRDD
b = auctionsRDD.map(x => (x[0],1))
c = b.groupByKey()
b
(1,1)
(2,1)
(3,1)
(3,1)
(1,<1>
(2,<1>
(3,<1,1>


id bid usr
1 50 A
2 60 B
3 70 C
3 80 A
auctionsRDD
b = auctionsRDD.map(x => (x[0],1))
c = b.reduceByKey(x,y => x+y)
b
(1,1)
(2,1)
(3,1)
(3,1)
1,[1]
2,[1]
3,[1,1]
?
c
1,[1]
2,[1]
3,[2]

Transformación map

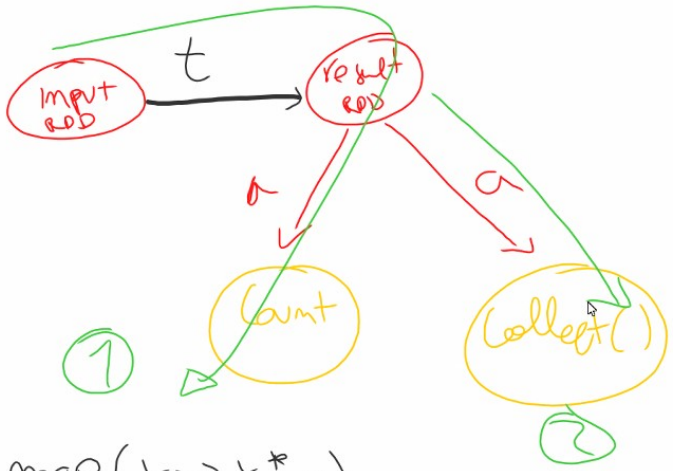


Pág 77 a 93. Apache Spark.

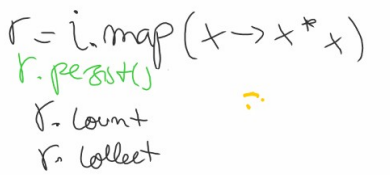
RDD (Resilient Distributed Dataset)**. Acciones.** No devuelve una RDD

Las acciones se realizan una vez y no se guardan en memoria. Cada vez que se llama una acción, se empieza el proceso desde el principio.

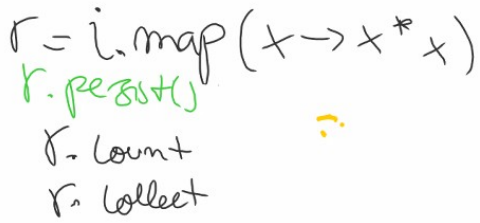
Lazy Evaluated: No se realiza ninguna transformación hasta que no se realize antes una acción.



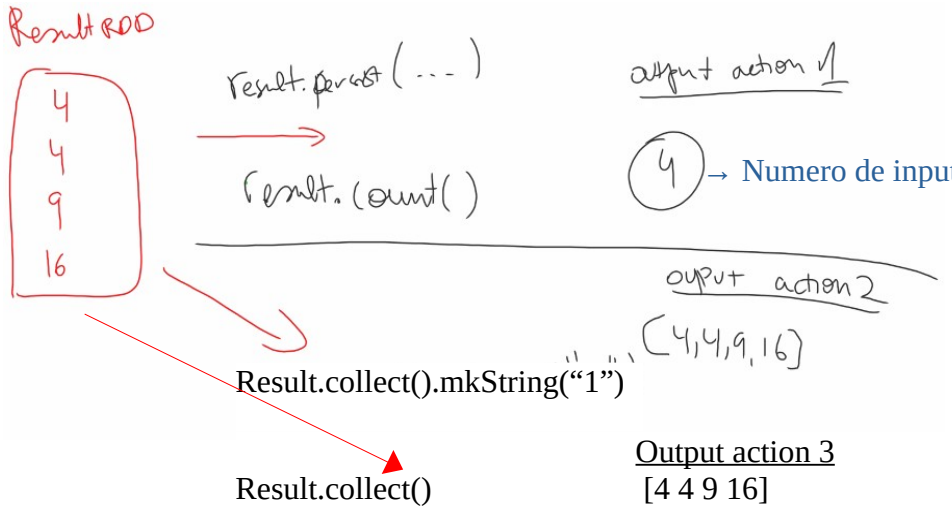
Por lo tanto, el siguiente **DAG** (Gráfico Acíclico Dirigido)



Lo que hace es:
• No ejecutar el map porque es una transformación.
• Hace el count (1)
• Hace el collect (2)



Con persist → Guarda en el estado que está el cluster.

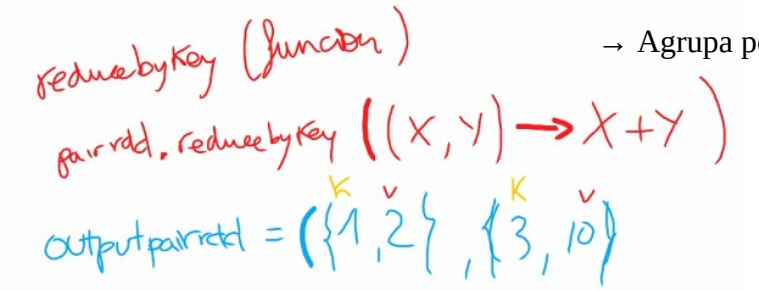


Unpersist() → Elimina los datos de la memoria de RDD

reduce by key. Pair rdd.

Transformaciones RDD. Pág 97 a 106 Apache Spark.

→ Input



→ Agrupa por key. Las que tienen las mismas key, las agrupa.

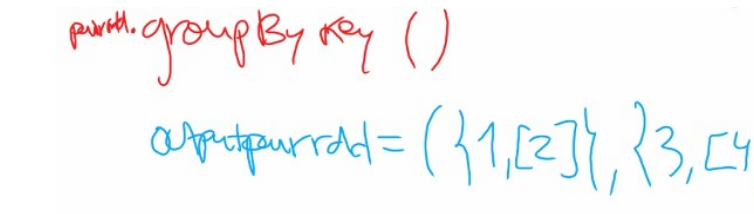
→ Suma todos los valores de las mismas key

→ Output

---

groupbykey

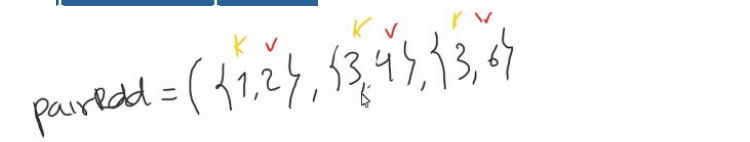
→ Input



→ Output

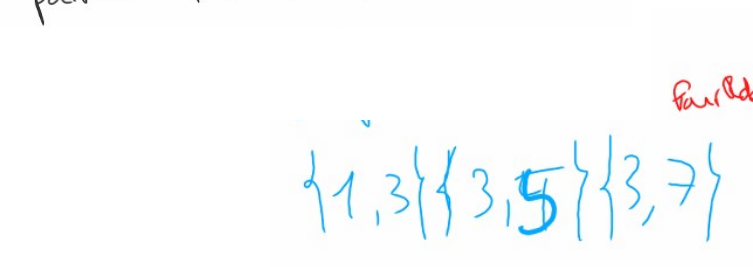
---

pair rdd. map values



→ Input

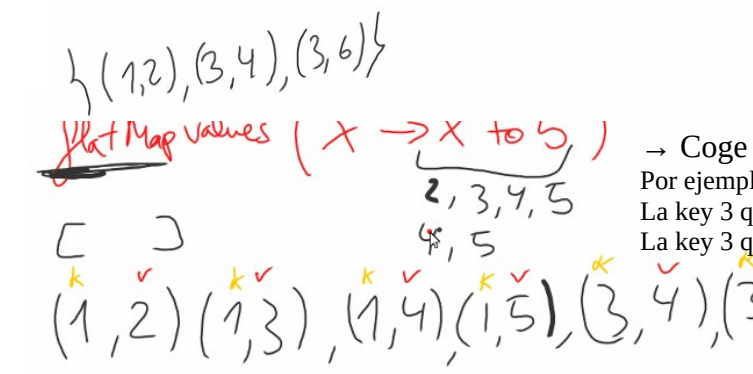
→ Suma 1 a cada valor



→ Output

---

flat map values



→ Input

→ Coge cada valor y lo lleva hasta 5.
Por ejemplo la key 1 que tiene valor 2 lo repite hasta llegar a 5.
La key 3 que tiene valor 4 lo repite hasta llegar a 5
La key 3 que tiene valor 6, lo elimina ya que su valor es menor que 5

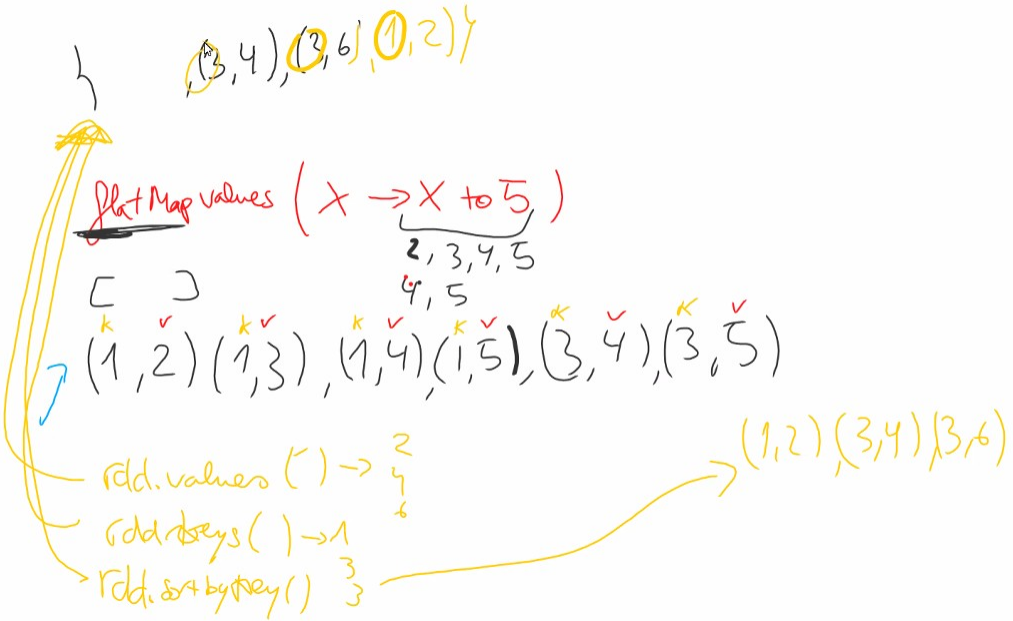
Como es flat no devuelve listas.
Devuelve tuplas.

→ Output

Para acceder a los valores podríamos hacer un map

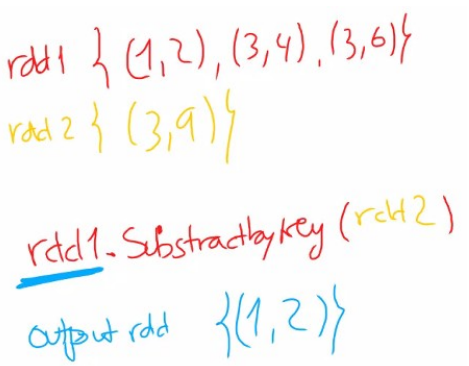
---

rdd.values ; rdd.keys ; rdd.sortbykey



---

rdd.substractbykey

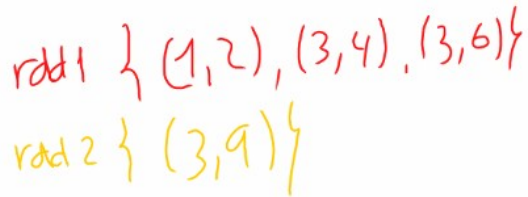


→ De la rdd1 quita los que tengan la misma key que el rdd2

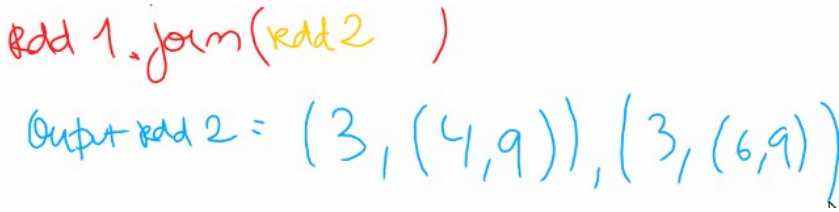
La rdd2 tiene key 3. De rdd1 se quitan todas las de la key3. (independientemente del value)

---

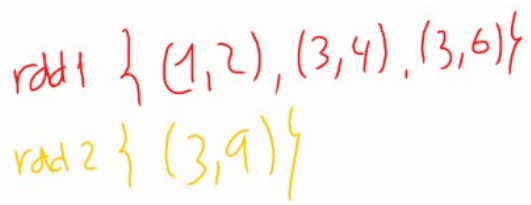
rdd.join → join és un inner join.



De las keys que coinciden, se añade el valor de la 2ª key en las (key, value de la 1ª)

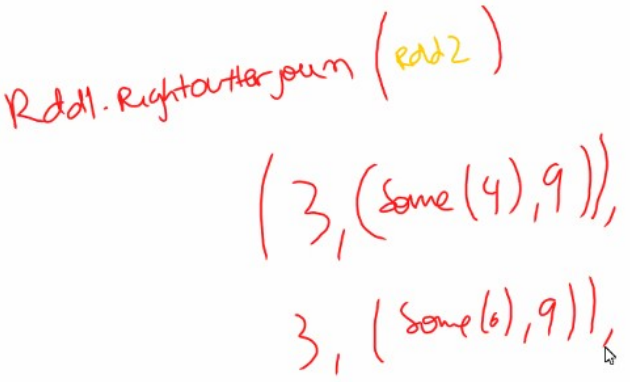


rdd.rightouterjoin



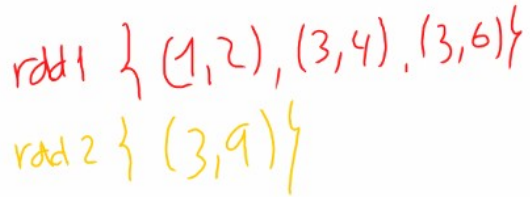
Hay la key 3 con algunos 4 todo de nueves.

Hay la key 3 con algunos 6 y en todos hay 9

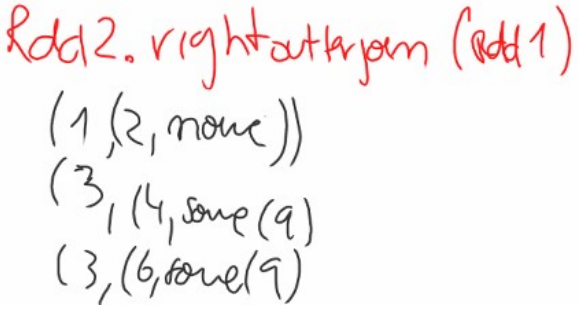


---

rdd.rightouterjoin

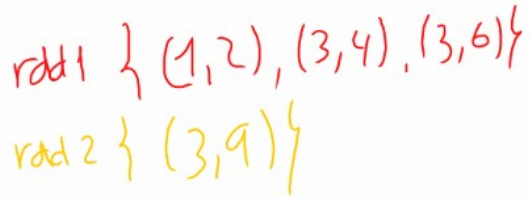


Es lo mismo que hacer rdd1.leftouterjoin(rdd2)



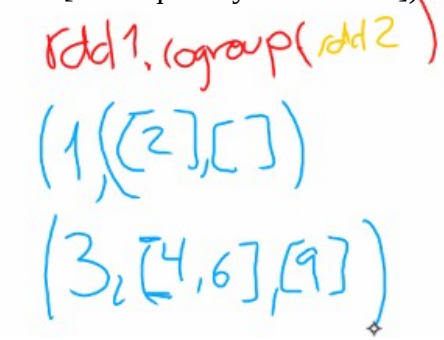
---

rdd.cogroup

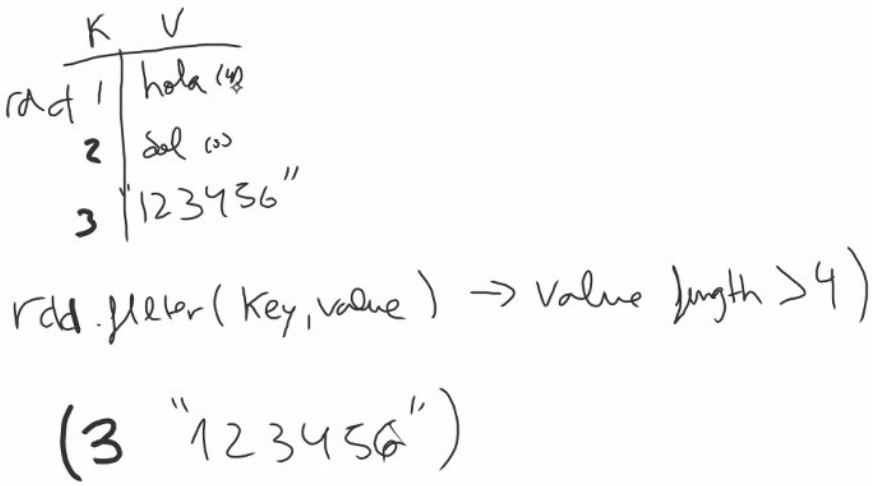


→ En rdd1 que keys hay? De las key que haya escribe los valores de esa key en la rdd1 en una lista tuplada con otra lista de valores de esa key en el rdd2. O sea:

(key, ([valor 1 de esa key en rdd1,[valor 2 de esa key en rdd1], [valor que haya en el rdd2])



rdd.filter lenght

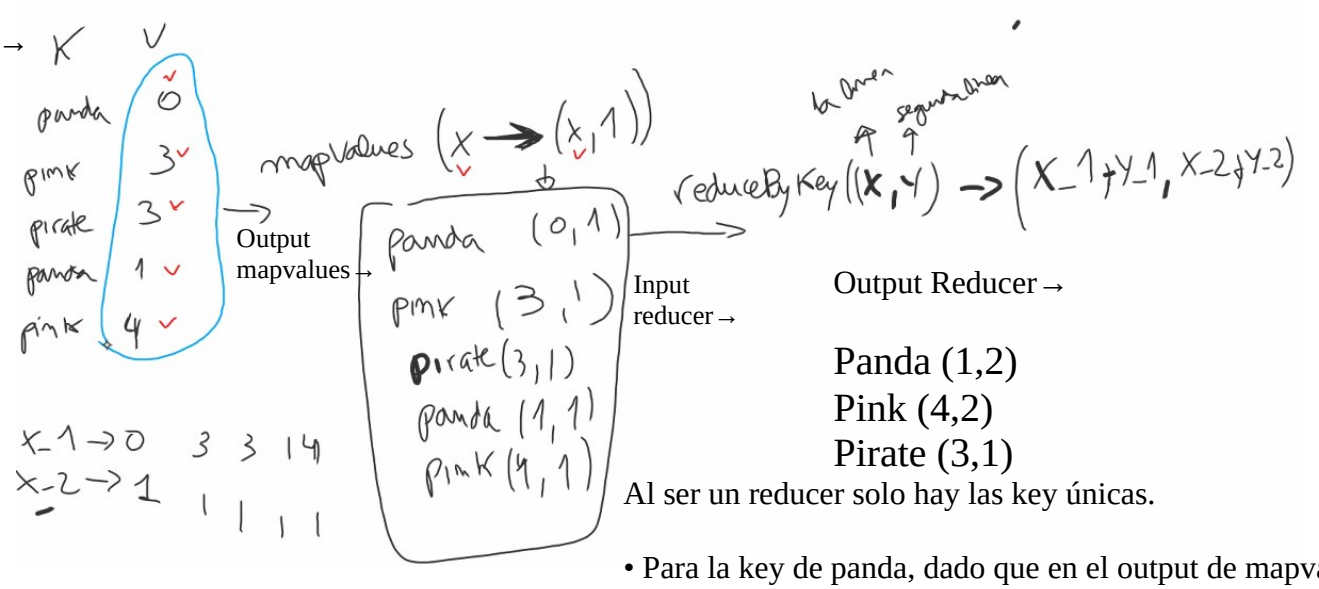


→ Devuelve las key y values en los que el tamaño de caracteres sea mayor a 4

---

rdd.mapvalues rdd.reducebykey

Input →



Output Reducer →

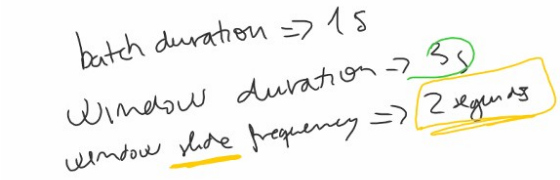
Panda (1,2)
Pink (4,2)
Pirate (3,1)

Al ser un reducer solo hay las key únicas.

• Para la key de panda, dado que en el output de mapvalues hay panda(0,1) y panda(1,1) el resultado del reducer es sumar x_1+y_1, x_2+y_2 o sea: 0+1, 1+1

• Para la key de pink, dado que en el output de mapvalues hay pink(3,1) y pink(4,1) el resultado del reducer es sumar x_1+y_1, x_2+y_2 o sea: 3+1, 1+1

• Para la key de pirate, dado que en el output de mapvalues solo hay pirate(3,1) el resultado del reducer directamente eso: pirate(3,1)



Batch duration = 1 segundo
Window duration = 3 segundos
Window slide frecuency = 2 segundos