APACHE SPARK
Sesión 2

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL
GENERALITAT VALENCIANA
Conselleria d'Educació, Investigació, Cultura i Esport
cefire
UNIÓ EUROPEA
Fons Social Europeu
L'FSE inverteix en el teu futur

¿QUÉ VEREMOS EN ESTA UNIDAD?

1. Repaso sesión 1 y dudas
2. Corrección ejercicios sesión 1
3. RRDs
4. RDDs Clave Valor
5. Repaso MapReduce
6. Persistencia
7. Particionado

Repaso sesión 1

¿Qué es Spark?

¿Cómo funciona Spark?

Preparación entorno práctica Spark

Repaso sesión 1

RDDs en Spark

Qué son Acciones

Qué son Transformaciones

Repaso sesión 1

CORRECCIÓN DE EJERCICIOS

RDD Clave - Valor

RDD ‘Simple’

Tipos básicos: enteros, caracteres, booleanos…

Secuencias: strings, listas, arrays, tuplas, diccionarios

Objetos serializables

Tipos mixtos

RDD Clave - Valor

Distinct()

Union()

First()

Top()

Sampe()

takeSample(()

...

RDD Clave - Valor

- Cada elemento tiene el formato (clave,valor)
- Las claves y los valores pueden ser de cualquier tipo
- Se usan para algoritmos MapReduce
- Hay muchas funciones de procesamiento de datos
  - Orden
  - Agrupación
  - join
  - Count
  - ...

RDD Clave - Valor

KEY	VALUE
A1	Maria
A2	Juan
B1	Jorge
C1	Laura
A1	Lucía

RDD Clave - Valor

¿Cómo generar un RDD de pares?

- Map
- keyBy
- zip
- ...

EJERCICIO PROPUESTO

Ejercicio 1

A partir de la lista:

Perro Gato Loro Pez Leon Tortuga Gallina

a) Crea un RDD a partir de esta lista, o desde un fichero con esta información
b) Convierte el RDD normal en un RDD de pares cuya clave sea la primera letra de animal
c) Crea otro pair rdd asignando a cada elemento un número no secuencial
d) ¿Y un índice incremental?
e) ¿Y un índice incremental que empiece en 100?

RDD Clave - Valor

Agregaciones para pair RDD

ReduceByKey():

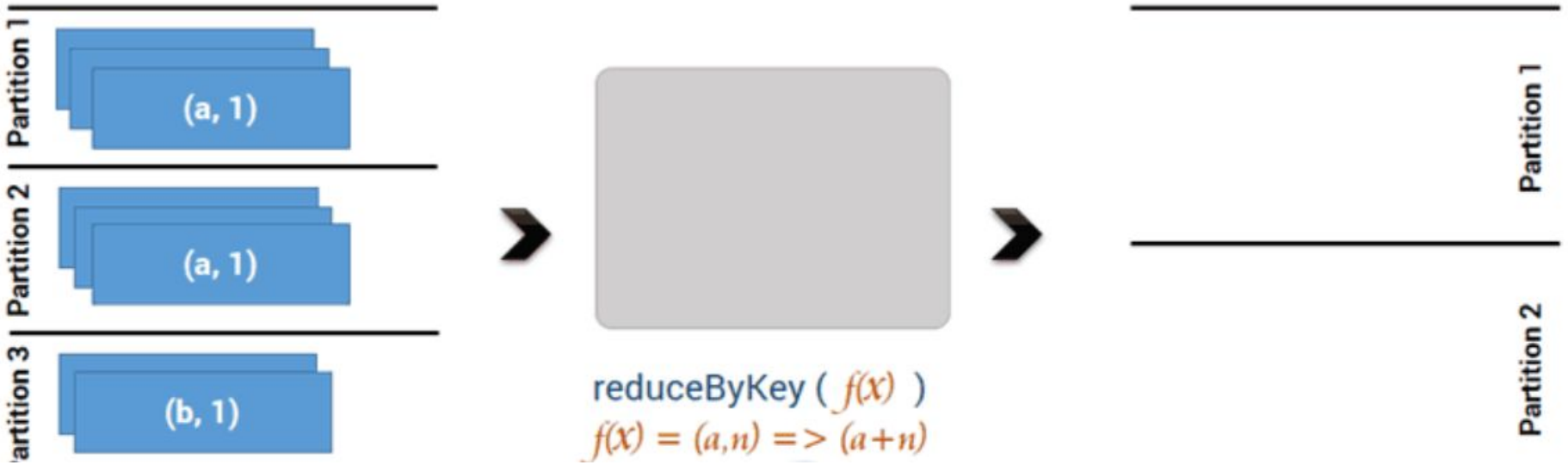
Se usa para fusionar los valores de cada clave usando una función asociativa de reducción. Es una transformación más amplia, ya que baraja datos en múltiples particiones y opera en par RDD

FoldByKey()

Agrega los valores de cada clave en un RDD, usando una función asociativa "func" y un "valor cero" neutral que se puede agregar al resultado un número arbitrario de veces, y no debe cambiar el resultado (por ejemplo, 0 para la suma, o 1 para multiplicar).

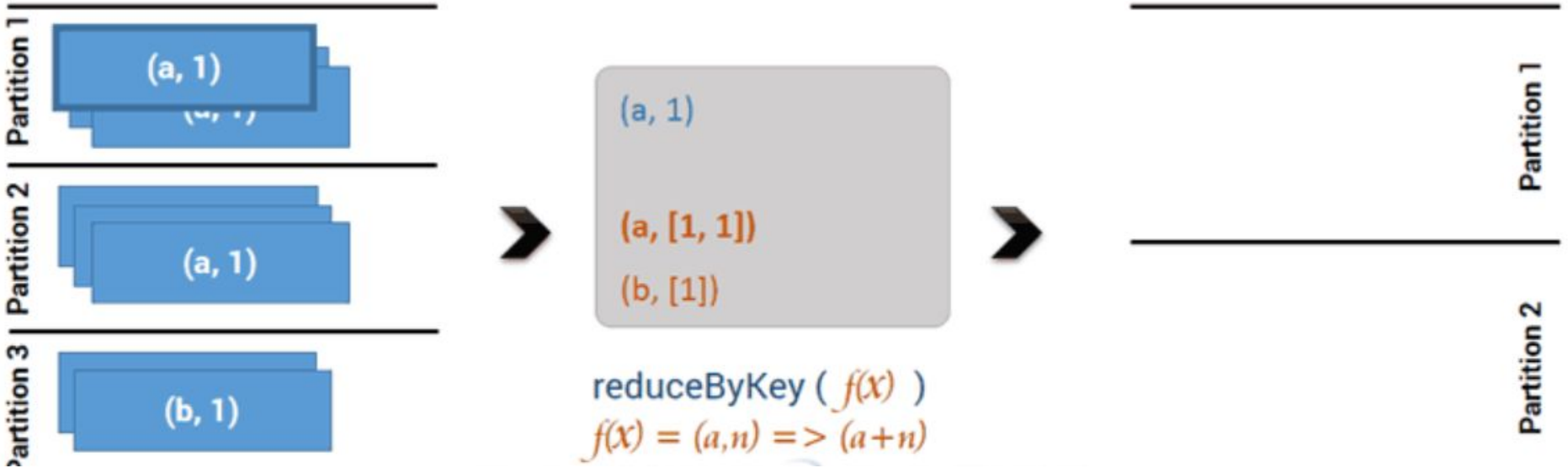
RDD Clave - Valor

REDUCEBYKEY



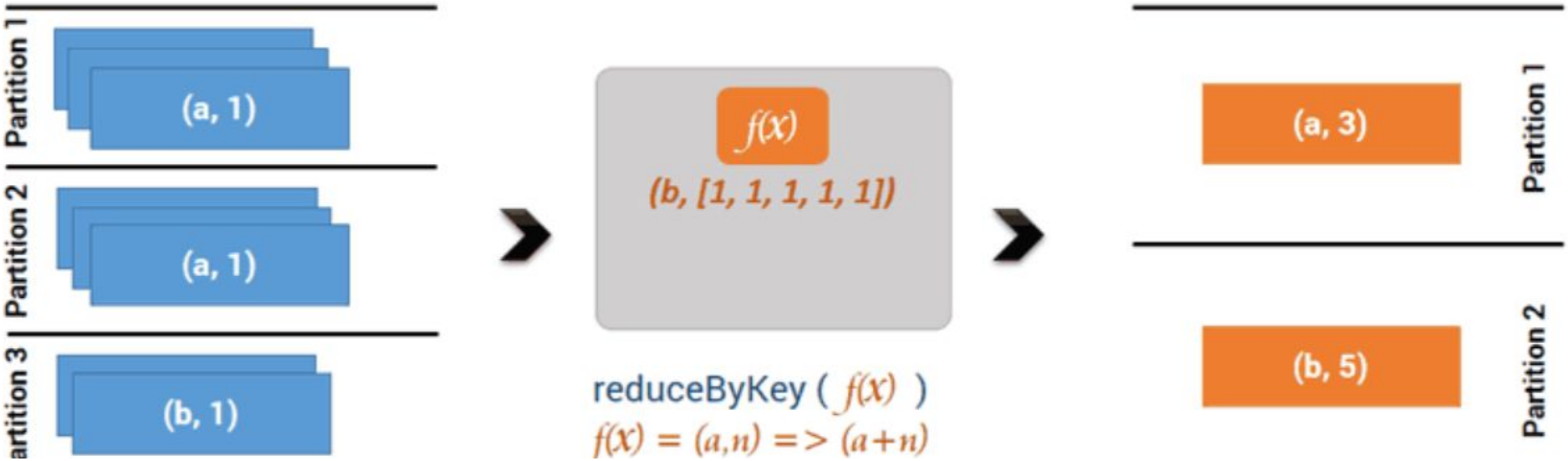
RDD Clave - Valor

REDUCEBYKEY



RDD Clave - Valor

REDUCEBYKEY



EJERCICIO PROPUESTO

Ejercicio 2

Dada una lista de compra calcula:

a) El total que se ha gastado por cada producto

b) Cuánto es lo máximo que ha pagado por cada producto

Lista compra:

Pan 3€	Cereales 3€
Agua 2€	Agua 0.5€
Azúcar 1€	Leche 2€
Leche 2€	Filetes 5€
Pan 1€	

RDD Clave - Valor

Agregaciones para pair RDD

GoupByKey

Agrupa los valores de cada clave en el RDD en una sola secuencia el RDD resultante con particiones numPartitions.

CombineByKey

Es una función genérica para combinar los elementos de cada clave utilizando un conjunto personalizado de funciones de agregación

EJERCICIO PROPUESTO

Tenemos las cuentas de las compras de 3 días:

compra día 1: pan 3€ ,agua 2€, azúcar 1€,leche 2€,pan 4€

compra día 2: pan 1€, cereales 3€, agua 0.5€, leche 2€, filetes 5€

compra día 3: filetes 2€, cereales 1€

¿Cómo haríamos ahora para calcular cuanto ha sido lo máximo que hemos pagado por cada producto?

RDD Clave - Valor

TRANSFORMACIONES para CLAVES o VALORES

- keys(): Nos devuelve las claves
- values(): Nos devuelve los valores
- sortByKey(): Ordena por clave
- mapValues(): Aplica la función sobre los valores
- fatMapValues() Aplica la función sobre los valores y crea una lista simplificada
- SubtrackByKey() Dados dos RDD elimina de uno los que la clave SI esta en el otro
- Cogroup() Dados dos RDDs agrupa por claves y devuleve un iterable como valor

RDD Clave - Valor

JOINS

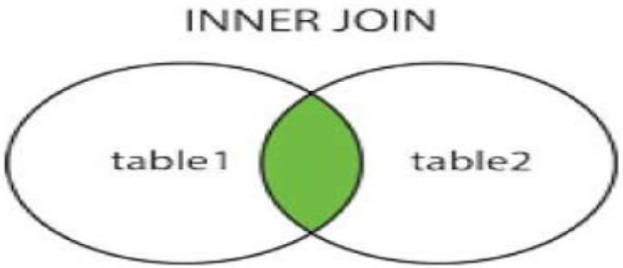
join() : Es equivalente a un inner join

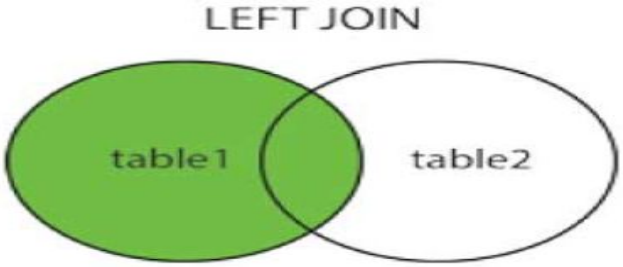
leftOuterJoin(): Equivalente a un Left join

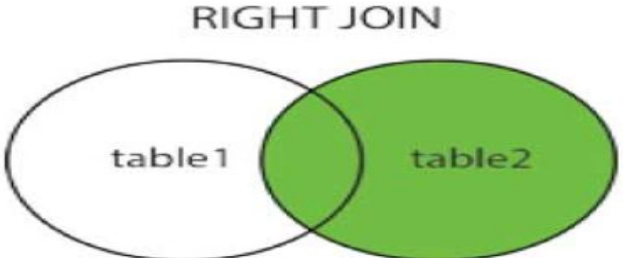
rightOuterJoin: Equivalente a un Right join

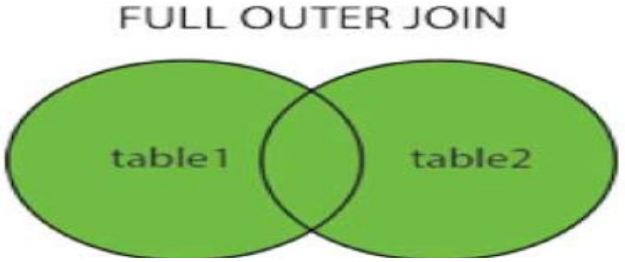
fullOuterJoin: Equivalente a full join

RDD Clave - Valor









EJERCICIO PROPUESTO

Somos dueños de dos bibilioteca y tenemos los libros repartidos de este modo:

Sala	Libro	Ejemplares
Sala A	La Comunidad del Anillo	3
Sala A	Las dos torres	2
Sala A	El retorno del Rey	4
Sala A	El hobbit	8

Sala	Libro	Ejemplares
Sala B	Prólogo	3
Sala B	La Comunidad del Anillo	1
Sala B	Las dos torres	5
Sala B	El retorno del Rey	2

EJERCICIO PROPUESTO

Se solicita:

a) Resumen de libros y cantidad por sala,

Por ejemplo Libro_A (3,2)

b) Los libros que están disponibles en ambas salas

c) De los libros que tenemos en las dos salas, cuántos tendríamos en total de cada uno de ellos entre las 2 salas

d) Saca una lista de los libros que tenemos, nos da igual el numero de ejemplares

e) Total de ejemplares ordenado ascendentemente

RDD Clave - Valor

Otras transformacion y acciones

SubstractByKey()

cogroup()

collectAsMap()

countByKey()

lookup()

EJERCICIO PROPUESTO

Dado el mismo enunciado que antes

Sala	Libro	Ejemplares
Sala A	La Comunidad del Anillo	3
Sala A	Las dos torres	2
Sala A	El retorno del Rey	4
Sala A	El hobbit	8

Sala	Libro	Ejemplares
Sala B	Prólogo	3
Sala B	La Comunidad del Anillo	1
Sala B	Las dos torres	5
Sala B	El retorno del Rey	2

EJERCICIO PROPUESTO

Se solicita

a) ¿Cómo podríamos saber el total de ejemplares de 'El hobbit' de la sala_A?

b) ¿Qué libros no están en la sala A pero si en la B?

:

RDD Clave - Valor

RDD con formato CLAVE - VALOR

Transformaciones

Acciones

Joins

EJERCICIO PROPUESTO

Supuesto práctico

a)Carga los logs de la carpeta weblogs

b)Crea un RDD de pares a partir de estos ficheros con el usuario como clave y un valor 1.

userid1,1

userid2,1

userid3,1

c) Suma los valores de cada ID de usuario

useid1,4

userid2,9

d) Muestra los 10 usuarios con máximas apariciones

EJERCICIO PROPUESTO

Crea un rdd de pares donde la clave sea el usuario y el valor una lista de Ips desde donde se ha conectado el usuario

ej:

usuario1,[ip1,ip2,ip3]

usuario2,[ip4,ip5,ip6]

En cvs de cuentas, tenemos información de los clientes, podrías añadir el nombre y apellido del usuario junto a las apariciones de ese usuario?

por ejemplo: Usuario1,(Nombre apellido,30)

Práctica Repaso

Tenemos las notas de Mates, Inglés y Física de los alumnos del colegio en 3 documentos txt, a partir de estos ficheros:

a) Crea 3 RDD de pares, uno para cada asignatura, con los alumnos y sus notas

b) Crea un solo RDD con todas las notas

c) ¿Cuál es la nota más baja que ha tenido cada alumno?

Práctica Repaso

d) ¿Cuál es la nota media de cada alumno?

e) ¿En qué asignatura suspende más gente?

f) Total de notables o sobresalientes por alumno

g) ¿Qué alumno no se ha presentado a inglés?

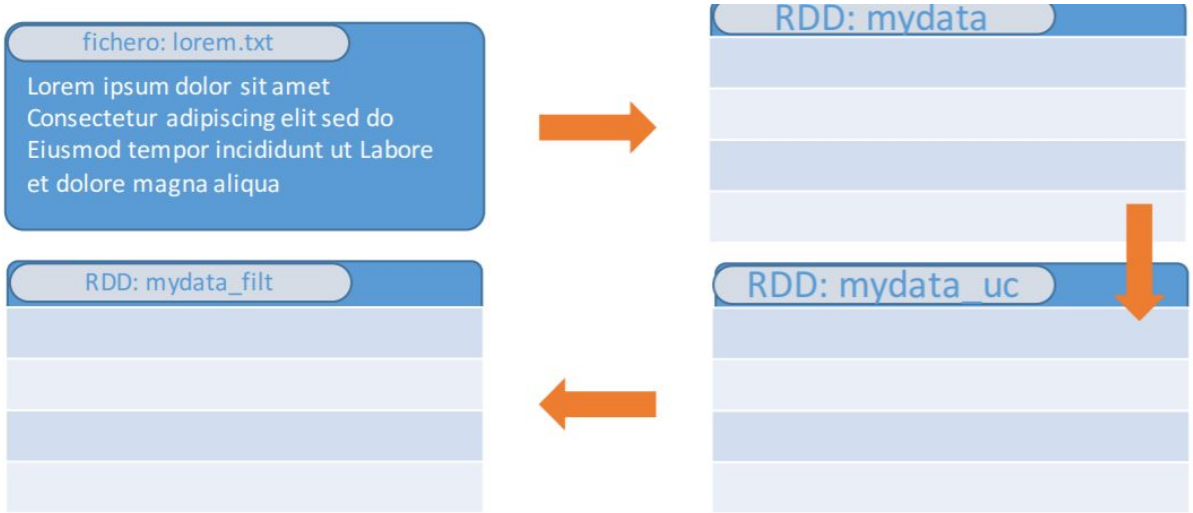
h) ¿A cuántas asignaturas se ha presentado cada alumno?

i) Obten un RDD con cada alumno con sus notas

PERSISTENCIA

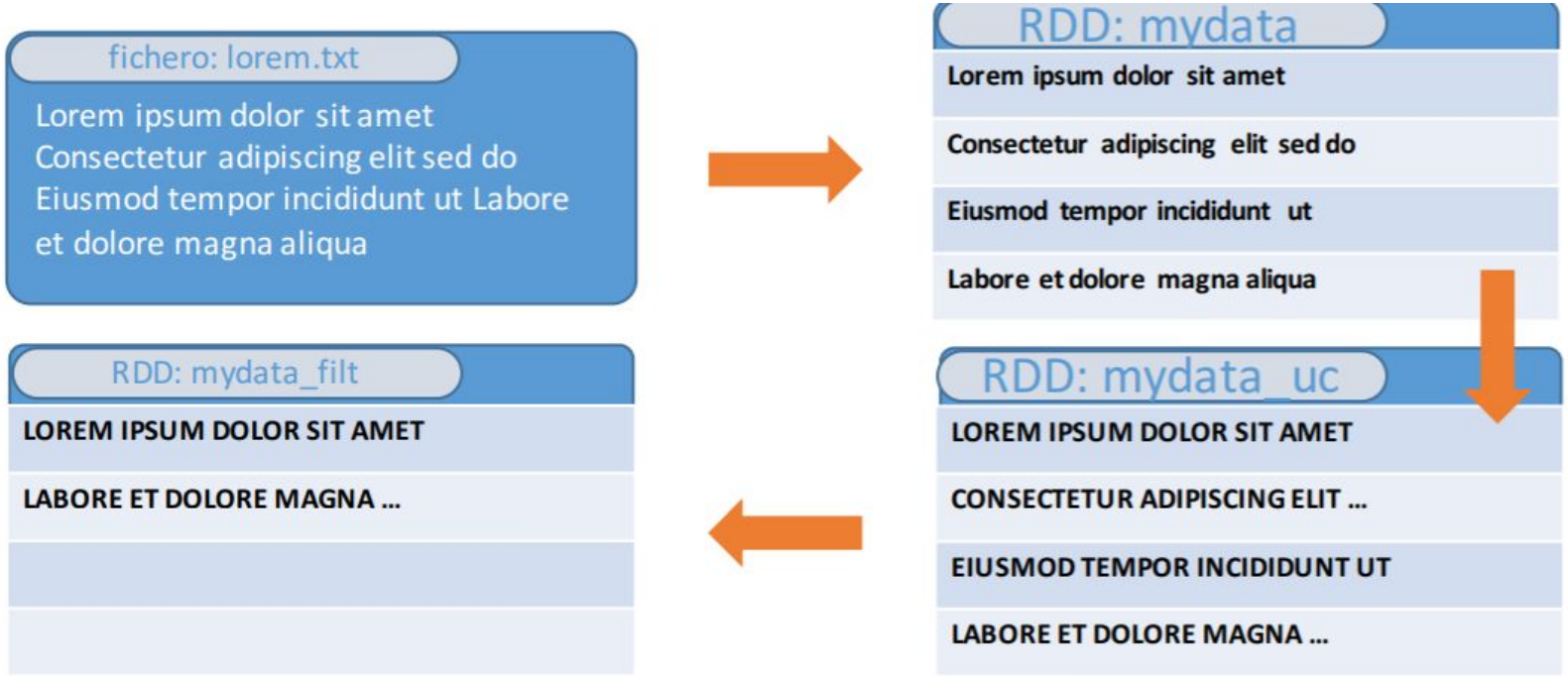
- Persistencia: Capacidad de guardar la información
- En spark los datos RDD no se procesan hasta que se realice una acción

mydata = sc.textFile("lorem.txt")



PERSISTENCIA

Aplicamos una función filter para quedarnos con las frases que empiecen por L



PERSISTENCIA

Ventajas del almacenamiento en caché y la persistencia

Rentable : los cálculos de Spark son muy costosos, por lo que la reutilización de los cálculos se utiliza para ahorrar costes.

Ahorro de tiempo : reutilizar los cálculos repetidos ahorra mucho tiempo.

Tiempo de ejecución : ahorra tiempo de ejecución del trabajo y podemos realizar más trabajos en el mismo clúster.

PERSISTENCIA

Persistencia.

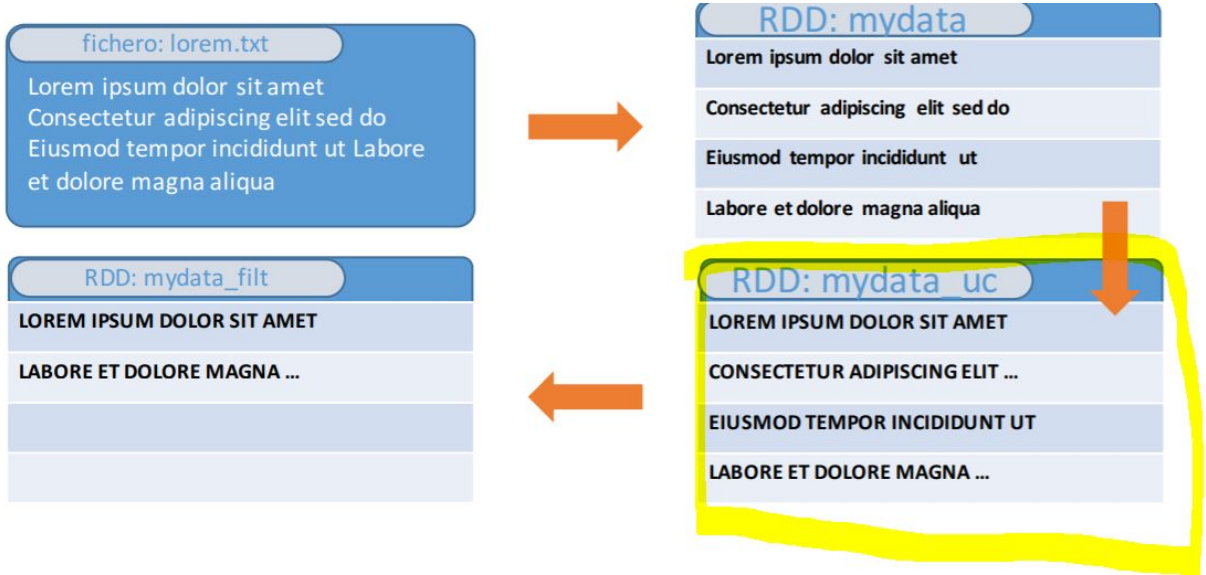
Conseguir que la información persista y no tener que recalcular cada vez que queramos hacer uso de ella.

Métodos:

- Cache()
- Persist()

PERSISTENCIA

Cache() es una “sugerencia” a Spark

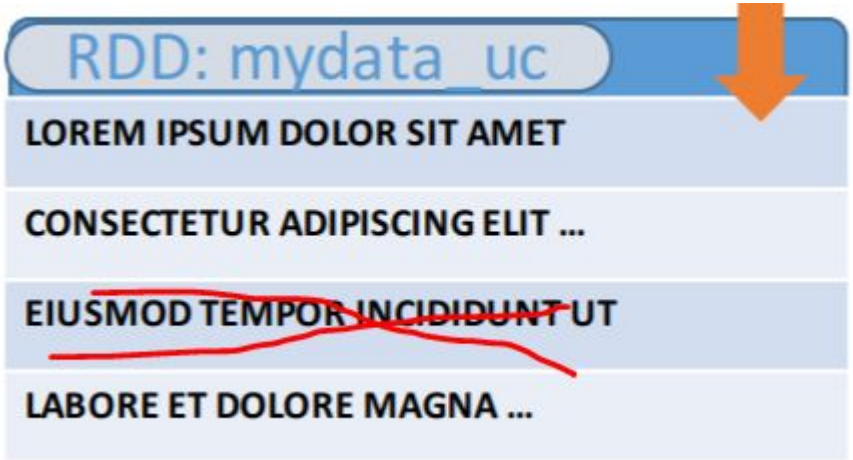


PERSISTENCIA

-Cache() conforma una cache distribuida

-Cada nodo guarda en memoria lo calculado

-¿Fallo en una partición cache()?



PERSISTENCIA

PERSIST()

MEMORY_ONLY - Este es el comportamiento predeterminado del cache() método RDD y almacena el RDD o DataFrame como objetos deserializados en la memoria JVM. Cuando no hay suficiente memoria disponible, no se guardarán los DataFrame de algunas particiones y se volverán a calcular cuando sea necesario.

PERSISTENCIA

MEMORY_AND_DISK - Este es el comportamiento predeterminado del DataFrame o Dataset. En este nivel de almacenamiento, el DataFrame se almacenará en la memoria JVM como objetos deserializados. Cuando el almacenamiento requerido es mayor que la memoria disponible, almacena algunas de las particiones sobrantes en el disco y lee los datos del disco cuando es necesario.

PERSISTENCIA

MEMORY_ONLY

MEMORY_ONLY_SER

MEMORY_ONLY_2

MEMORY_ONLY_SER_2

MEMORY_AND_DISK

MEMORY_AND_DISK_SER

MEMORY_AND_DISK_2

MEMORY_AND_DISK_SER_2

DISK_ONLY

DISK_ONLY_2

PERSISTENCIA

Según el nivel de almacenamiento:

StorageLevel.**DISK_ONLY** = StorageLevel(**True**, **False**, **False**, **False**)
StorageLevel.**DISK_ONLY_2** = StorageLevel(**True**, **False**, **False**, **False**, 2)
StorageLevel.**MEMORY_ONLY** = StorageLevel(**False**, **True**, **False**, **False**)
StorageLevel.**MEMORY_ONLY_2** = StorageLevel(**False**, **True**, **False**, **False**, 2)
StorageLevel.**MEMORY_AND_DISK** = StorageLevel(**True**, **True**, **False**, **False**)
StorageLevel.**MEMORY_AND_DISK_2** = StorageLevel(**True**, **True**, **False**, **False**, 2)
StorageLevel.**OFF_HEAP** = StorageLevel(**True**, **True**, **True**, **False**, 1)

PERSISTENCIA

UNPERSIST(): Elimina la información almacenada

Es necesario para cambiar el nivel de persistencia hacer un unpersist previo

PERSISTENCIA

Error de Persistencia en disco

-Almacenamiento **en fichero** en disco

-Persistencia **sin** replicación

-Persistencia **con** replicación

PERSISTENCIA

¿Cuando cachear?

Cuando se va a reutilizar varias veces para cálculos iterativos, por ejemplo en machine learning

¿Qué nivel usar?

Memoria: Mejor rendimiento, pero la memoria es limitada

Disco: Más costoso leer de disco pero mas capacidad

Replicación: Cuando es más costoso recalcular que replicar

EJERCICIO PROPUESTO

Vamos a leer un fichero más o menos, por ejemplo el del quijote que teníamos de la sesión anterior y hacerle alguna operación(en el caso de no ser suficiente, podemos ampliarlo o elegir un fichero más grande), vamos a contar las palabras que son mayores que 10

Esto mismo lo haremos con persistencia y sin persistencia para comprobar el tiempo que tarda cada una

PERSISTENCIA

CHECKPOINT()

- Guardamos en el tiempo la información para poder reutilizarla
- Nos sirve para evitar problemas de recálculos en caídas

PERSISTENCIA

Ejemplo cuando usar persist o checkpoint

- Cálculos que una vez calculados ya no vayamos a calcular pero los necesitamos (CHECKPOINT)

- Cálculos que hacemos y ya no vamos a necesitar nunca más (PERSIST)

PERSISTENCIA

LINAJE de los datos

Los datos originales van cambiando a lo largo del camino para pasar a ser algo completamente diferente cuando llegan al final. Y muy pocos entienden cómo llegaron a convertirse en algo tan distinto a la versión original.

PERSISTENCIA

LINAJE de los datos

El linaje hace posible la tolerancia a fallos

PERSISTENCIA

- Realmente no almacenamos los datos en sí.
- Almacenamos todo el linaje

EJERCICIO PROPUESTO

Vamos a crear un RDD simple por ejemplo:

rdd = 1,2,3,4,5

A este RDD vamos a iterar y cada iteración vamos a sumar 1 a cada elemento del RDD.

Vamos a ir subiendo el número de iteraciones hasta que la memoria ya no de más de sí y nos de error.

Para solucionar este error usaremos Check Points

Acumuladores

- Son variables globales
- Sirven para tener una anlitica de lo que hemos hecho
- Tipo long y double

EJERCICIO PROPUESTO

Dados los ficheros de log de la carpeta weblogs, queremos contar cuantas veces se han enviado datos en formato png,jpg,html,css y txt.

Usa acumuladores para recorrer los ficheros y calcular cuantos hay de cada formato en total entre todos los logs

Variable de transmisión

Variables BROADCAST:

Permiten al programador mantener una variable de solo lectura almacenada en caché en cada máquina en lugar de enviar una copia de ella con las tareas

Variable de transmisión

¿COMO FUNCIONAN?

- PySpark divide el trabajo en etapas que han distribuido la mezcla y las acciones se ejecutan en la etapa.
- Las etapas posteriores también se dividen en tareas.
- Spark transmite los datos comunes (reutilizables) necesarios para las tareas dentro de cada etapa.
- Los datos transmitidos se almacenan en caché en formato serializado y deserializados antes de ejecutar cada tarea.

EJERCICIO PROPUESTO

EJERCICIO PROPUESTO

Queremos filtrar las peticiones web para solo ver las que aparezcan en una lista de modelos que tenemos (fichero: targetmodels.txt)

Necesitamos filtrar los logs del servidor para solo ver las peticiones de esta lista de modelos, como son muchos ficheros spark los va a repartir por muchos nodos y nos interesa crear una variable con estos datos en cada nodo.

Los logs estarán en una carpeta llamada weblogs de nuestro hdfs (o en local)

(Nota: Repasar funciones any y strip de python)

PARTICIONADO

- Todos los nodos de un clúster de Spark contienen una o varias particiones.
- El número de particiones que utiliza Spark es configurable y no es aconsejable tener demasiadas particiones
- Las particiones en Spark no abarcan más de una máquina.
- Se garantiza que las tuplas de una misma partición estarán todas en la misma máquina.
- Spark asigna una tarea por partición y cada trabajador puede procesar una tarea cada vez.

PARTICIONADO

- **GetNumParticions()**

Devuelve el número de particiones

- **Glom()**

Visualización de los datos con sus particiones

Funciones de reparticionado:

- ReduceByKey()
- repartirion(n)
- partitionBy()
- coalesce(n)

APACHE SPARK
Sesión 2

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL
GENERALITAT VALENCIANA
Conselleria d'Educació, Investigació, Cultura i Esport
cefire
UNIÓ EUROPEA
Fons Social Europeu
L'FSE inverteix en el teu futur