Taller Big Data - Parte 1

Carlos Eiras Franco

Department of Computer Science
University of A Coruña (Spain)

"Data Science es el arte de transformar los datos en acciones"

The field guide to Data Science - Booz Allen Hamilton

Las herramientas para Big Data

Procesamiento distribuido y origen de Spark

LIDIA Laboratory for R&D in Artificial Intelligence

Supermáquinas

El software tradicional para analizar datos se ejecuta en una sola máquina.

https://en.wikipedia.org/wiki/Supercomputer

Más datos $\rightarrow$ Aumentar la potencia de la máquina $\rightarrow$ Caro e insuficiente.

LIDIA Laboratory for R&D in Artificial Intelligence

Procesamiento distribuido

Agrupaciones de ordenadores de consumo doméstico.

https://en.wikipedia.org/wiki/Computer_cluster

Traslada la complejidad al software.

Complicaciones del procesamiento distribuido

- HW de consumo falla **mucho** $\rightarrow$ Tolerancia a fallos
- No todos los ordenadores procesan igual de rápido $\rightarrow$ Sincronización
- Comunicación por red lenta $\rightarrow$ Evitar envío de información

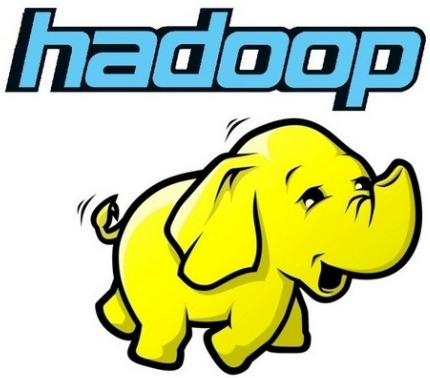
MapReduce

En 2004 Google presentó el paradigma MapReduce.

1. Dividir el trabajo en tareas independientes
2. Procesar cada tarea independientemente (Map)
3. Juntar los resultados parciales (Reduce)

No importa el orden ni la velocidad de ejecución.

Implementado en Hadoop



Inconveniente

Hadoop almacena los datos entre pasos MapReduce.

Spark intenta mantener los datos en memoria el mayor tiempo posible.

RDDs

Los Resilient Distributed Datasets son el núcleo de Spark, sobre el que se realizan todas las operaciones.

Resilient $\rightarrow$ Spark se ocupa de reconstruirlos si se han tenido que borrar.

Sobre este objeto se realizan operaciones y Spark se ocupa de que se lleven a cabo en el entorno distribuido.

Ciclo de vida de los RDDs

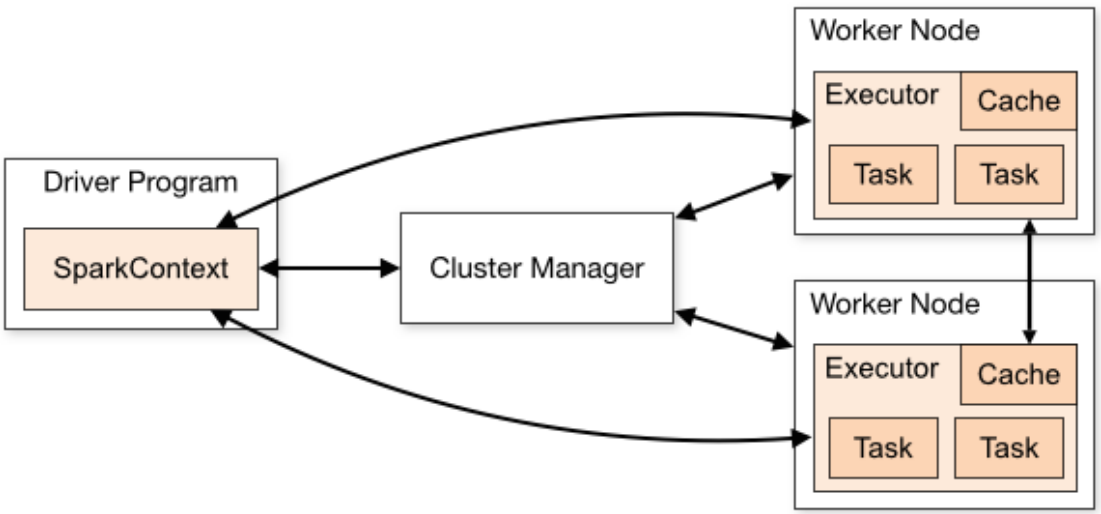
1. **Creación**: A partir de una variable, fichero u otra fuente de datos.
2. **Transformaciones**: Operaciones map
3. **Acciones**: Operaciones reduce
4. **Almacenamiento**: Si se requiere salvar el dataset transformado.

Organización

Las máquinas del cluster pueden tener dos roles:

- El **driver** distribuye y coordina las tareas.
- Los **workers** ejecutan el trabajo.

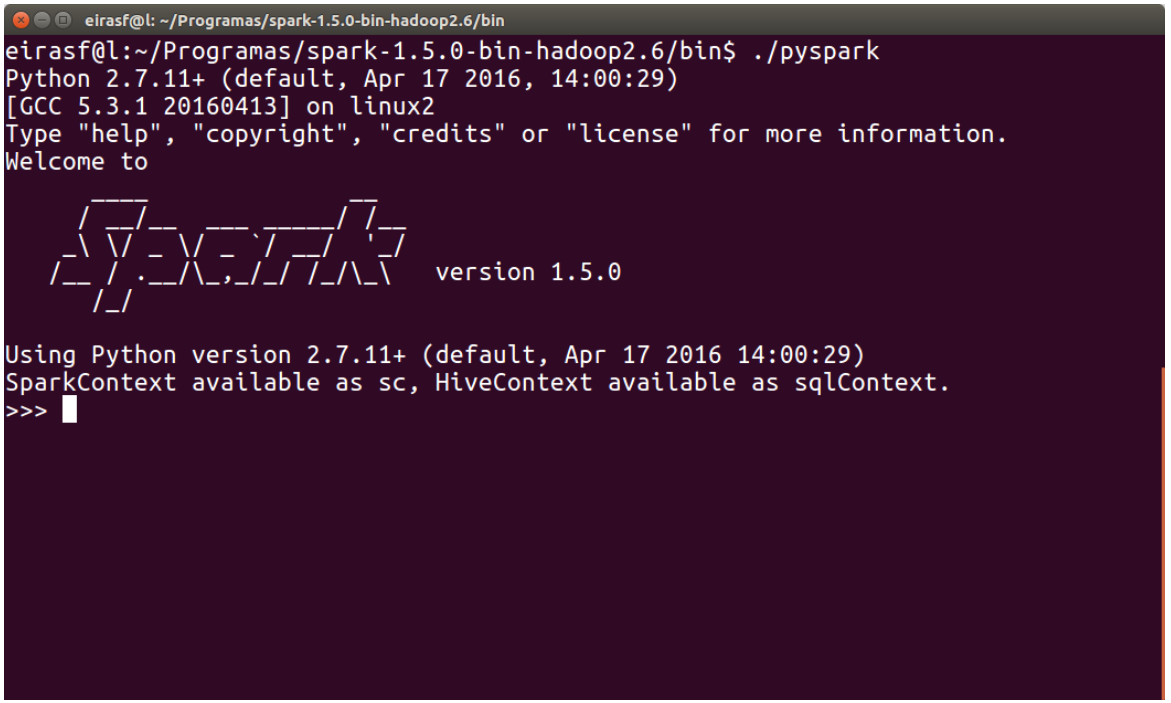
Entre ambos puede haber un gestor del clúster (p.e. Yarn o Mesos) o no (modo standalone).



Tutorial Spark - Requisitos

- Python
- Spark (archivo comprimido)
- Zip de contenidos del curso (`http://tinyurl.com/EVIA2016-TallerBD`) tutorial.html

Lanzar pySpark



Consola python con variables precargadas.

Ejercicio 1

Trampas comunes

Puntos habituales de pérdida de eficiencia

LIDIA
Laboratory for R&D in Artificial Intelligence

1. Pasos map mal optimizados

La función map se envía (con todas las variables de su closure) a cada partición y se ejecuta una vez por cada elemento.

<CODE >

```
>incremento=5 #Variable local
>rddEnteros.map(lambda x: x+incremento) #Suma 5 a cada elemento
```

LIDIA
Laboratory for R&D in Artificial Intelligence

1. Pasos map mal optimizados

La función map se envía (con todas las variables de su closure) a cada partición y se ejecuta una vez por cada elemento.

<CODE >

```
>incremento=5 #Variable local
>rddEnteros.map(lambda x: x+incremento) #Suma 5 a cada elemento
```

Es importante:

- Escribir funciones eficientes

1. Pasos map mal optimizados

La función map se envía (con todas las variables de su closure) a cada partición y se ejecuta una vez por cada elemento.

<CODE >

```
>incremento=5 #Variable local
>rddEnteros.map(lambda x: x+incremento) #Suma 5 a cada elemento
```

Es importante:

- Escribir funciones eficientes
- Evitar variables pesadas en el closure.
  Si es muy grande $\rightarrow$ Broadcast

LIDIA Laboratory for R&D in Artificial Intelligence

1. Pasos map mal optimizados

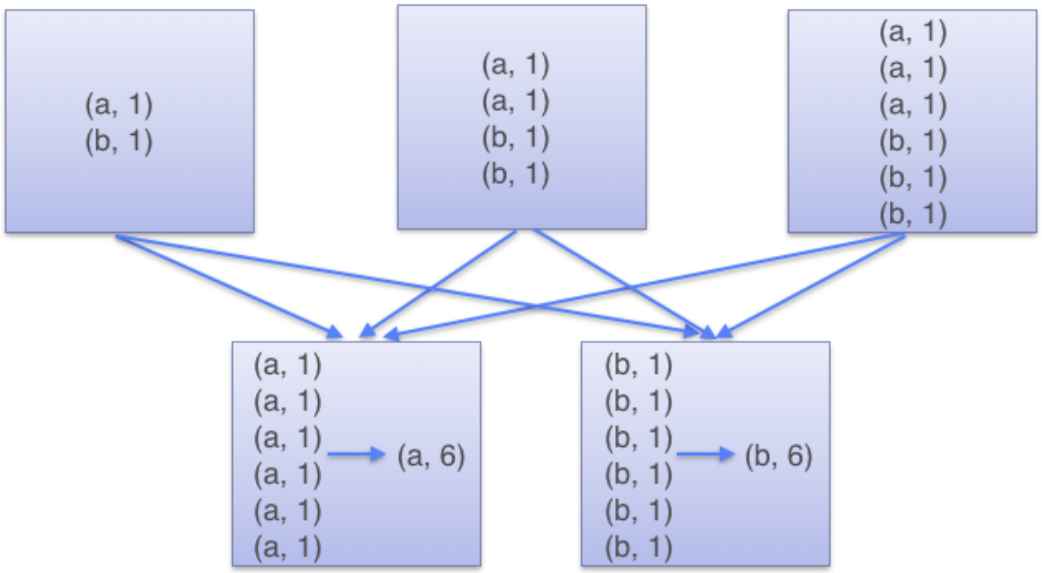
<CODE >

```
>variable_grande=list(xrange(10000000))
>broadcastVG=sc.broadcast(variable_grande)
>rddEnteros.map(lambda x: x+broadcastVG.value[5]) #Suma 5 a cada elemento (el quinto elemento del array)
```

2. GroupByKey vs ReduceByKey

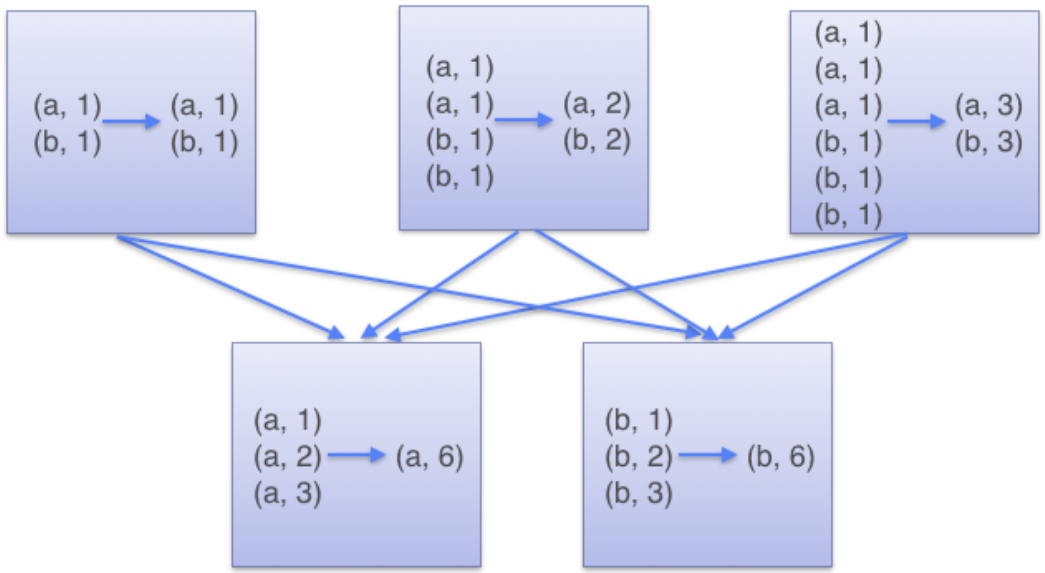
No utilizar GroupByKey cuando se pueda utilizar ReduceByKey, porque requiere mover todos los datos.

2. GroupByKey vs ReduceByKey



GroupByKey primero agrupa y luego requiere un paso de reducción

2. GroupByKey vs **ReduceByKey**



ReduceByKey solo requiere transmitir un valor por clave desde cada nodo

LIDIA Laboratory for R&D in Artificial Intelligence

3. Particionado inadecuado

El particionado determina cómo se reparten los datos entre los nodos.

Carga equilibrada $\rightarrow$ Mayor paralelismo

<CODE >

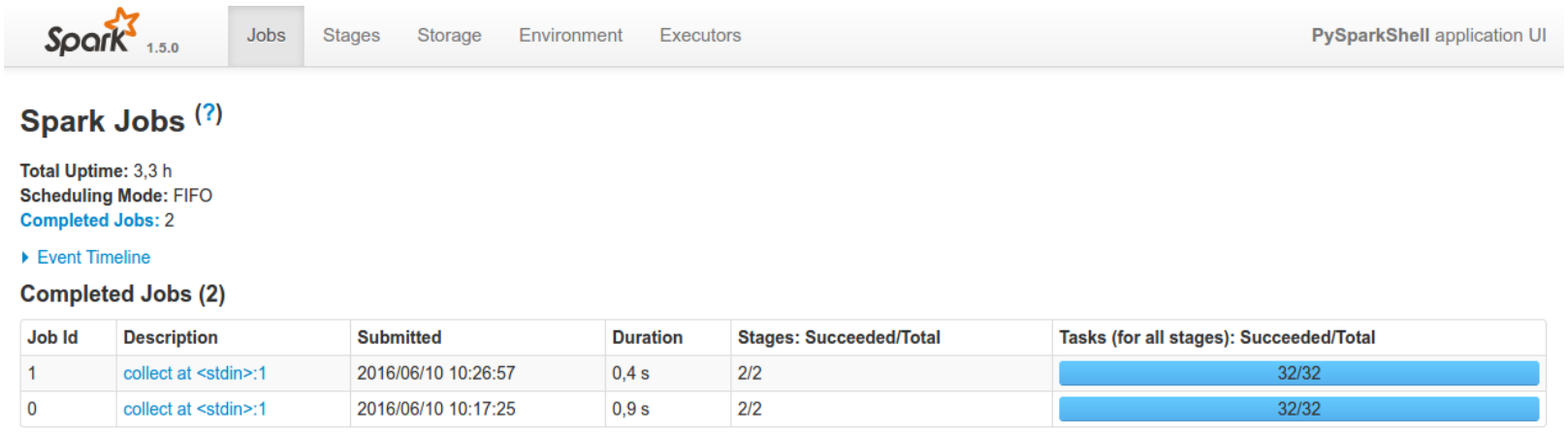
>cualquierRDD.**repartition(número_particiones)** #Reparte de nuevo todos los datos

>cualquierRDD.**coalesce(número_particiones)** #Decrementa el número de particiones (evita trasiego de datos)

>cualquierRDD.**getNumPartitions()** #Indica de cuántas particiones se compone

Interfaz web

Permite comprobar el estado de los trabajos mientras se ejecutan.



http://localhost:4040 - Mientras se está ejecutando

Lanzar aplicaciones

spark-submit <nombre_aplicación>

Deberemos definir el SparkContext en nuestra aplicación

<CODE >

```
from pyspark import SparkContext, SparkConf
conf = SparkConf()
        .setAppName(”ejercicio2”)
        .setMaster(”local[8]”)#Se puede definir con un parámetro de
spark-submit
sc = SparkContext(conf=conf)
```