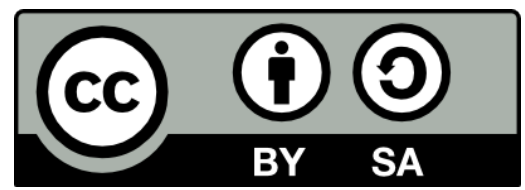
Big Data I:

Ingeniería de datos

Felipe Ortega
Dpto. de Estadística e Investigación Operativa
Universidad Rey Juan Carlos

April 7, 2015

Universidad
Rey Juan Carlos



Procesamiento offline vs. real-time

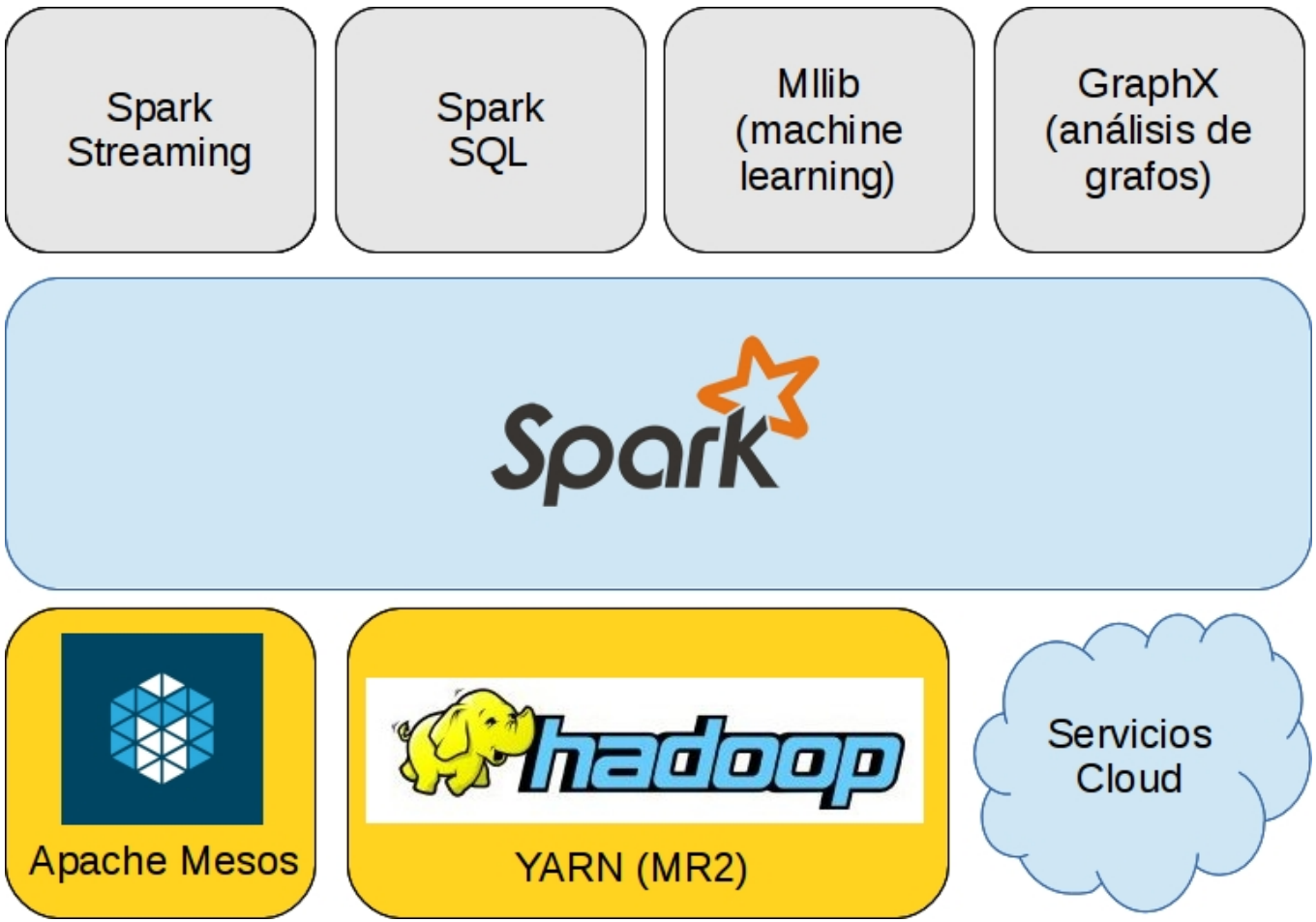
- Un gran problema de Hadoop es que, a pesar de distribuir tareas y datos entre muchos nodos, puede tardar mucho.
- Necesidad de sistemas que puedan realizar consultas a gran velocidad (**interactivas**) sobre grandes volúmenes de datos.
- Ejemplos
  - Apache Spark.
  - Presto (Facebook).

Apache Spark

- Framework para análisis de datos *veloz*.
- Puede utilizar HDFS, pero no está ligado al diseño en dos fases característico de MapReduce.
- Soporte para grafos de operaciones arbitrarios, computación en memoria (cuidado con requisitos del sistema).
- APIS: Scala, Java y Python, caché de datos en memoria, interfaces para exploración interactiva de datos.
- Sobre el framework de análisis de flujo de datos se colocan módulos que ofrecen funcionalidades específicas.

Stack de herramientas en Spark

- `https://amplab.cs.berkeley.edu/software/.`



Installing Spark

- `http://spark.apache.org/downloads.html`.
- `Pre-built for Hadoop 2.4 and later`
- Viene con 3 entornos de programación a elegir:
  - Shell Python (`./bin/pyspark`).
  - Shell Scala (`./bin/spark-shell`).
  - Java API (standalone app + Maven u otro builder).
- Ejemplo lanzamiento shell IPython:
  - `PYSPARK_DRIVER_PYTHON=ipython ./bin/pyspark`.
  - `PYSPARK_DRIVER_PYTHON=ipython PYSPARK_DRIVER_PYTHON_OPTS="notebook --pylab inline" ./bin/pyspark`

Spark en Acción (I)

- Contado de palabras en Spark [3] con Python.

```
file = spark.textFile("hdfs://my_doc_file.txt")

file.flatMap(lambda line: line.split())
    .map(lambda word: (word, 1))
    .reduceByKey(lambda a, b: a+b)
```

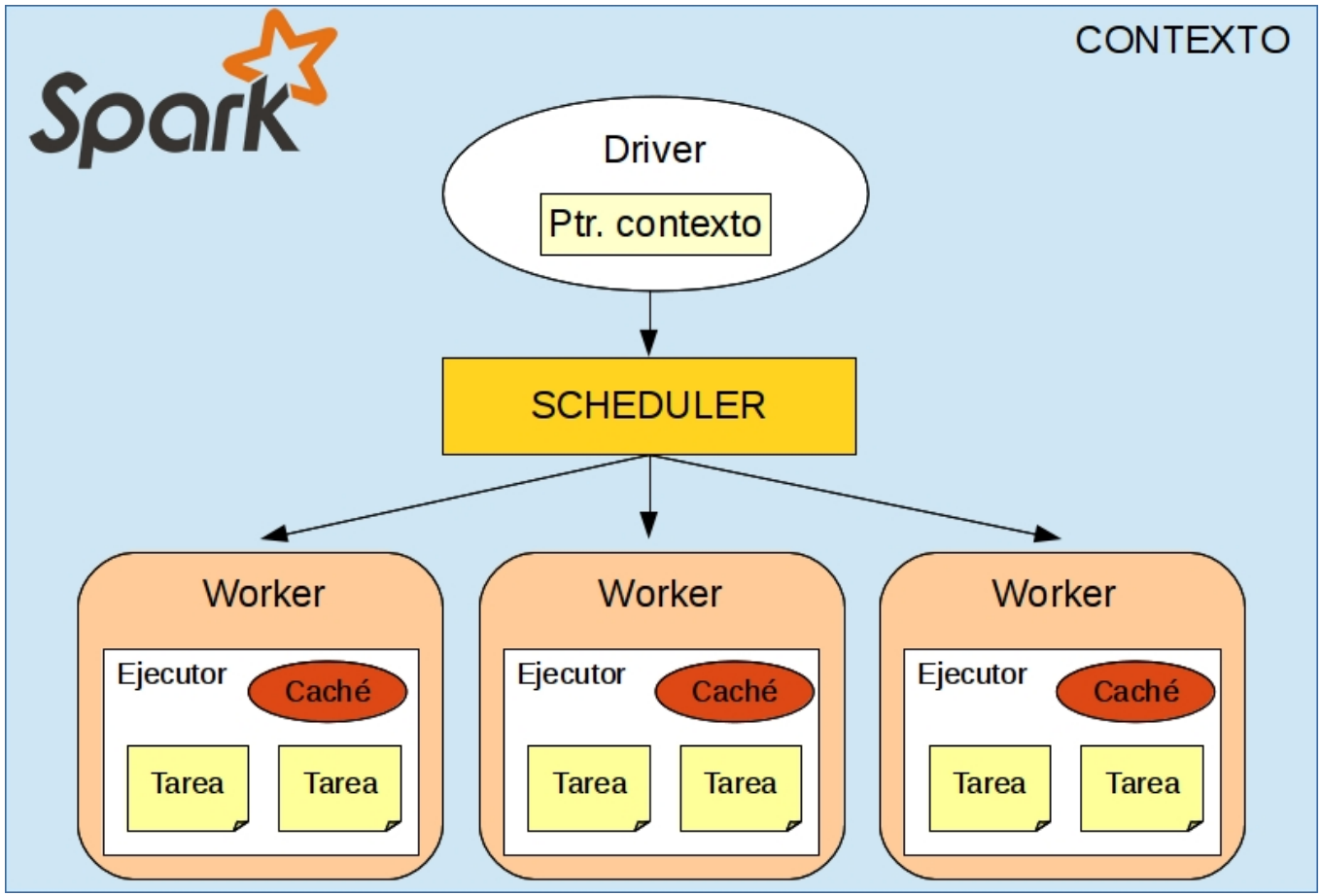
Spark en Acción (II)

- Regresión logística en Spark [3] con Python.

```
points = spark.textFile(...).map(parsePoint).cache()
w = numpy.random.ranf(size = D) # current separating plane
for i in range(ITERATIONS):
    gradient = points.map(
        lambda p: (1 / (1 + exp(-p.y*(w.dot(p.x)))) - 1) *
    p.y * p.x).reduce(lambda a, b: a + b)
    w -= gradient
print "Final separating plane: %s" % w
```

Arquitectura cluster en Spark

- `http://spark.apache.org/docs/latest/cluster-overview.html`.



Contexto en Spark

- Instancia a alto nivel que controla los datos de una aplicación (nodos, recursos, etc.).
- El objeto `SparkContext` coordina las tareas en ejecución.
- El objeto `SparkContext` puede usar varios planificadores intermediarios (YARN, Mesos, etc.) para acceder a los recursos del cluster.

```
# Creación de un SparkContext en pyspark
from pyspark import SparkConf, SparkContext

conf = SparkConf().setMaster("local").setAppName("My App")
sc = SparkContext(conf = conf)
```

Planificadores de tareas

- Por defecto, Spark utiliza un planificador de tareas FIFO.
  - Simple, no añade sobrecarga.
  - Puede retrasar tareas que queden encoladas a la espera de que finalicen trabajos complicados.
- Desde la versión 0.8, Spark permite configurar un planificador de tareas `FAIR` (similar a Round Robin).
  - También admite configuración de *pools* de tareas con diferente prioridad.
- Desde la versión 1.2 Spark también permite asignación dinámica de recursos del clúster (solo YARN).

```
# FAIR scheduler con pyspark
val conf = new SparkConf().setMaster(...).setAppName(...)
conf.set("spark.scheduler.mode", "FAIR")
val sc = new SparkContext(conf)
```

Resilient Distributed Datasets

- Elemento central de programación en Spark.
- Modelan una colección de objetos **inmutable**, **distribuida** y **tolerante a fallos**.
- Los objetos que contiene se pueden distribuir en diferentes nodos del cluster para procesamiento en paralelo.
- Pueden contener cualquier objeto (Python, Scala o Java) que sea serializable.

Creación de RDDs

- Dos formas de creación:
  - Cargando un conjunto de datos externo.
  - Paralelizando una colección de objetos ya existente.

```
# Carga de un conjunto de datos externo
lines = sc.textFile("README.md")

# Paralelización de una colección de objetos
data = ['tokenA', 'tokenB', 'tokenC', 'tokenD', 'tokenE']
distData = sc.parallelize(data)

# Comprobamos
In [5]: type(distData)
Out[5]: pyspark.rdd.RDD
```

Operaciones sobre RDDs

- **Transformación**: Creación de un nuevo conjunto de datos a partir de otro conjunto de datos inicial.
- **Acción**: Devuelven un valor al programa *driver* tras su ejecución sobre el conjunto de datos.
- *Lazyness*: Los RDDs solo se computan tras la primera acción aplicada sobre ellos.
- `http://spark.apache.org/docs/latest/programming-guide.html`

```
lines = sc.textFile("data.txt")  ## Solo almacena puntero a datos
lineLengths = lines.map(lambda s: len(s))  ## Todavía no se computa
## Genera y distribuye trabajos
totalLength = lineLengths.reduce(lambda a, b: a + b)

lineLengths.persist()  ## Salva en memoria para uso posterior
```

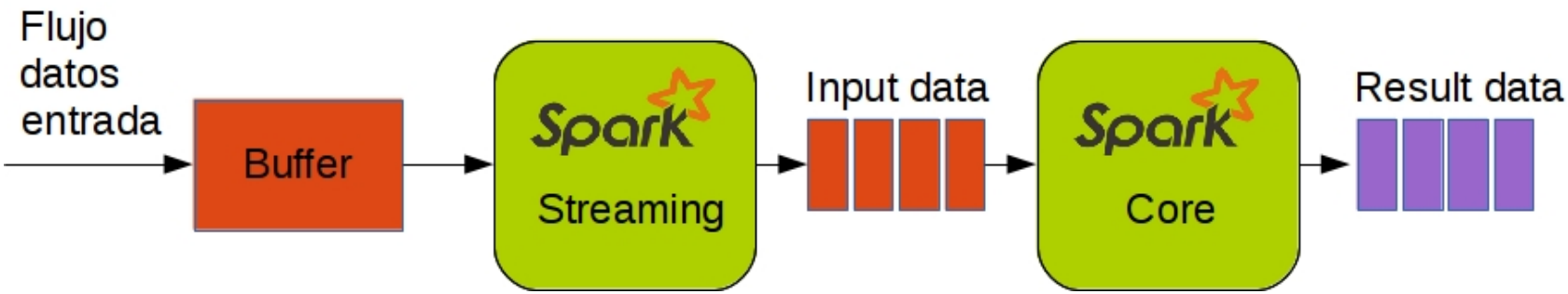
RDDs clave-valor

- Abstracción muy útil para composición de elementos básicos en aplicaciones.
- Podemos aprovechar operaciones específicas para operar sobre este tipo de conjuntos de datos.
  - `reduceByKey`
  - `groupByKey`
- La forma de creación difiere en cada lenguaje. En Python debemos pasar una función que genere tuplas de dos elementos (duplas).

```
# Número de ocurrencias de una línea en un archivo de texto
lines = sc.textFile("data.txt")
pairs = lines.map(lambda s: (s, 1))
counts = pairs.reduceByKey(lambda a, b: a + b)
```

Spark Streaming

- `http://spark.apache.org/docs/latest/streaming-programming-guide.html`.
- Interfaz en Python todavía en modo experimental.



Data Frames y Spark SQL

- Módulo de Spark para trabajo con datos estructurados.
- Permite tratar un esquema relacional como RDDs, en cualquiera de los tres lenguajes soportados por el entorno.
- Compatibilidad con numerosos estándares: tablas Hive, formato Parquet, datos JSON (!?).
- DataFrame: Abstracción para trabajar con tablas de un modelo relacional y distribuir consultas SQL en varios nodos de un cluster.

Programación con Spark SQL

- DataFrame: Similar a una tabla en modelo relacional, equivalente al mismo elemento en Python (Pandas) o en R, pero incluye optimizaciones para determinadas operaciones.
- Se pueden crear a partir de un RDD existente, de una tabla Hive, o de otros orígenes de datos (formato Parquet, archivos JSON, etc.).

```
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)

df = sqlContext.jsonFile("examples/src/main/resources/people.json")
df.printSchema()  # Imprime esquema del documento JSON
```

Programación con Spark SQL

- Podemos procesar los datos como RDDs o bien como un modelo relacional con SQL.

```
# Modo RDD
df.filter(lambda x: x.age > 21).collect()

# Modo SQL
df.registerAsTable("df_json")
result = sqlContext.sql("SELECT * FROM df_json where age > 21").
         collect()
```

MLlib

- Biblioteca Spark para algoritmos de *machine learning*.
- Incluye abstracciones para tipos de datos importantes en este contexto.
  - Vectores, vectores etiquetados.
  - Matrices locales, matrices distribuidas.
  - Soporte para matrices densas y dispersas.
- Numerosos algoritmos.
  - Clasificación, regresión, clustering, filtrado colaborativo, reducción de dimensionalidad, etc.
  - `https://spark.apache.org/docs/latest/mllib-guide.html`.

GraphX

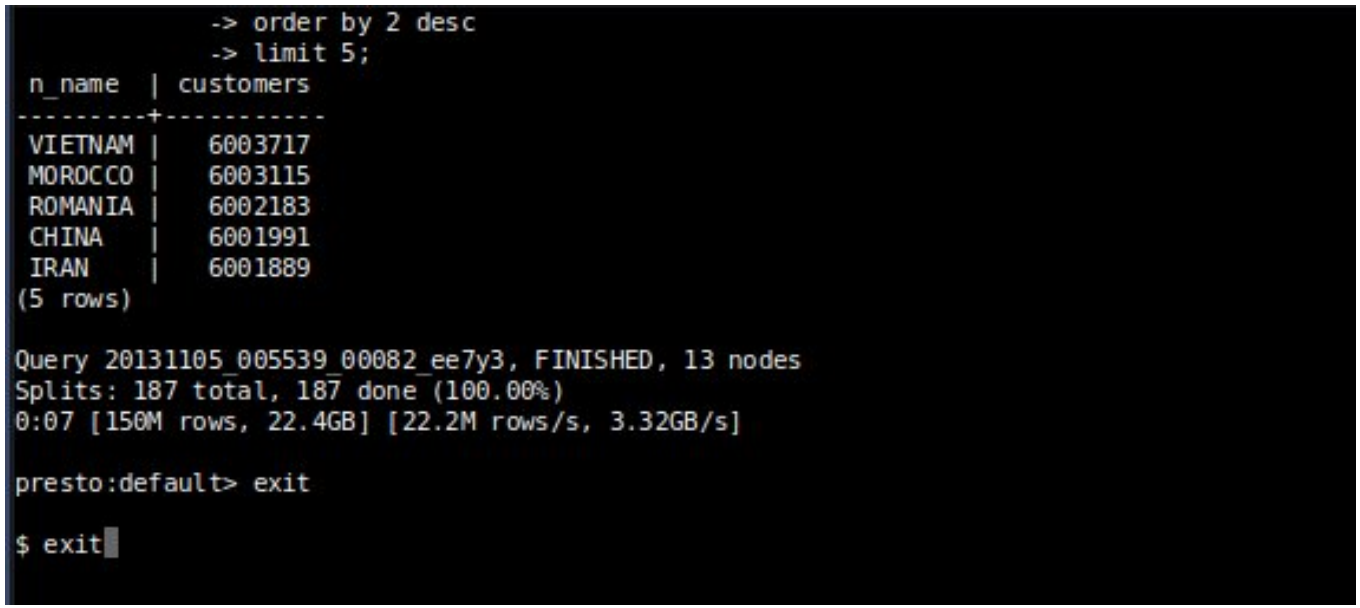
- Procesado de grafos (también en paralelo).
- Introduce abstracción `Graph`: multigrafo dirigido que soporta propiedades para enlaces y nodos.
- Incluye una biblioteca de algoritmos para facilitar la construcción y el análisis de grafos.
  - PageRank.
  - Aproximación de clusters (*connected components*, *triangle counting*.
- Por el momento, la única API de programación disponible es en Scala.

Presto (Facebook)

- Facebook posee uno de los almacenes de datos de mayor tamaño del mundo (+300 Petabytes).
- Necesidades: Análisis de grafos, machine learning y análisis interactivo.
- Motor de consultas SQL interactivo, enfocado en minimizar el tiempo de respuesta.

Presto (Facebook)

- Pequeña demo en línea (vídeo) [5].



Referencias

1. Karau, H., Konwwinski, A., Wendell, P., Zaharia, M. *Learning Spark*. O'Reilly Media Inc. Feb. 2015.
2. Karau, H., Sankar K. *Fast Data Processing with Spark*. 2nd Ed. Packt Publishing. Mar. 2015.
3. `http://spark.apache.org/examples.html`
4. `https://databricks-training.s3.amazonaws.com/index.html`
5. `https://prestodb.io/`