



Grupo de Usuarios de R de Sevilla

# Big Data con Spark y R

Manuel Chacón y Juan Andrés Tejero

4 Junio 2019, 19 h

Sala TIC4, CRAI Reina Mercedes

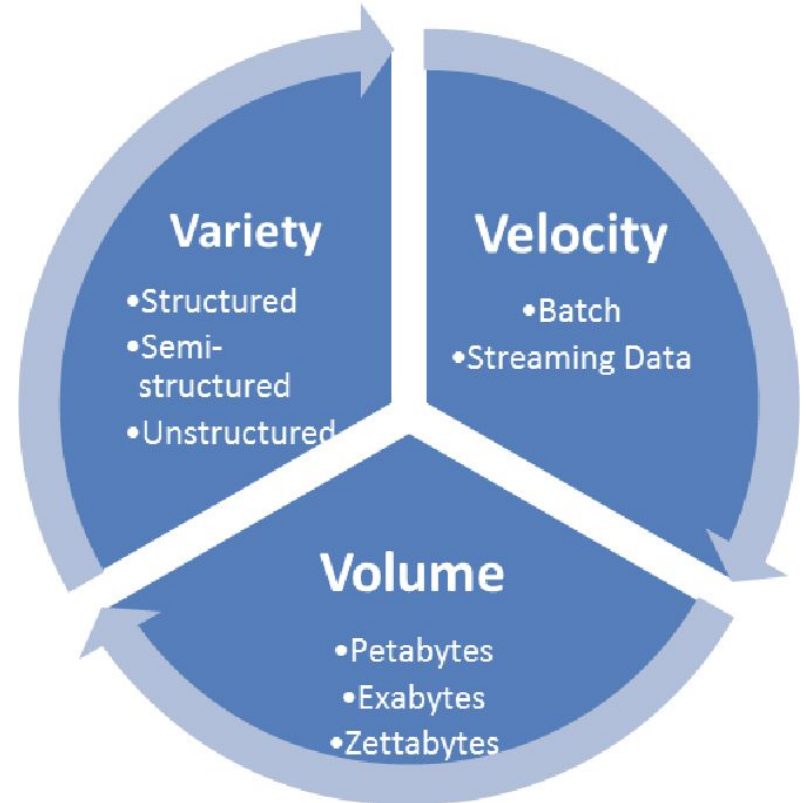
<http://bit.ly/SevillaRmeetup>

# Índice

1. Introducción al Big Data
2. Apache Hadoop
3. Apache Spark
  - a. Spark en R
4. Demos
5. Conclusiones

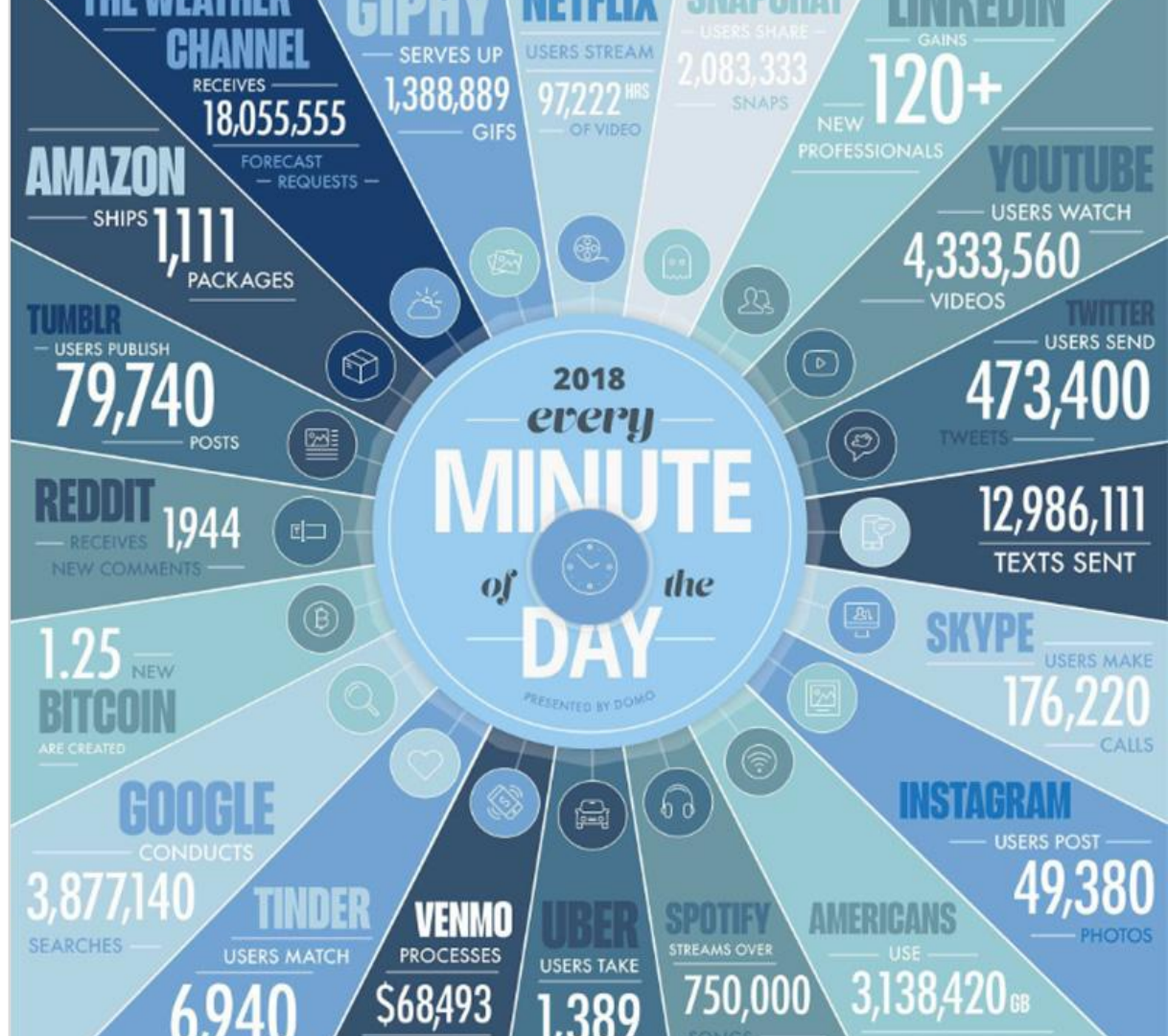
# Crecimiento de generación de datos.

- Big data es aplicar herramientas de ciencia del dato para procesar grandes volúmenes de datos.
- Aumento de la generación y consumo de datos.



# Volumen

- Auge de la tecnología.
- Internet
- Dispositivos móviles.
- Más usuarios.

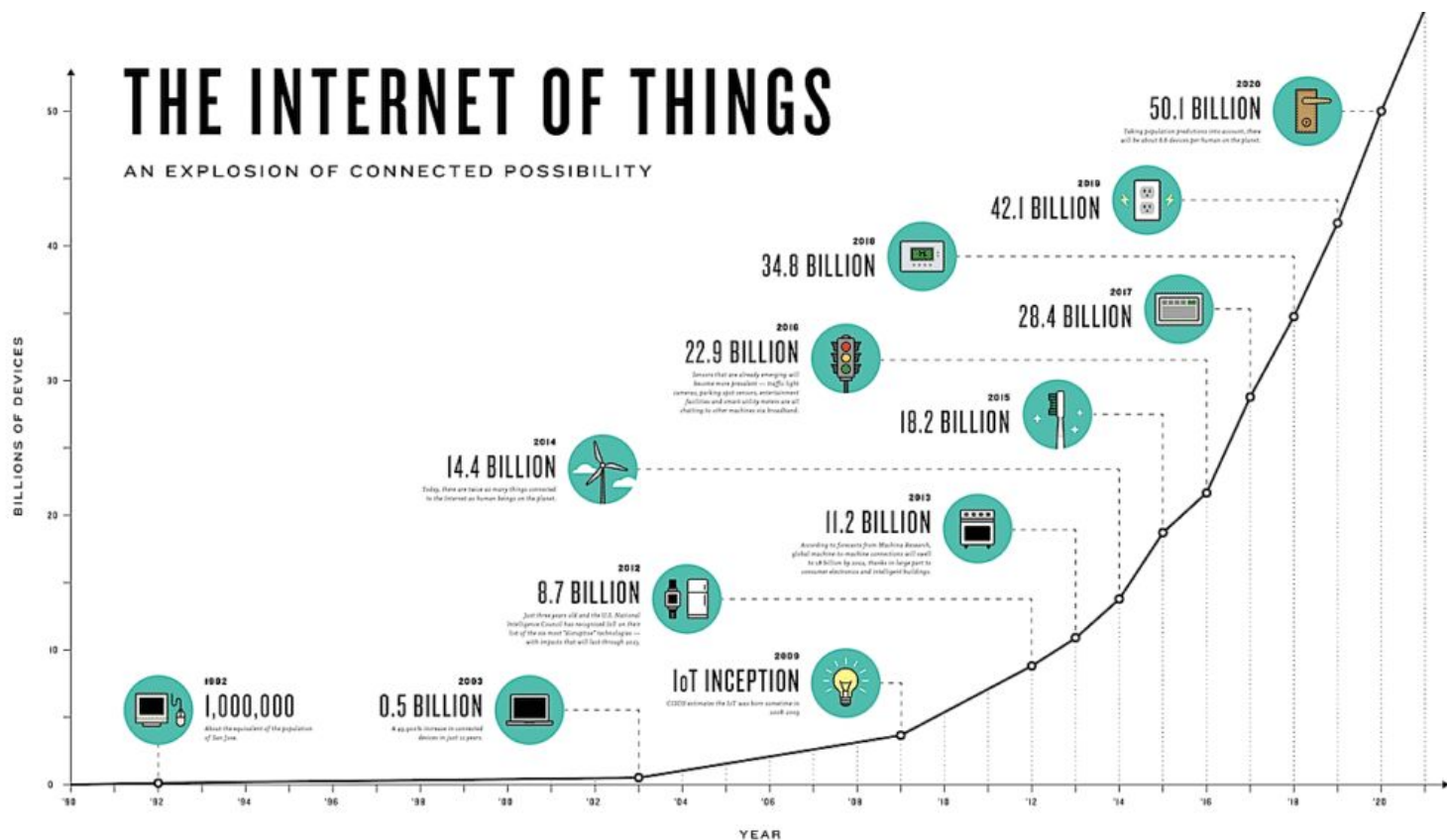




# Velocidad

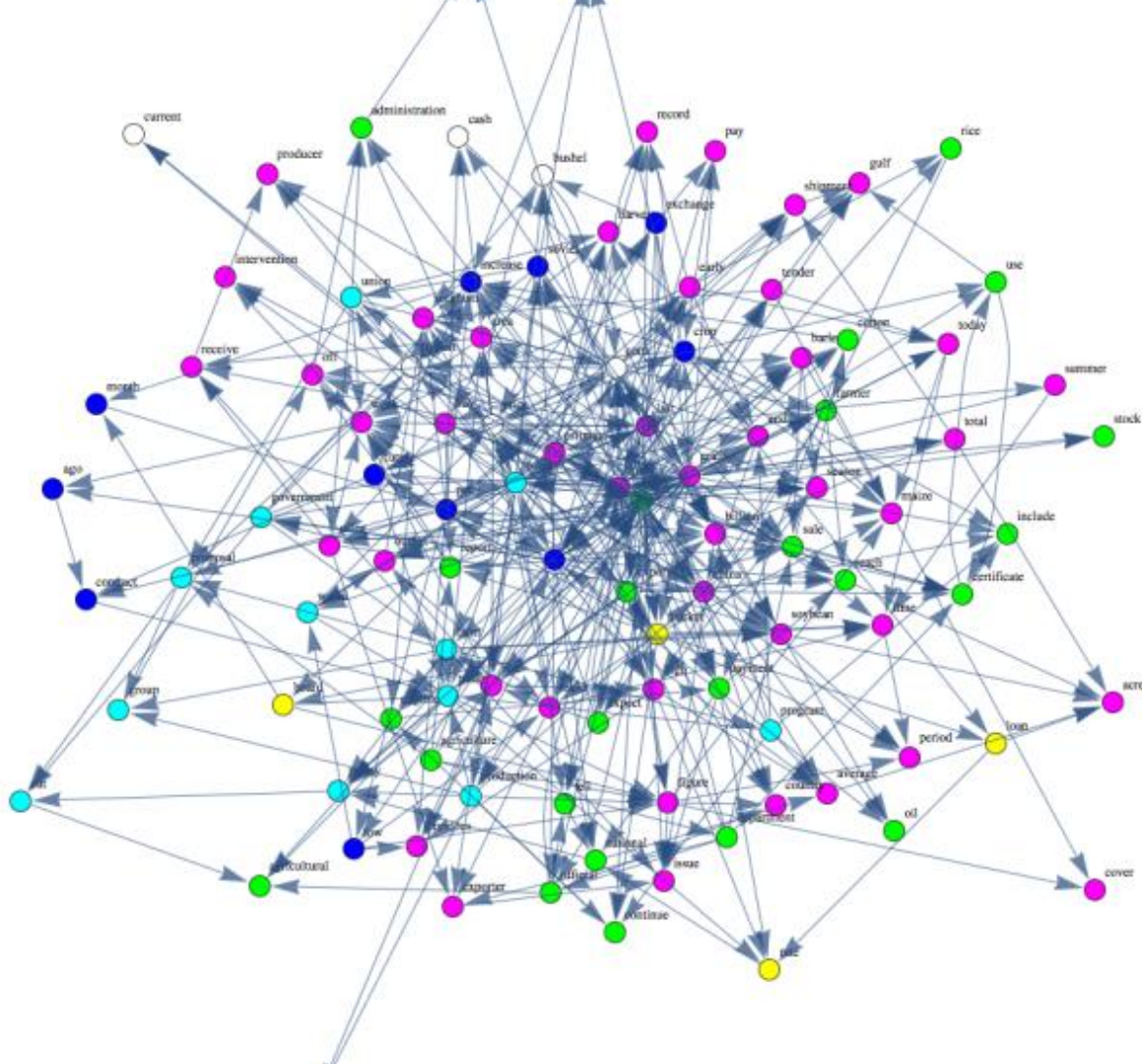
- Otra de la claves es la velocidad a la que los datos son generados y deben ser procesados.
- IoT, transacciones, imágenes, etc...
- Streaming

# Velocidad



# Variedad

- **Análisis de datos no estructurados:**
  - Textos libres
  - Todo tipo de registros de actividad
  - Datos multimedia





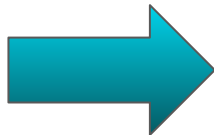
# ¿Cuál es la revolución?

## ANTES

Sistemas  
centralizados

Crecimiento vertical

Una única máquina



## AHORA

Sistemas distribuidos

Crecimiento horizontal

N máquinas

Arquitectura  
Maestro/esclavo



VS



# Hadoop

- Primera solución Big Data completa de propósito general.
- Lenguajes de consulta de alto nivel y transparentes.
- La comunicación entre nodos es mínima.
- Redundancia de datos, protección contra pérdida de datos.
- Tolerancia a fallos.

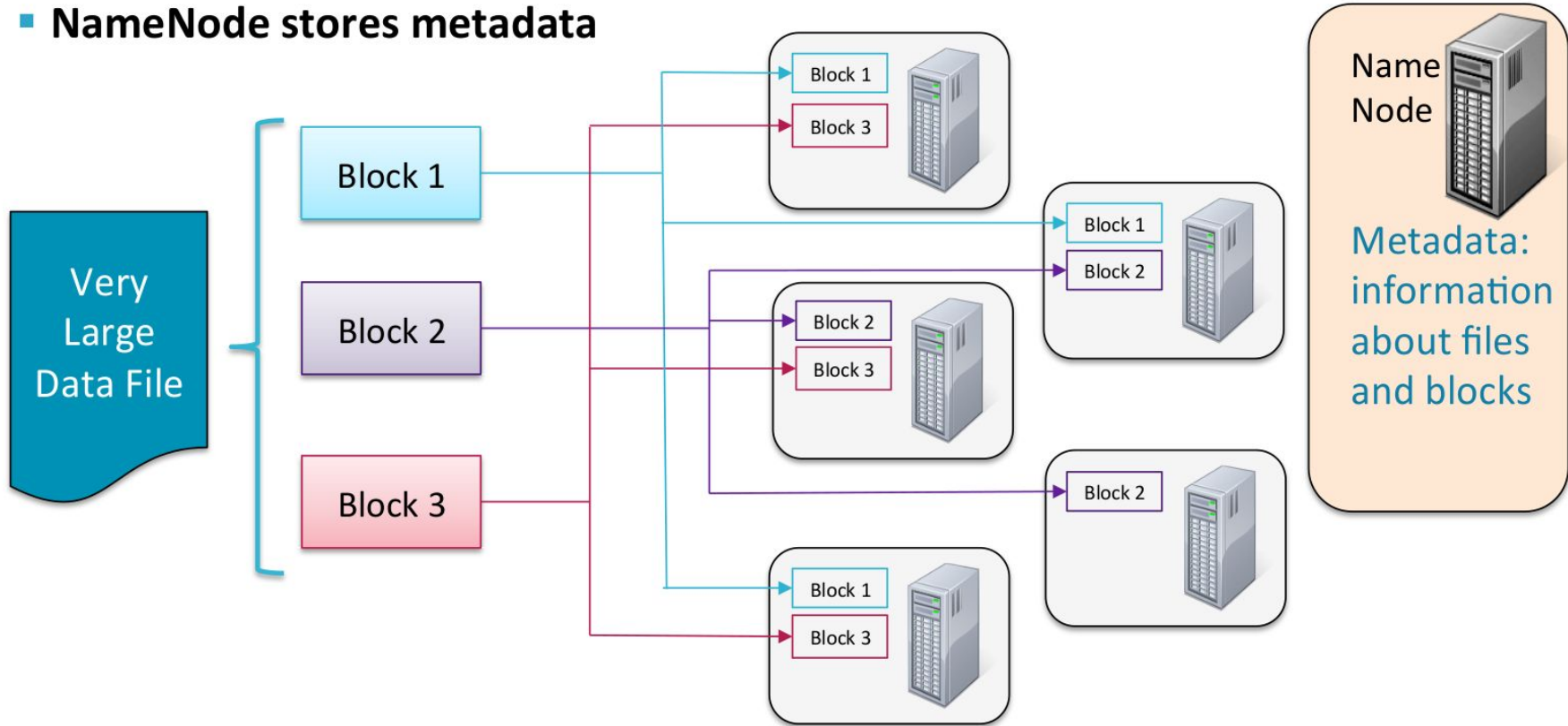


# HDFS: Hadoop Distributed Filesystem

- HDFS almacena ficheros→ texto, binarios, comprimidos, etc...
- Cuando almacenamos un fichero en HDFS, este es dividido en bloques.
- Tiene un tamaño de bloque fijo configurable.
- Estos bloques serán las unidades de datos en tiempo de procesamiento.



- Data files are split into blocks and distributed at load time
- Each block is replicated on multiple data nodes (default 3x)
- NameNode stores metadata



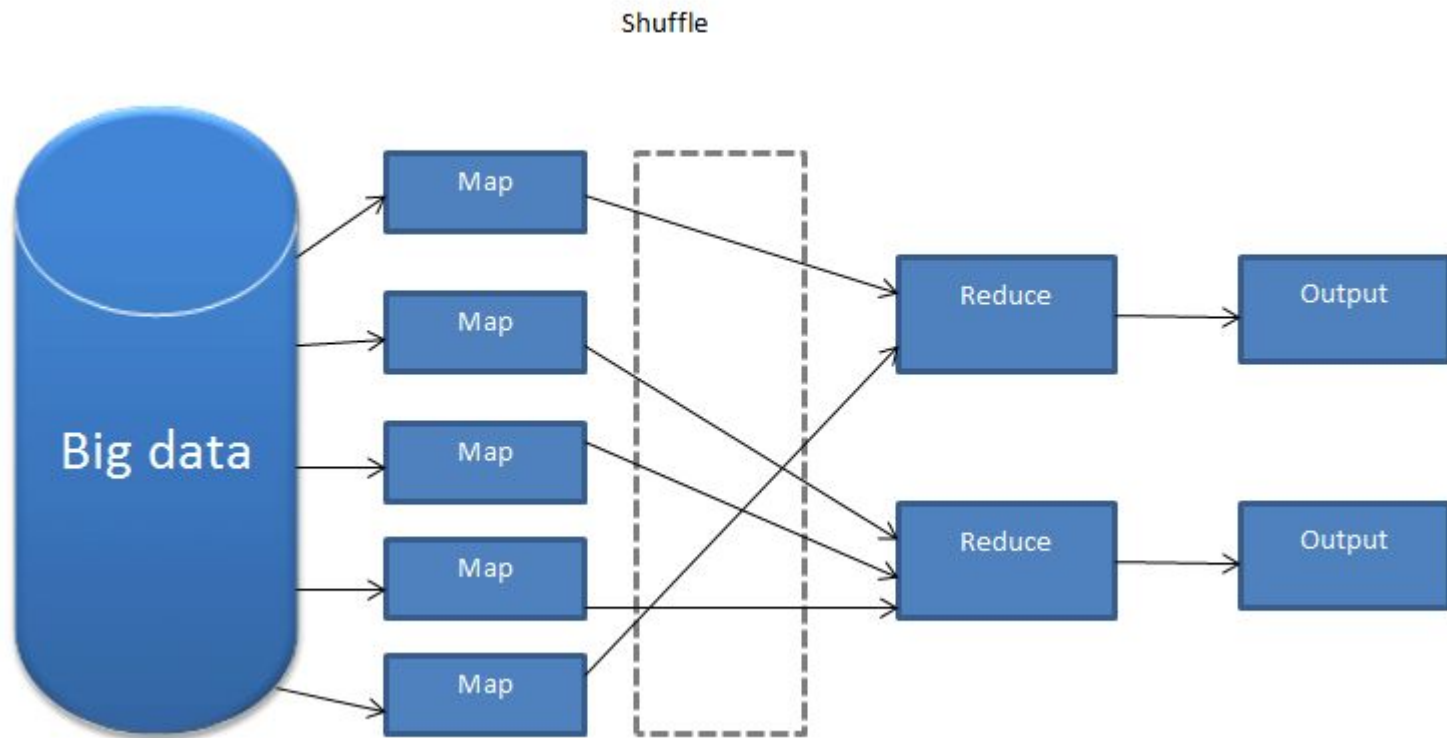
# MapReduce

- Es un framework de programación que permite distribuir tareas entre nodos.
- El procesamiento siempre consta de dos fases Map y Reduce.
- Map → Dado un elemento, aplica una operación de transformación (mapeo).
- Reduce → Dado un conjunto de elementos, aplica una operación de agregación (reducción).
- Utiliza un modelos clave-valor.
- Es transparente para el programador, solo diseña las fases Map y Reduce. Igual para una que para 100 máquinas.



# MapReduce

- Mapper
  - Cada tarea map actúa sobre un bloque de datos.
  - Se ejecuta donde se almacena el bloque.
- Shuffle
  - Organiza los datos originados como resultados de los mappers.
  - Esta fase tiene lugar una vez que todos los mappers han terminado y antes de que se inicien los reducers.
- Reducer
  - Opera sobre la salida (ordenada) de los mappers.
  - Aquí se genera el resultado final de la tarea.



# BIG DATA & AI LANDSCAPE 2018



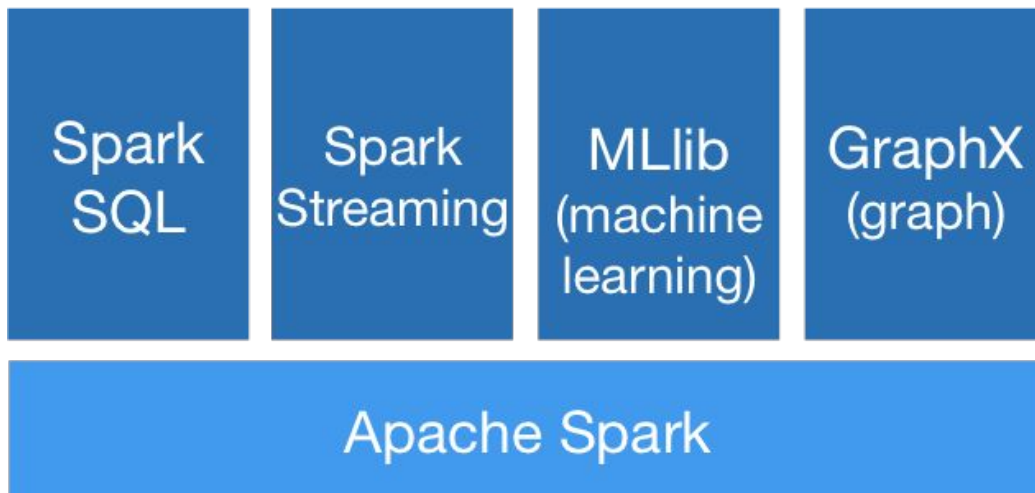


# Apache Spark

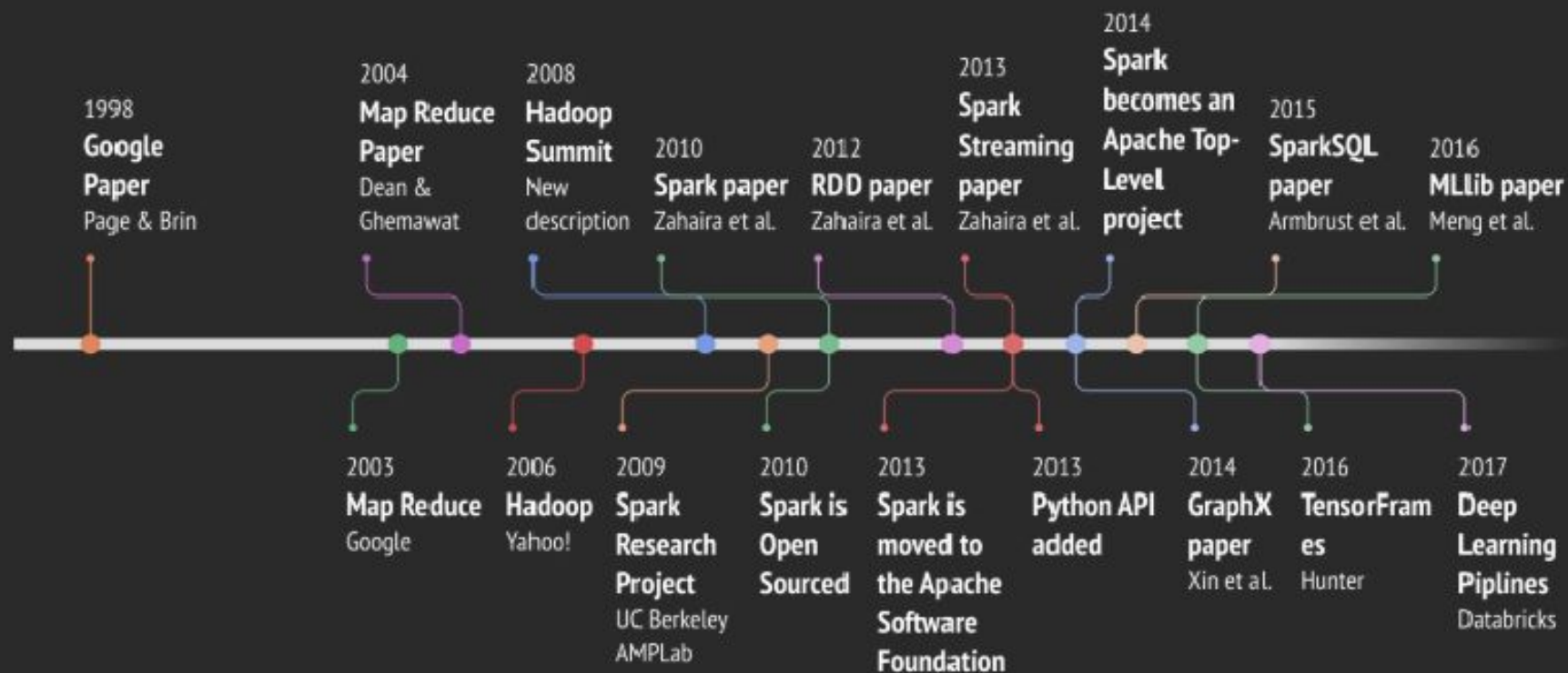


Framework open-source para el procesamiento distribuido en memoria de datos

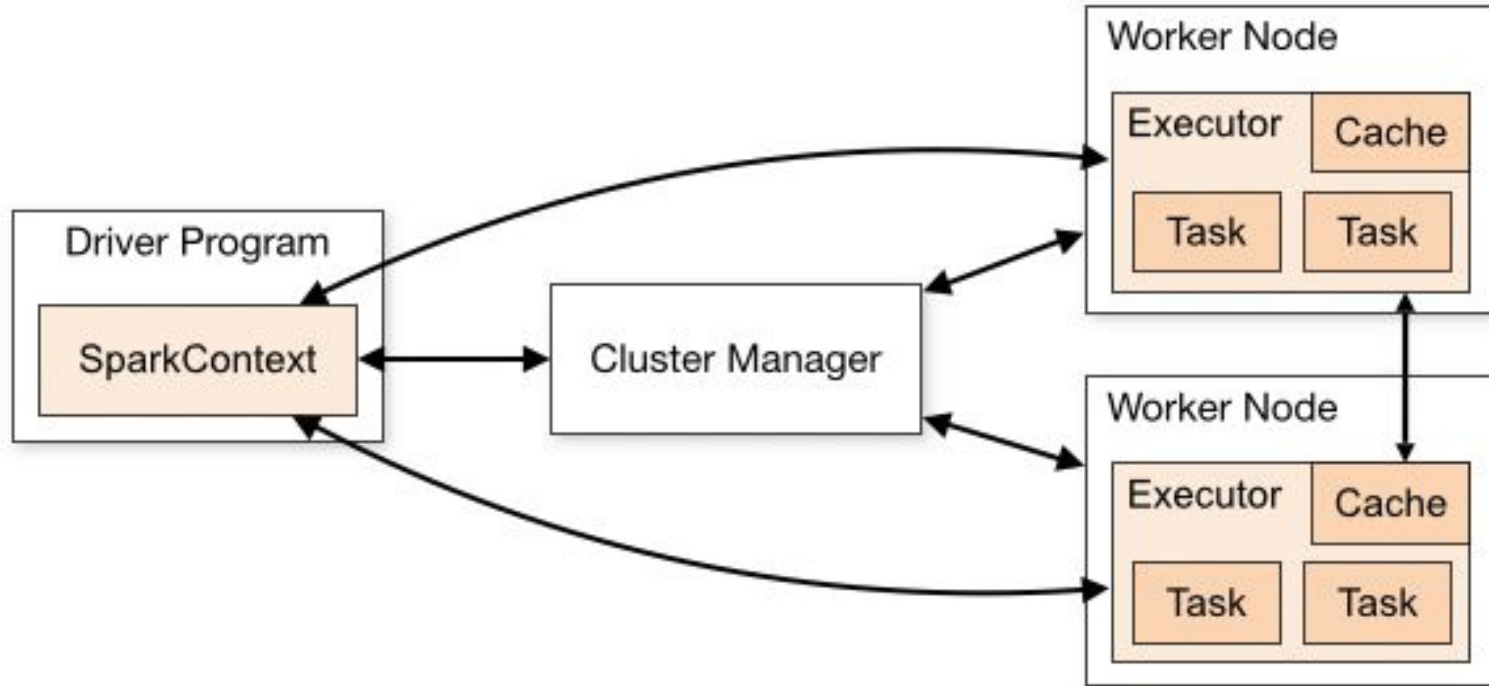
- Escrito en Scala → APIs para Scala, Java, Python ... y R
- Desarrollado por el AMPLab (Berkeley, CA) → Comunidad Open-Source



# Apache Spark Timeline

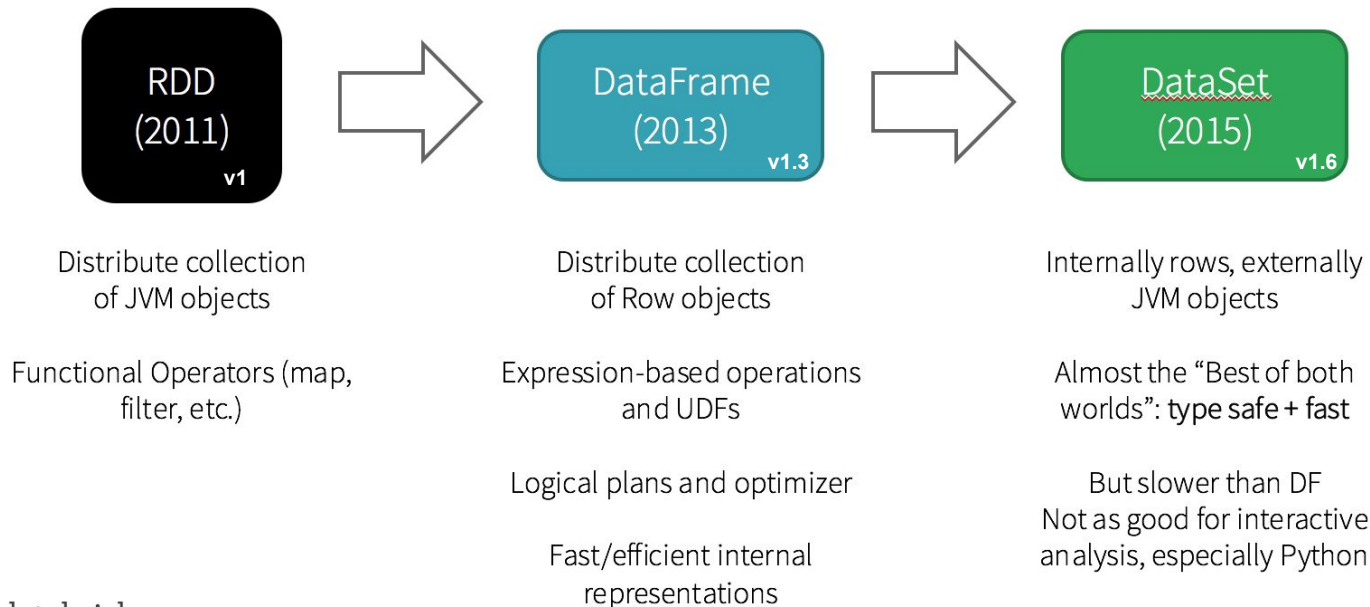


# Apache Spark: Funcionamiento



# Apache Spark: Funcionamiento - Estructura datos

## History of Spark APIs

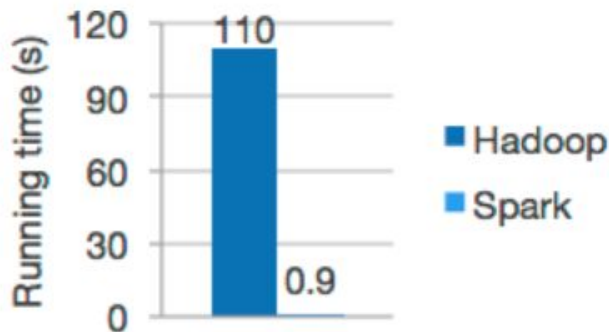


# Apache Spark: Funcionamiento - Memoria/Disco

Puede ejecutarse tanto en memoria como en disco

- x10 de velocidad en disco
- x100 de velocidad en memoria

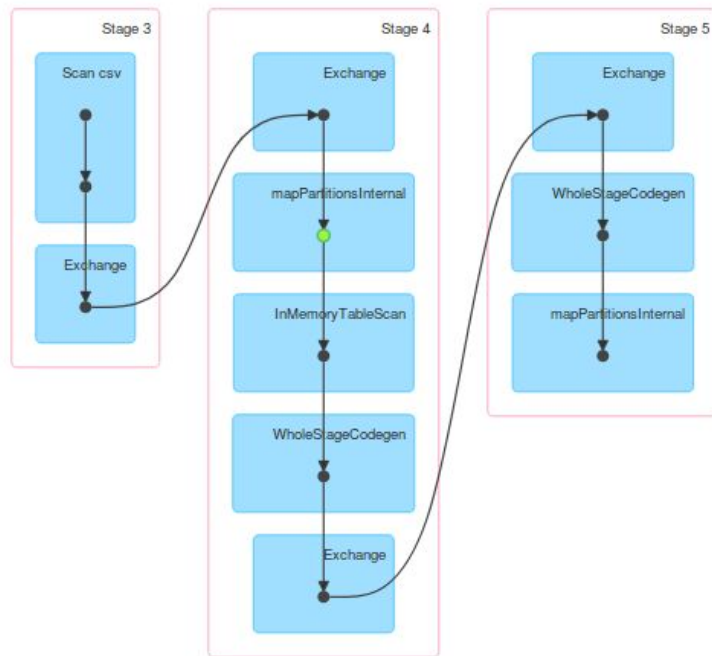
} vs. Hadoop



Logistic regression in Hadoop and Spark

# Apache Spark: Funcionamiento - DAGs

- Operaciones en Spark
  - Lazy evaluation
- { transformación  
acción



# Apache Spark: Funcionamiento - Modos

## Standalone

→ Incluido con Spark. Simple y fácil de configurar

## Apache Mesos

→ RM de propósito general para datacenters



## Hadoop YARN

→ RM incluido con Hadoop v2. Óptimo si se usa HDFS

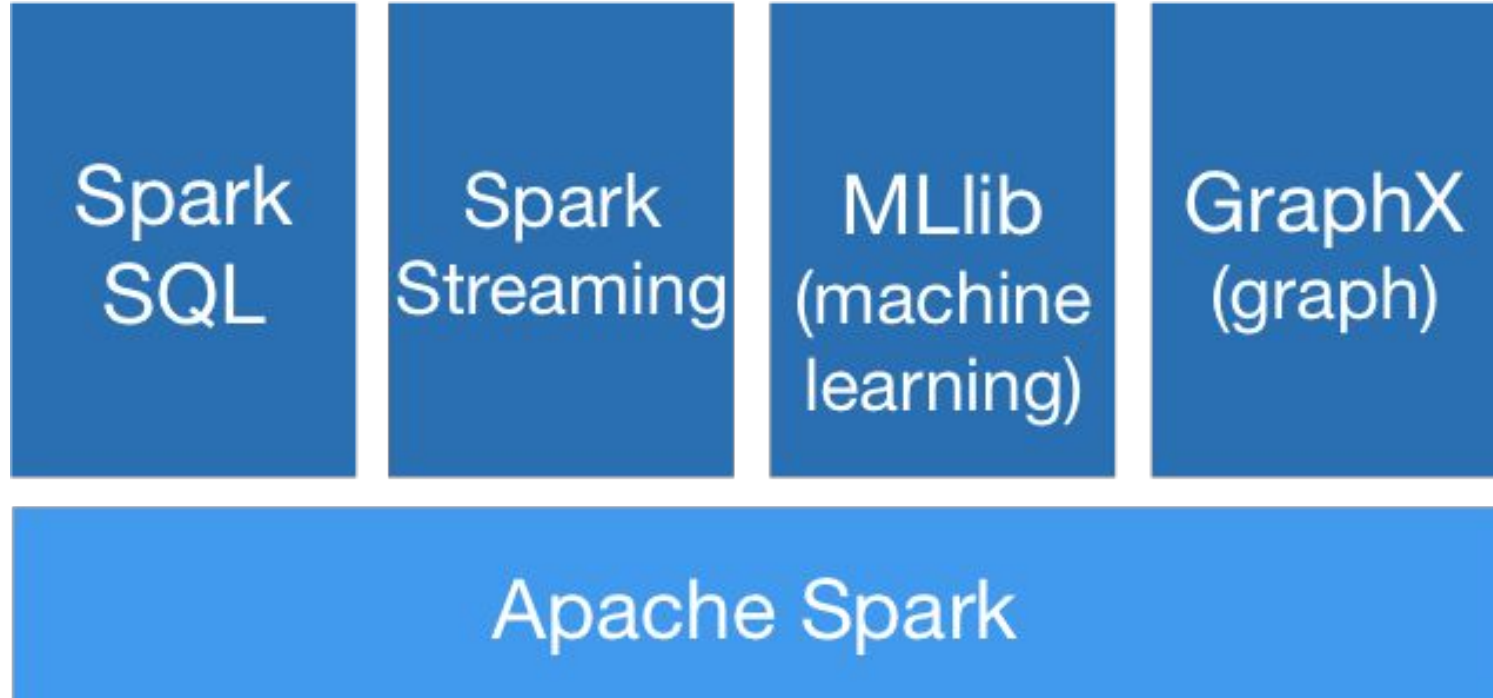


## Kubernetes





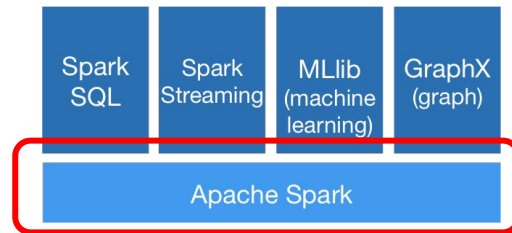
# Apache Spark: Módulos



# Apache Spark: Módulos

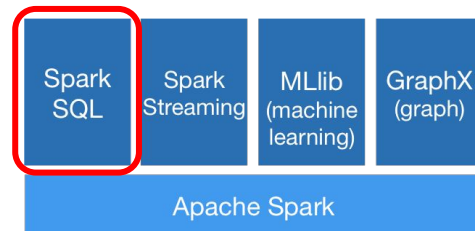
## Spark Core

- Funcionalidades básicas de spark
  - Scheduling
  - Gestión de memoria
  - Gestión de errores
  - Operaciones E/S
  - RDDs y operaciones básicas
    - Map, filter, reduce

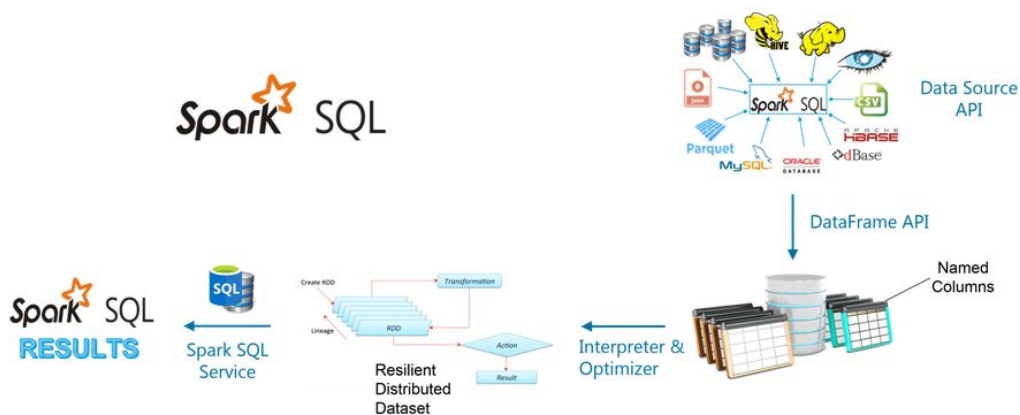


# Apache Spark: Módulos

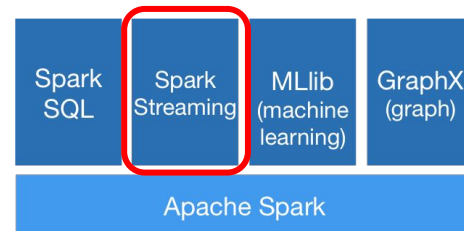
## Spark SQL



- Herramienta para trabajar con datos estructurados con SQL (HQL)
- Múltiples fuentes de datos
  - Hive, Parquet, ORC, JSON...
- Permite combinar consultas básicas SQL con analítica compleja

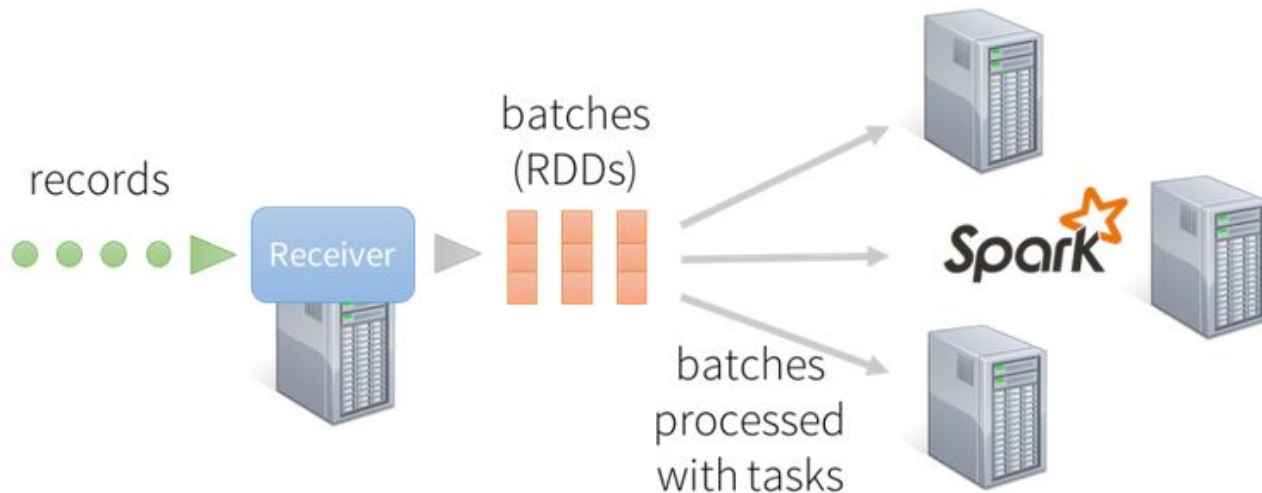


# Apache Spark: Módulos

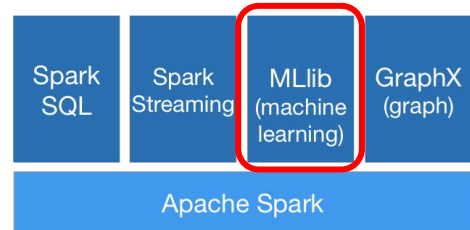


## Spark Streaming

- Módulo para el procesamiento en tiempo real de datos
- Micro-batching → Apache Flink



# Apache Spark: Módulos



**MLlib:** Librería de ML nativa de Spark

- Ejecución de manera distribuida
- LR, Naive Bayes, GLM, Random Forest, GBT, ALS, K-means...
- Catálogo limitado → Necesidad de usar librerías de terceros

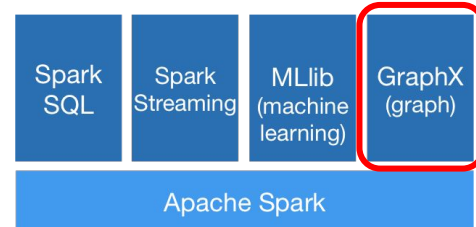
H2O → Sparkling Water

MMLlib → LightGBM, CNTK

Tensorflow on Spark

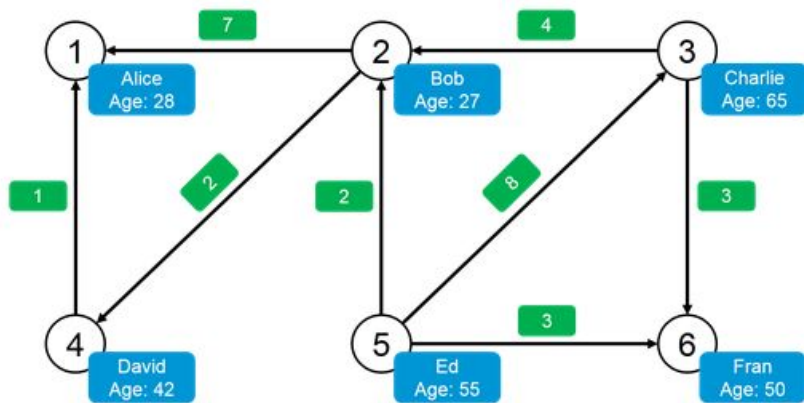


# Apache Spark: Módulos



## GraphX

- Extensión para trabajar con datos estructurados en forma de grafos
  - RRSS, redes, transporte, sistemas de recomendaciones, problemas geoespaciales...
  - RDD → Resilient Distributed Property Graph (RDPG)
- Grafos múltiples dirigidos
  - Permiten la ejecución en paralelo de acciones
- Operaciones especiales sobre grafos
  - `joinVertices`, `subgraph`, `aggregateMessages`...



# Apache Spark: SparkR

- Paquete oficial de Spark para R
- Desarrollado inicialmente por el AMPLab → Open-source
  - Integración con el proyecto Apache Spark desde la v1.4
- Bastantes limitaciones
  - Spark Streaming y GraphX aún no implementados
  - No soporta conexión con aplicaciones de terceros
  - No soporta modo yarn-cluster
  - `Data.table...` :(



# WELCOME TO *ggplot2* Unit



FEAT. MARKOV CHAIN MC

**Wicked Wickham**

# Apache Spark: Sparklyr



# Apache Spark: Sparklyr

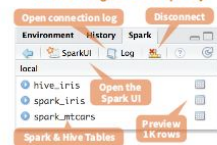
## Data Science in Spark with Sparklyr : : CHEAT SHEET

### Intro

**sparklyr** is an R interface for Apache Spark™. It provides a complete **dplyr** backend and the option to query directly using **Spark SQL** statement. With sparklyr, you can orchestrate distributed machine learning using either **Spark's MLlib** or **H2O Sparkling Water**.

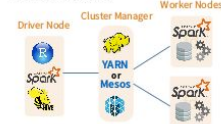
Starting with **version 1.044**, **RStudio Desktop**, **Server and Pro** include **integrated support for the sparklyr package**. You can create and manage connections to Spark clusters and local Spark instances from inside the IDE.

#### RStudio integrates with sparklyr



### Cluster Deployment

#### MANAGED CLUSTER



#### STAND ALONE CLUSTER



### Data Science Toolchain with Spark + sparklyr



### Getting Started

#### LOCAL MODE (No cluster required)

1. Install a local version of Spark:  
`spark_install("2.0.1")`
2. Open a connection  
`sc <- spark_connect(master = "local")`

#### ON A MESOS MANAGED CLUSTER

1. Install RStudio Server or Pro on one of the existing nodes
2. Locate path to the cluster's Spark directory
3. Open a connection  
`spark_connect(master="mesos URL", version = "1.6.2", spark_home = [Cluster's Spark path])`

#### USING LIVY (Experimental)

1. The Livy REST application should be running on the cluster
2. Connect to the cluster  
`sc <- spark_connect(method = "livy", master = "http://host:port")`

### Tuning Spark

#### EXAMPLE CONFIGURATION

```
config <- spark_config()
config$spark.executor.cores <- 2
config$spark.executor.memory <- "4G"
sc <- spark_connect(master="yam-client",
  config=config, version = "2.0.1")
```

#### IMPORTANT TUNING PARAMETERS with defaults

```
• spark.yam.am.cores • spark.executor.instances
• spark.yam.am.memory 512m • spark.executor.extraJavaOptions
• spark.network.timeout 120s • spark.executor.heartbeatInterval 10s
• spark.executor.memory 1g • sparklyr.shell.executor-memory
• spark.executor.cores 1 • sparklyr.shell.driver-memory
```

### Using sparklyr

A brief example of a data analysis using Apache Spark, R and sparklyr in local mode

```
library(sparklyr); library(dplyr); library(ggplot2);
library(tidy);
set.seed(100)
```

`spark_install("2.0.1")` **Connect to local version**

`sc <- spark_connect(master = "local")`

`import_iris <- copy_to(sc, iris, "spark_iris", overwrite = TRUE)` **Copy data to Spark memory**

`partition_iris <- sdf_partition(import_iris, training=0.5, testing=0.5)` **Partition data**

`sdf_register(partition_iris, c("spark_iris_training", "spark_iris_test"))`

**Create a hive metadata for each partition**

`tidy_iris <- tbl(sc, "spark_iris_training") %>% select(Species, Petal_Length, Petal_Width)`

`model_iris <- tidy_iris %>% ml_decision_tree(response="Species", features=c("Petal_Length", "Petal_Width"))` **Spark ML Decision Tree Model**

`test_iris <- tbl(sc, "spark_iris_test")` **Create reference to Spark table**

`pred_iris <- sdf_predict(model_iris, test_iris) %>% collect` **Bring data back into R memory for plotting**

`pred_iris %>% inner_join(data.frame(prediction=0.2, lab=model_iris$model.parameters.labels) %>% ggplot(aes(Petal_Length, Petal_Width, col=lab)) + geom_point())`

`spark_disconnect(sc)` **Disconnect**

# Apache Spark: SparkR vs. Sparklyr

- Integrado con los principales paquetes de R: dplyr, DBI, broom...
- Soporta ML Pipelines y procesamiento de grafos (GraphX)
- Compatible con Livy y YARN
- Es posible exportar modelos de ML a Java usando MLeap
- Integración con RStudio y RStudio Server
- Compatible con extensiones Scala
- Soporta extensiones de terceros: H2O, SAS data...

# Apache Spark: SparkR vs. Sparklyr

Feature	SparkR	sparklyr
Data input & output	++	++
Data manipulation	-	+++
Documentation	++	++
Ease of setup	++	++
Function naming	--	+++
Installation	+	++
Machine learning	+	++
Range of functions	+++	++
Running arbitrary code	+	++
Tidyverse compatability	---	+++

<https://eddyberry.netlify.com/post/2017-12-05-sparkr-vs-sparklyr/>

**IT'S**



**DEMO TIME!**

[memegenerator.es](http://memegenerator.es)

# Conclusiones

- Spark nos permite procesamiento en memoria distribuido de Big Data
  - a. Streaming, graph processing, SQL y ML
  - b. Tener en cuenta sobrecarga de dividir datos
- Usable desde R
  - a. SparkR → aún verde
  - b. Sparklyr → más maduro y con compatibilidad con Tidyverse, H2O...
- Aún limitaciones en R
  - a. Necesidad de interconexión con librerías de ML (TF on Spark, MMLlib, scikit-learn...)





Grupo de Usuarios de R de Sevilla

# Big Data con Spark y R

Manuel Chacón y Juan Andrés Tejero

4 Junio 2019, 19 h

Sala TIC4, CRAI Reina Mercedes

<http://bit.ly/SevillaRmeetup>