Introducción a Spark

Daniel Higuero (daniel.higuero@gmail.com)

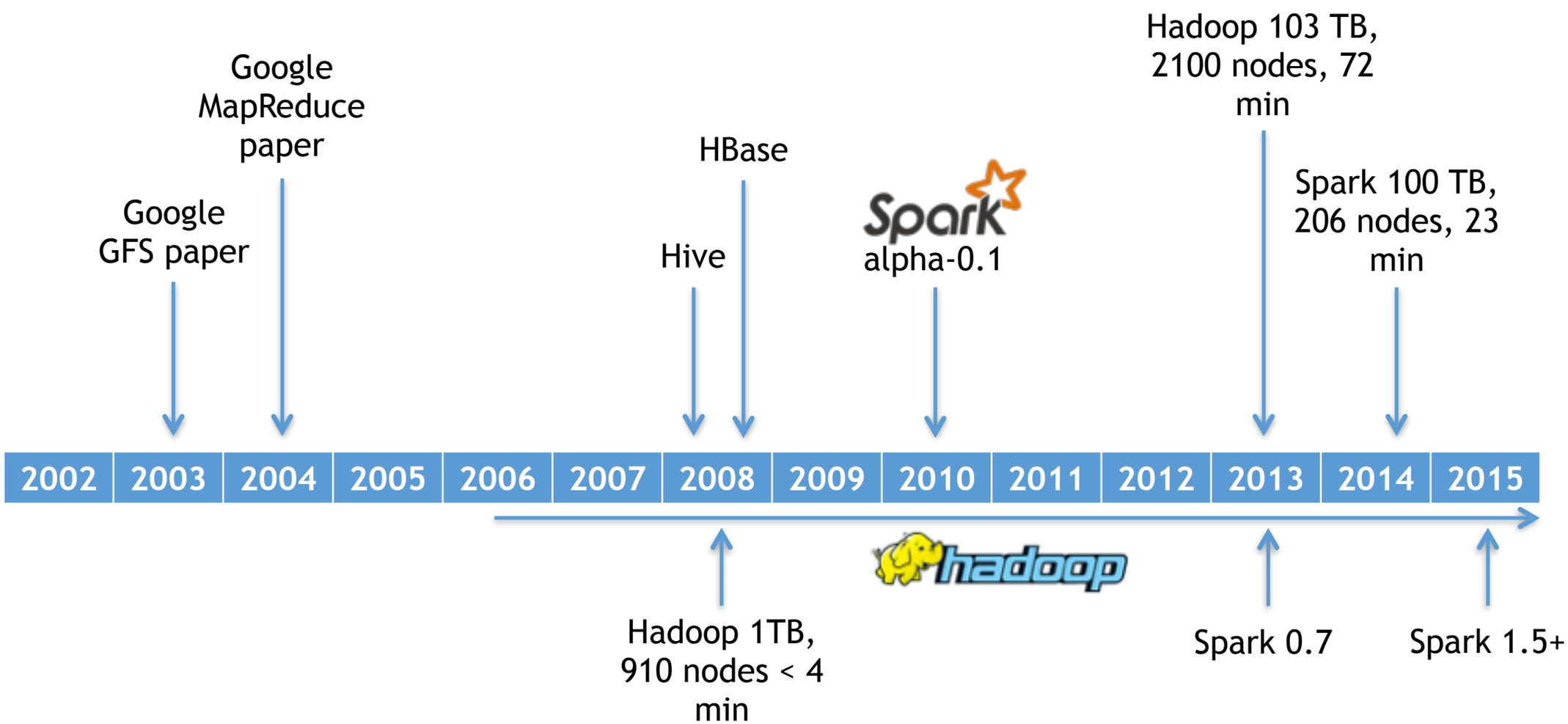
@dhiguero

Agenda

- Introducción
- Instalación
- Conceptos básicos
- Ejercicios

¿Qué es Spark ?

Timeline


Google MapReduce paper
Google GFS paper
HBase
Hive
Spark alpha-0.1
Hadoop 103 TB, 2100 nodes, 72 min
Spark 100 TB, 206 nodes, 23 min
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
hadoop
Hadoop 1TB, 910 nodes < 4 min
Spark 0.7
Spark 1.5+

¿Qué es Spark?

- Creado en Scala
- Framework de procesamiento paralelo
  - Orientado a Batch
  - Orientado a Streaming

amplab UC BERKELEY → Apache Software Foundation → databricks™

Map-reduce

- Programación funcional

```
(map 'list (lambda (x) (+ x 10)) '(1 2 3 4))
                                    => (11 12 13 14)
(reduce #'+ '(1 2 3 4)) => 10
```

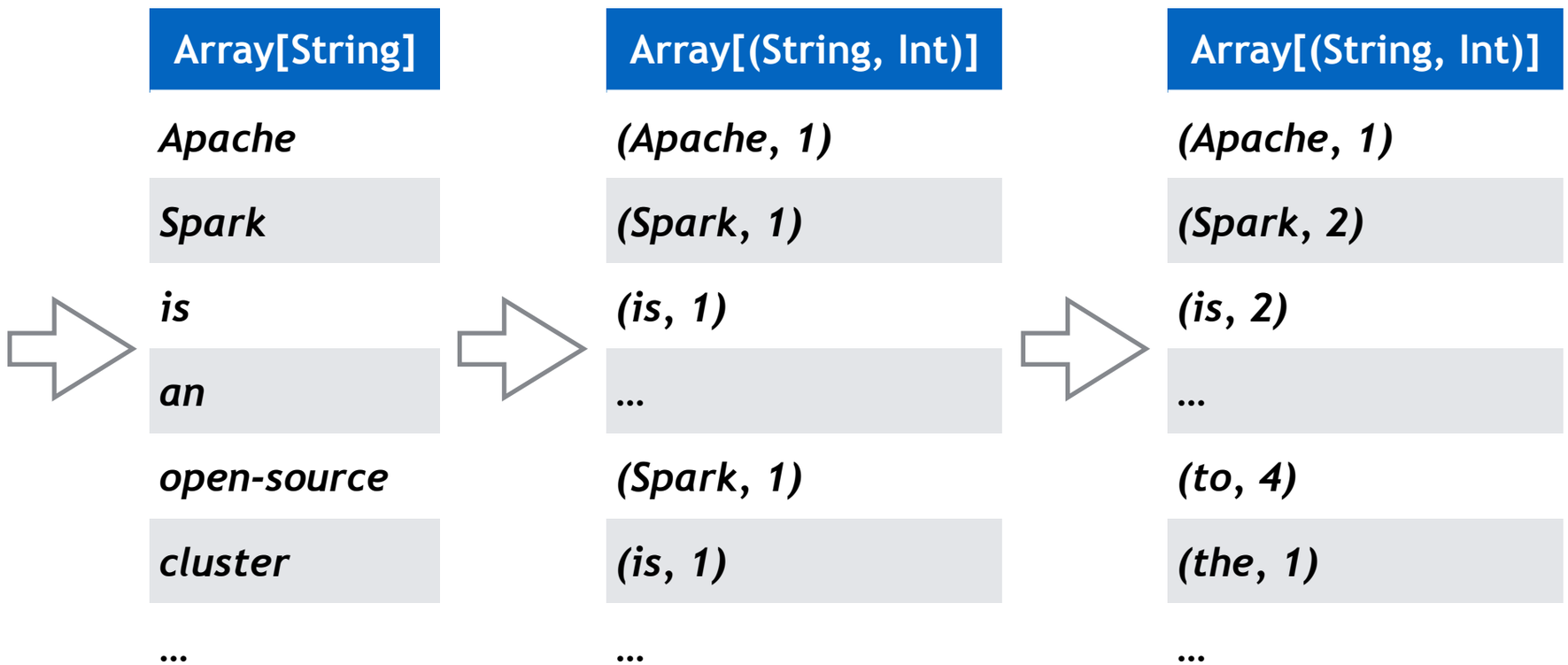
- Popularizado por Google

Jeff Dean and Sanjay Ghemawat. "*MapReduce: Simplified Data Processing on Large Clusters*." *OSDI* (2004)

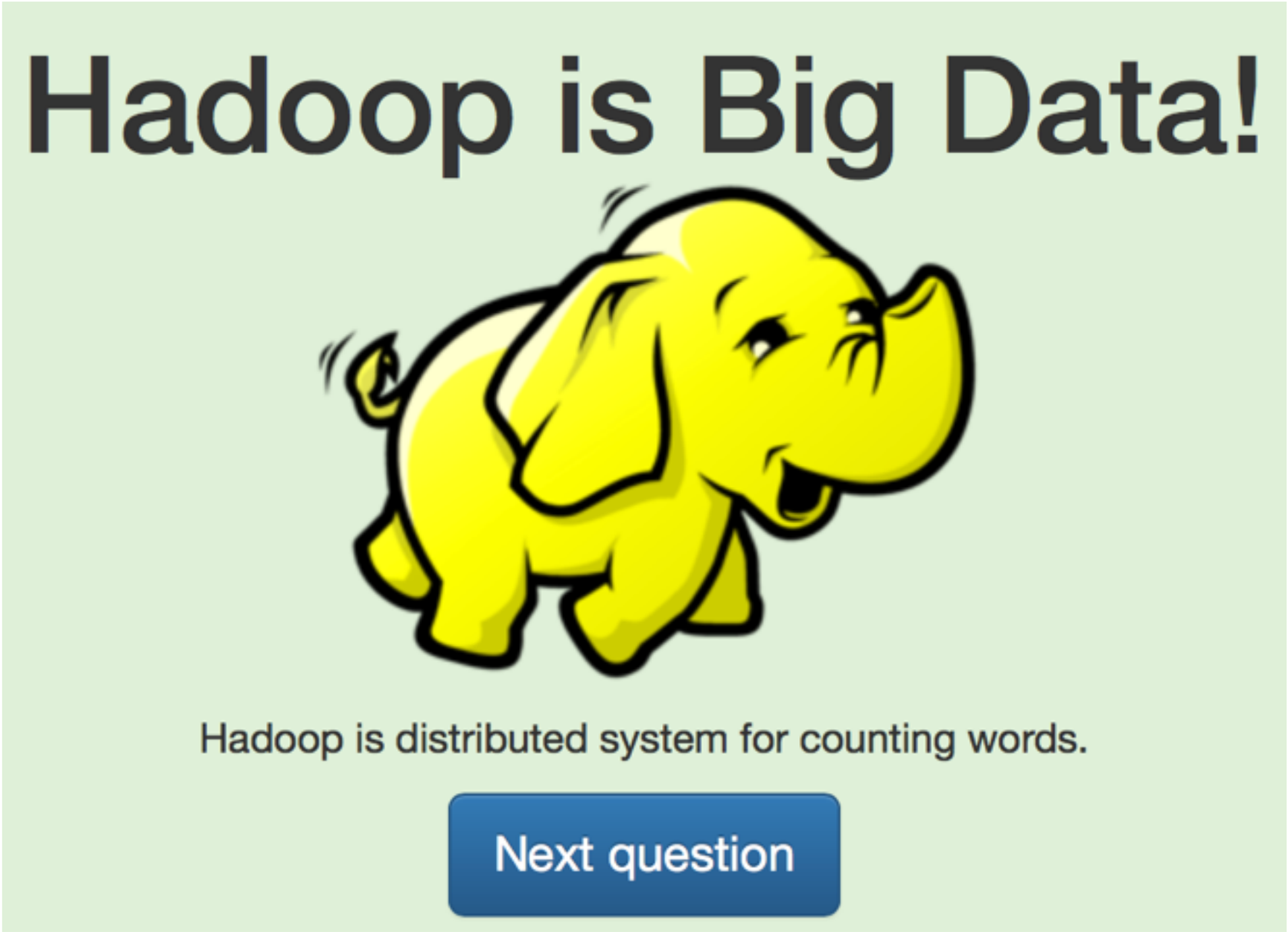
Map-reduce

```
val wordCounts = textFile.flatMap(line => line.split(" "))
                         .map(word => (word, 1))
                         .reduceByKey(_ + _)
```

Apache Spark is an open source cluster computing framework originally developed in the AMPLab at University of California, Berkeley but was later donated to the Apache Software Foundation where it remains today. In contrast to Hadoop's two-stage disk-based MapReduce paradigm, Spark's multi-stage in-memory primitives provides performance up to 100 faster for certain applications.



Yet another framework for counting words?



http://pixelastic.github.io/pokemonorbigdata/

Ventajas de Spark

- Uso más eficiente del almacenamiento
- Flexibilidad en la definición del pipeline de transformación
- Tolerancia a fallos
- Gran tracción de la comunidad
- Convertido en sistema de facto para Big Data
- Roadmap vivo

¿Qué alternativas conoces a Spark?

Instalación

Instalación

http://spark.apache.org/downloads.html

Spark Lightning-fast cluster computing

Download | Libraries | Documentation | Examples | Community | FAQ

Download Spark

The latest release of Spark is Spark 1.5.0, released on September 9, 2015 (release notes) (git tag)

1. Choose a Spark release: 1.5.0 (Sep 09 2015)
2. Choose a package type: Pre-built for Hadoop 2.6 and later
3. Choose a download type: Select Apache Mirror
4. Download Spark: spark-1.5.0-bin-hadoop2.6.tgz
5. Verify this release using the 1.5.0 signatures and checksums.

Note: Scala 2.11 users should download the Spark source package and build with Scala 2.11 support.

Instalación

```
$ $ curl -O http://apache.rediris.es/spark/spark-1.5.0/
spark-1.5.0-bin-hadoop2.6.tgz
$ tar xvzf spark-1.5.0-bin-hadoop2.6.tgz
$ cd spark-1.5.0-bin-hadoop2.6
$ cp conf/spark-env.sh.template conf/spark-env.sh
$ ./bin/spark-shell
```

Spark-Shell

```
$ ./bin/spark-shell
. . .
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 1.5.0
      /_/

Using Scala version 2.10.4 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_60)
Type in expressions to have them evaluated.
Type :help for more information.
Spark context available as sc.
SQL context available as sqlContext.

scala>
```

Basics

Inmutable

In-memory

RDD and Dataframes

RDD

- Abstracción fundamental en Spark
- Abstracción básica en Spark
- Contiene las transformaciones que se van a realizar sobre un conjunto de datos
  - Inmutable
  - Lazy evaluation
  - En caso de fallo se puede recuperar el estado
  - Control de persistencia y particionado

Paralelizando datos

```
val data = Array(1, 2, 3, 4, 5)
val distData = sc.parallelize(data)
```

Map-reduce

```
val lines = sc.textFile("data.txt")
val lineLengths = lines.map(s => s.length)
val totalLength = lineLengths.reduce((a, b) => a + b)
lineLengths.persist()
```

Transformaciones vs acciones

- Las transformaciones crean un nuevo RDD a partir de uno existente
  - Modelo de ejecución lazy
- Las acciones disparan la ejecución de las etapas de un RDD

Dataframe

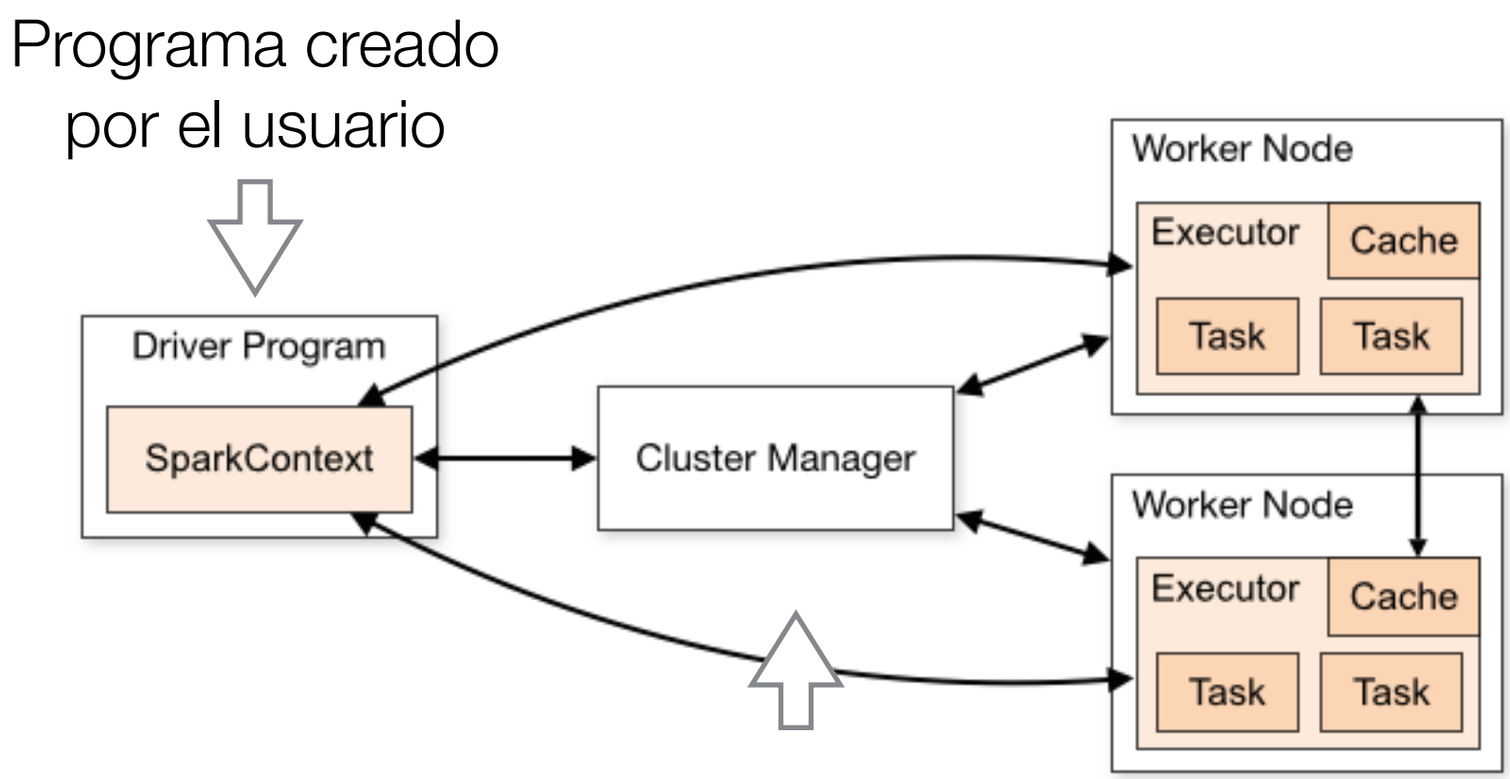
- Abstracción de alto nivel sobre un RDD
- Datos organizados en forma de tabla-columna
  - RDD + Schema
- Escalable en tamaño
- Aprovecha Catalyst
- Compatibilidad con la visión SQL
- Similar a Python Pandas o R dataframes

SparkContext

- Representa la conexión a un cluster de Spark
- Sólo puede existir uno activo
- Permite crear distintos tipos de variables
  - RDD
  - Accumulators
  - Broadcast

```
new SparkContext(master: String,
                 appName: String, conf: SparkConf)
```

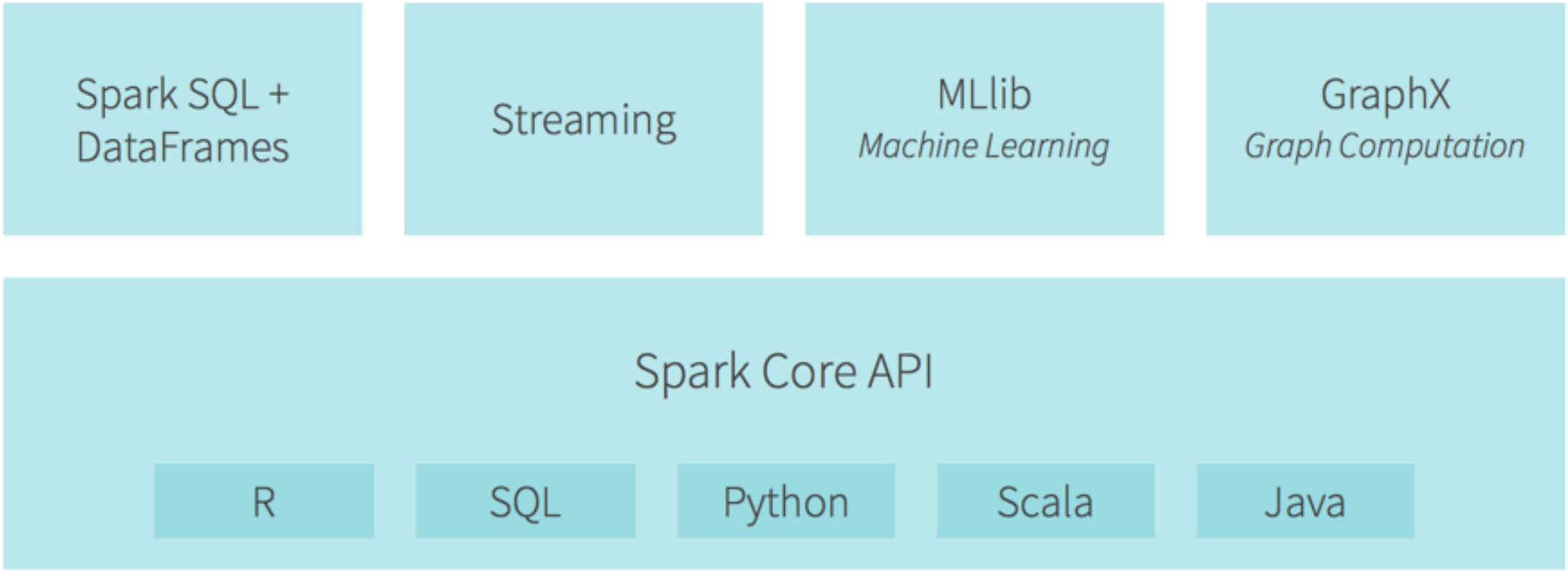
Cluster de Spark



Gestor de recursos:
local/local[k], standalone,
yarn, o mesos

Ecosistema Spark

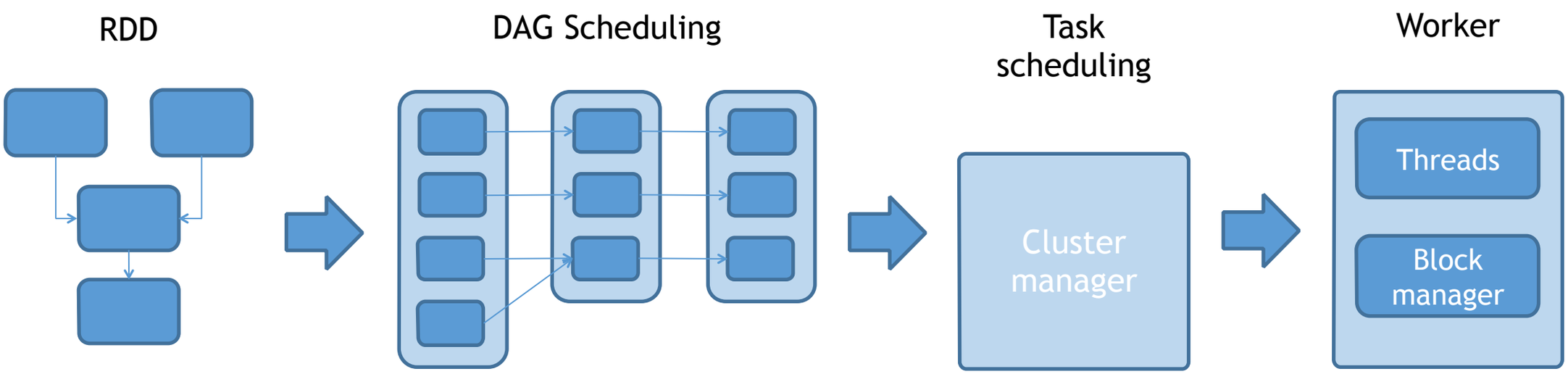
Ecosistema



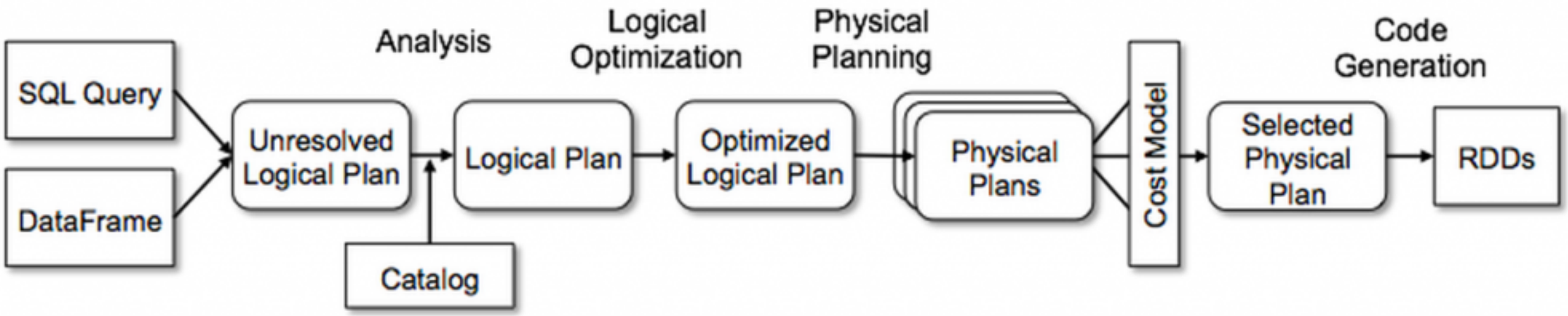
© databricks

Spark Core API

- Proporciona las abstracciones básicas y se encarga del schedulling



Catalyst


Analysis
Logical Optimization
Physical Planning
Code Generation
SQL Query
DataFrame
Unresolved Logical Plan
Catalog
Logical Plan
Optimized Logical Plan
Physical Plans
Cost Model
Selected Physical Plan
RDDs

SparkSQL + Dataframes

- Ofrece una abstracción unificada para la realización de consultas SQL a distintos datastores
- Representación unificada de resultados mediante dataframes
- SQLContext

```
// To create DataFrame using SQLContext
val people = sqlContext.read.parquet("...")
val department = sqlContext.read.parquet("...")

people.filter("age > 30")
  .join(department, people("deptId") === department("id"))
  .groupBy(department("name"), "gender")
  .agg(avg(people("salary")), max(people("age")))
```

Hello Spark(SQL)

```
scala> val df = sqlContext.read.json(
 "examples/src/main/resources/people.json")
df: org.apache.spark.sql.DataFrame = [age: bigint, name: string]

scala> df.show
+----+-------+
| age|   name|
+----+-------+
|null|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+


scala>
```

Hello Spark(SQL)

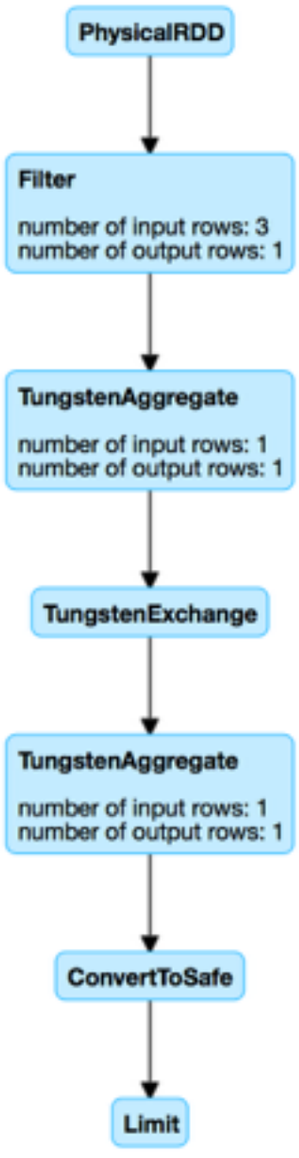
```
scala> df.printSchema()
root
 |-- age: long (nullable = true)
 |-- name: string (nullable = true)

scala> df.select("name").show()
+-------+
|   name|
+-------+
|Michael|
|   Andy|
| Justin|
+-------+
```

Hello Spark(SQL)

```
scala> df.filter(df("age") > 21).show()
+---+----+
|age|name|
+---+----+
| 30|Andy|
+---+----+


scala> df.groupBy("age").count().show()
+----+-----+
| age|count|
+----+-----+
|null|    1|
|  19|    1|
|  30|    1|
+----+-----+
```

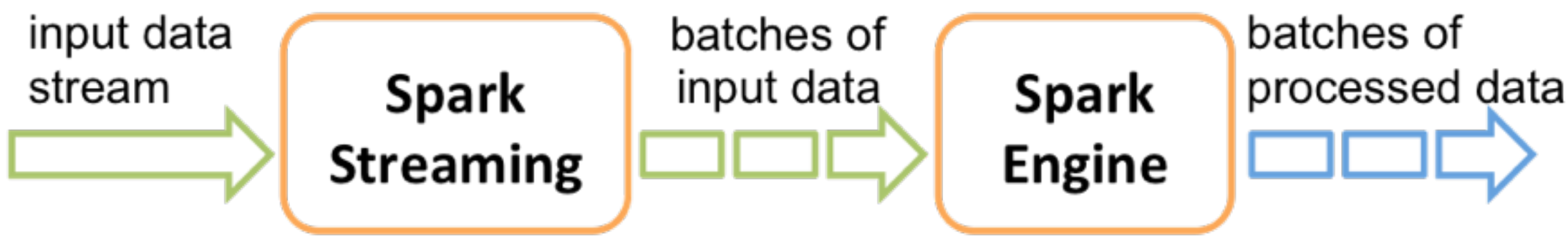
Catalyst in action

```
== Parsed Logical Plan ==
Limit 21
 Aggreg
  Filte
   Rela

== Anal
age: b
Limit 2
 Aggreg
  Filte
   Rela

scala> df.filter(df("age") > 21).groupBy("age").count.show
+---+-----+
|age|count|
+---+-----+
| 30|    1|
+---+-----+

== Optimized Logical Plan ==
Limit 21
 Aggregate [age#0L], [age#0L,count(1) AS count#8L]
  Project [age#0L]
   Filter (age#0L > 21)
    Relation[age#0L,name#1] JSONRelation[file:/.../examples/src/main/resources/people.json]

== Physical Plan ==
Limit 21
 ConvertToSafe
  TungstenAggregate(key=[age#0L], functions=[(count(1),mode=Final,isDistinct=false)], output=[age#0L,count#8L])
   TungstenExchange hashpartitioning(age#0L)
    TungstenAggregate(key=[age#0L],
        functions=[(count(1),mode=Partial,isDistinct=false)], output=[age#0L,currentCount#11L])
     Filter (age#0L > 21)
      Scan JSONRelation[file:/.../examples/src/main/resources/people.json][age#0L]

Code Generation: true
```



Spark Streaming

- Permite transformar una fuente en streaming en un conjunto de mini-batch
- Definición de una ventana



MLlib

- Librería para Machine Learning
  - Simplifica el desarrollo de pipelines de machine learning
- Abstracciones útiles para cómputo
  - Vectores, Matrices dispersas, etc.
- Implementación de algoritmos conocidos
  - Clasificación, regresión, collaborative filtering, clustering, etc.

```
val clusters = KMeans.train(data, 5, numIterations = 20)
```

GraphX

- API especializada para el procesado de grafos
- Estructuras para el almacenado y procesado específicas
- Integrado en el ecosistema
  - Representación de un grafo como tabla de vértices y de aristas

```
SELECT src.Id, dst.Id, src.attr, e.attr, dst.attr
FROM edges AS e
JOIN vertices AS src,
 vertices AS dst
 ON e.srcId = src.Id AND e.dstId = dst.Id
```

Errores Comunes

Errores más comunes

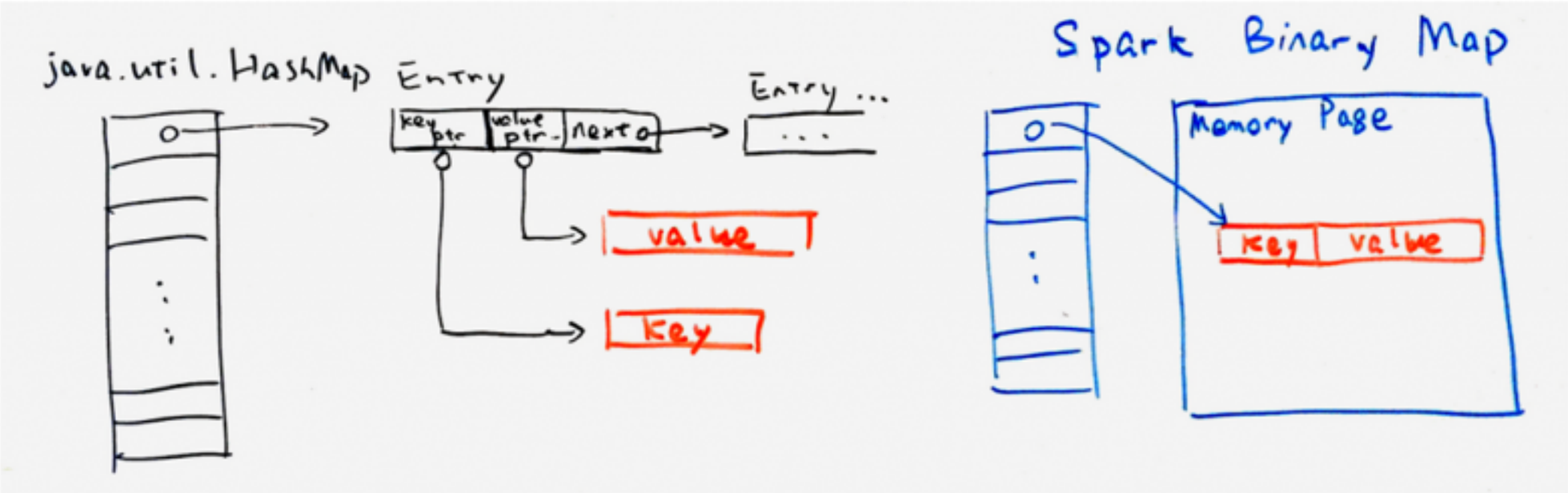
- URL del master
- No distribuir los JAR entre los workers
- Funciones con clases no serializables
- Funciona en local => funciona en distribuido
- Memory leaks y eficiencia GC en operadores
- Confusión operadores (reduce vs group-by)

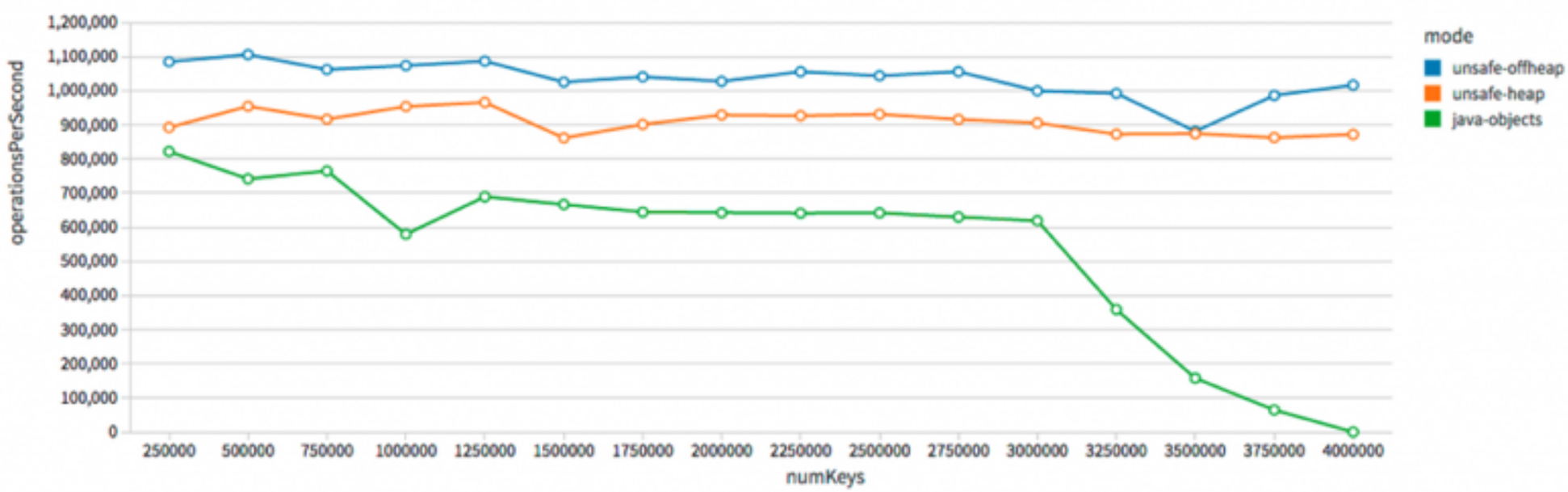
Errores comunes

```
var counter = 0
var rdd = sc.parallelize(data)

// Wrong: Don't do this!!
rdd.foreach(x => counter += x)

println("Counter value: " + counter)
```

Tungsten

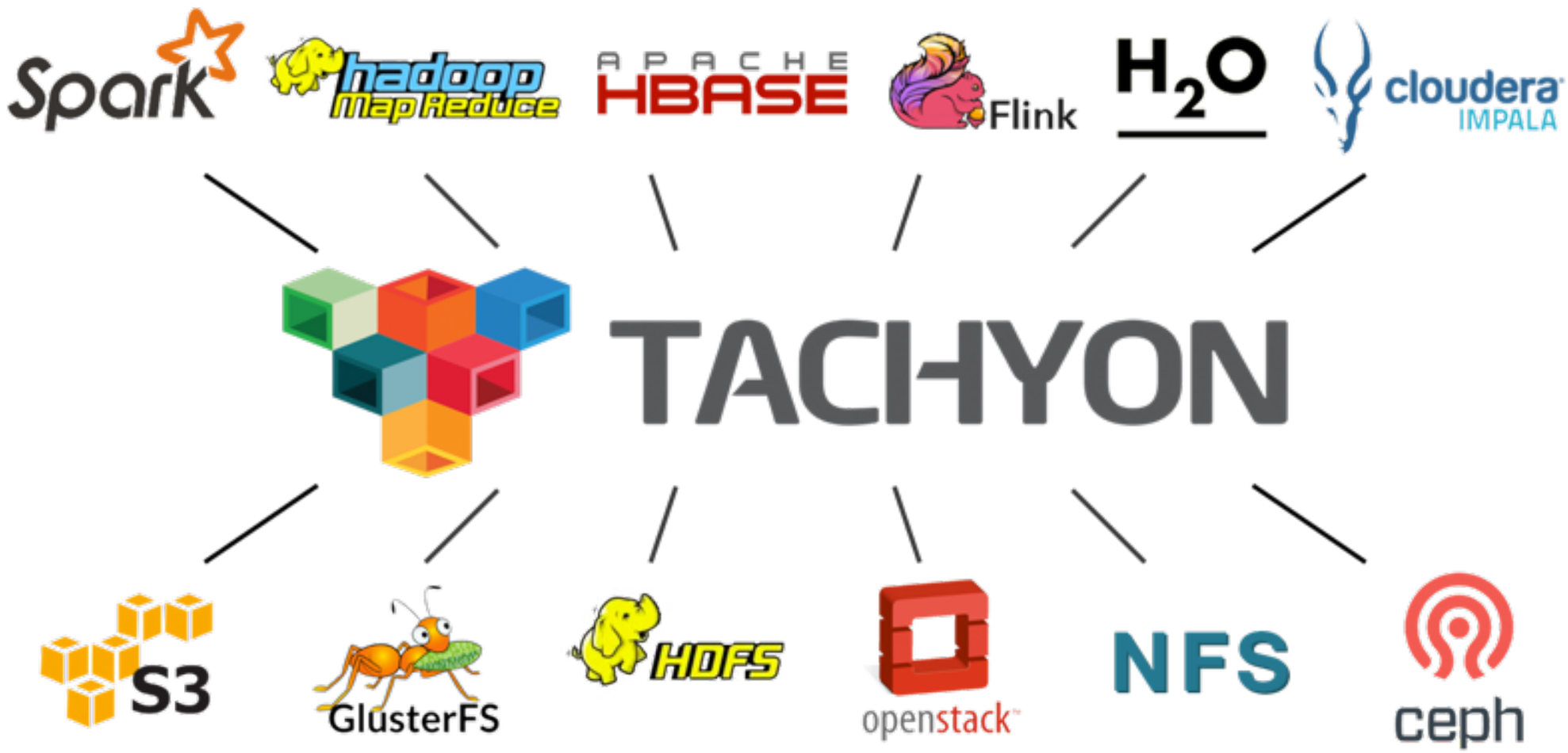
Project Tungsten





Tachyon

Project Tungsten



Entendiendo la ejecución

Consola Web

http://localhost:4040/jobs/

Spark 1.5.0 | Jobs | Stages | Storage | Environment | Executors | SQL | Spark shell application UI

Spark Jobs (?)

Total Uptime: 1.5 h
Scheduling Mode: FIFO
Completed Jobs: 13

▸ Event Timeline

Completed Jobs (13)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
12	show at <console>:20	2015/09/19 09:55:47	9 ms	1/1	1/1
11	show at <console>:20	2015/09/19 09:55:47	30 ms	1/1	1/1
10	show at <console>:22	2015/09/19 09:46:59	0.5 s	1/1 (1 skipped)	199/199 (2 skipped)
9	show at <console>:22	2015/09/19 09:46:59	0.1 s	2/2	3/3
8	show at <console>:22	2015/09/19 09:42:43	0.7 s	1/1 (1 skipped)	199/199 (2 skipped)
7	show at <console>:22	2015/09/19 09:42:43	0.4 s	2/2	3/3
6	show at <console>:22	2015/09/19 09:42:33	12 ms	1/1	1/1
5	show at <console>:22	2015/09/19 09:42:33	0.2 s	1/1	1/1

Introducción a Spark

Daniel Higuero (daniel.higuero@gmail.com)

@dhiguero