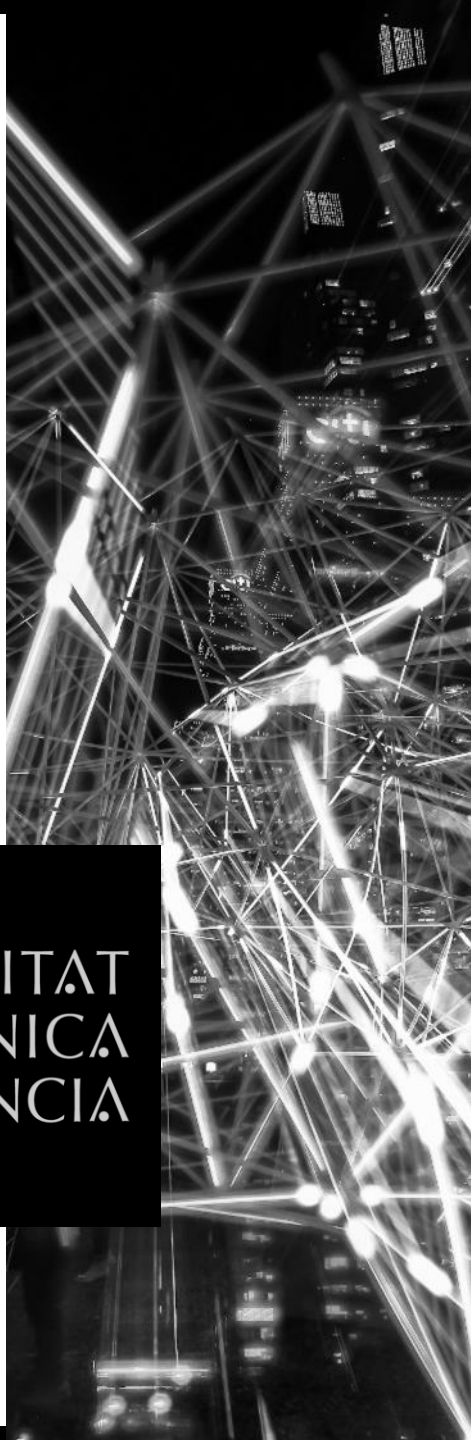
INTRODUCCIÓN A APACHE SPARK

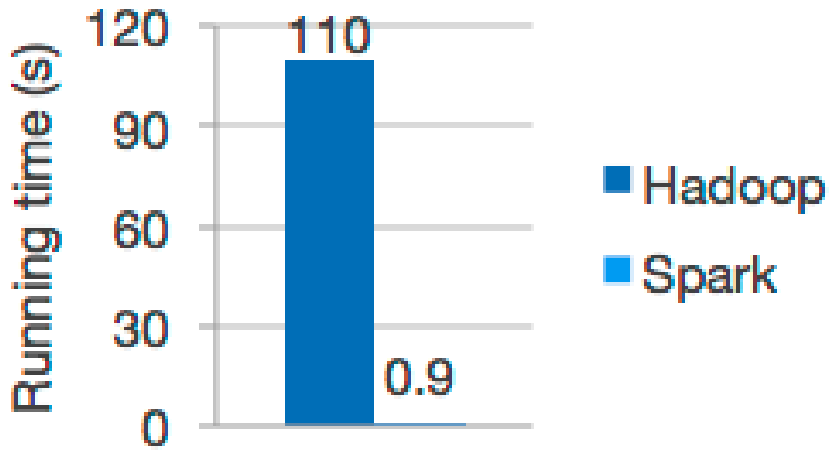


UNIVERSITAT POLITÈCNICA DE VALÈNCIA

HADOOP ES INVENCIBLE (?)

- Hadoop tuvo un gran impacto sobre las infraestructuras de procesado de datos
- Permitió a muchas organizaciones resolver problemas que hasta ese momento eran muy costosos a nivel técnico
- Permitió la popularización de las infraestructuras de Big Data, así como introdujo la primera ola de este tipo de tecnologías en las empresas
- La gente pensaba que Hadoop era todo lo que se necesitaba para afrontar la era del procesado de Big Data

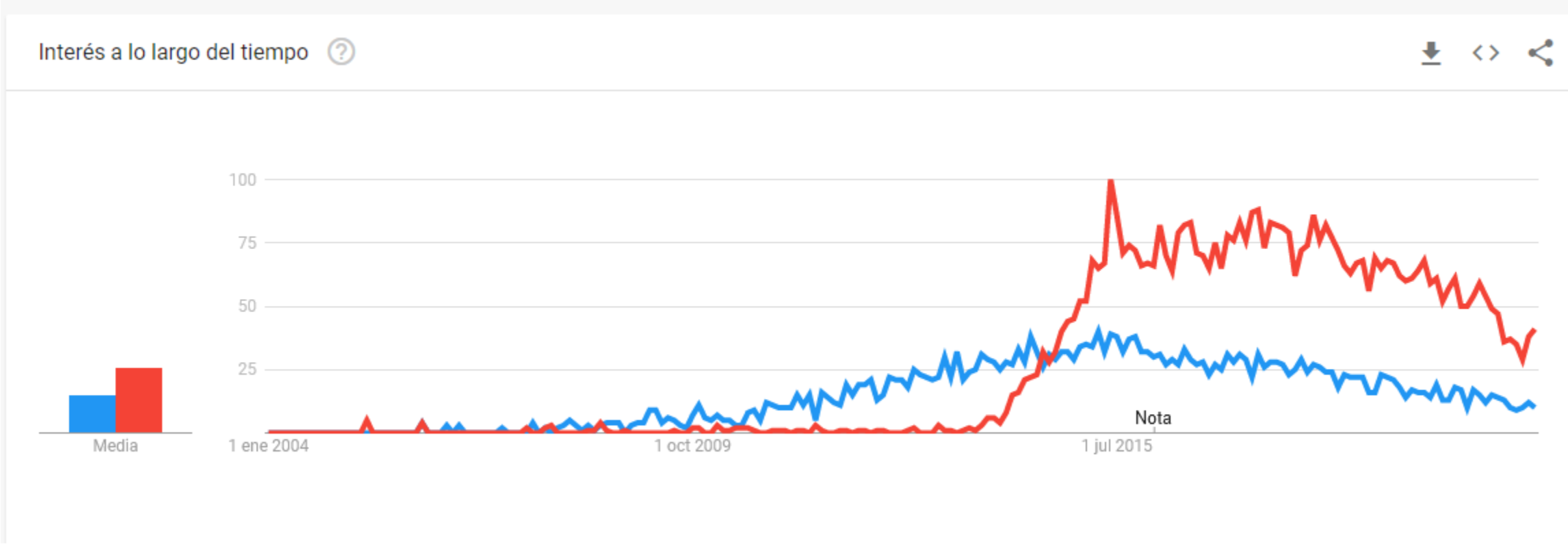
HADOOP ES INVENCIBLE (?)



Logistic regression in Hadoop and Spark

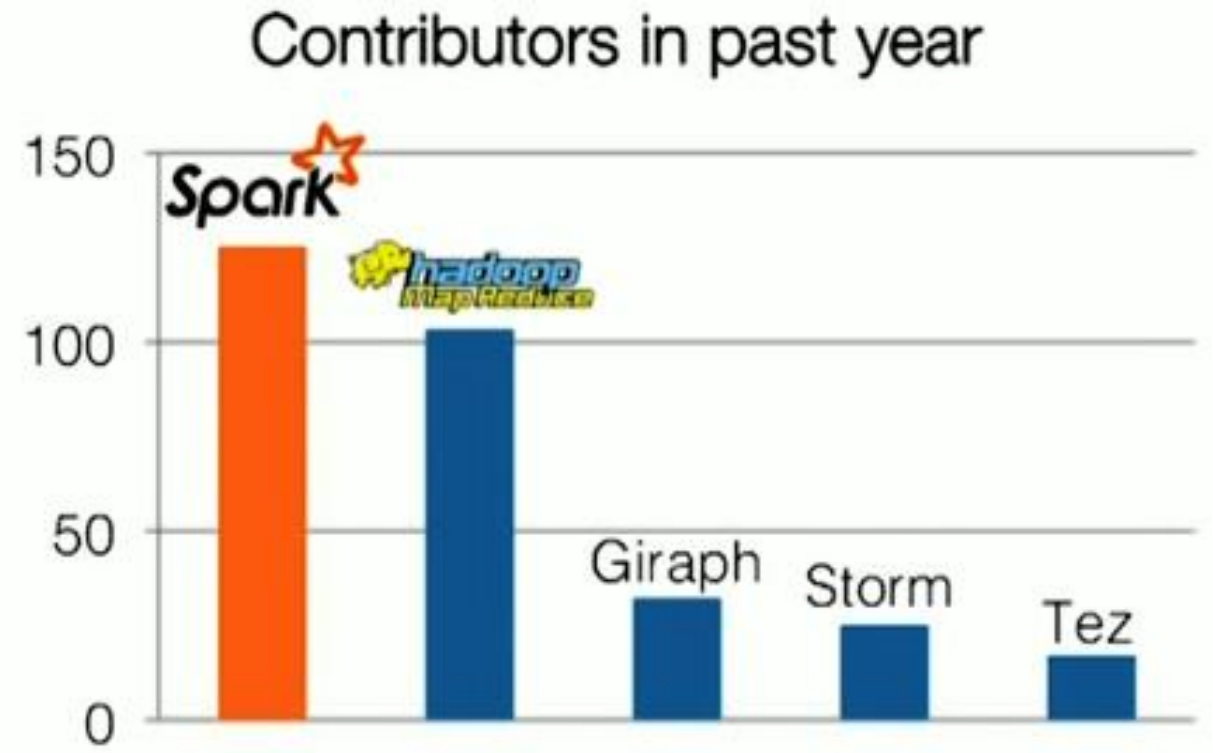
https://spark.apache.org

HADOOP ES INVENCIBLE (?)



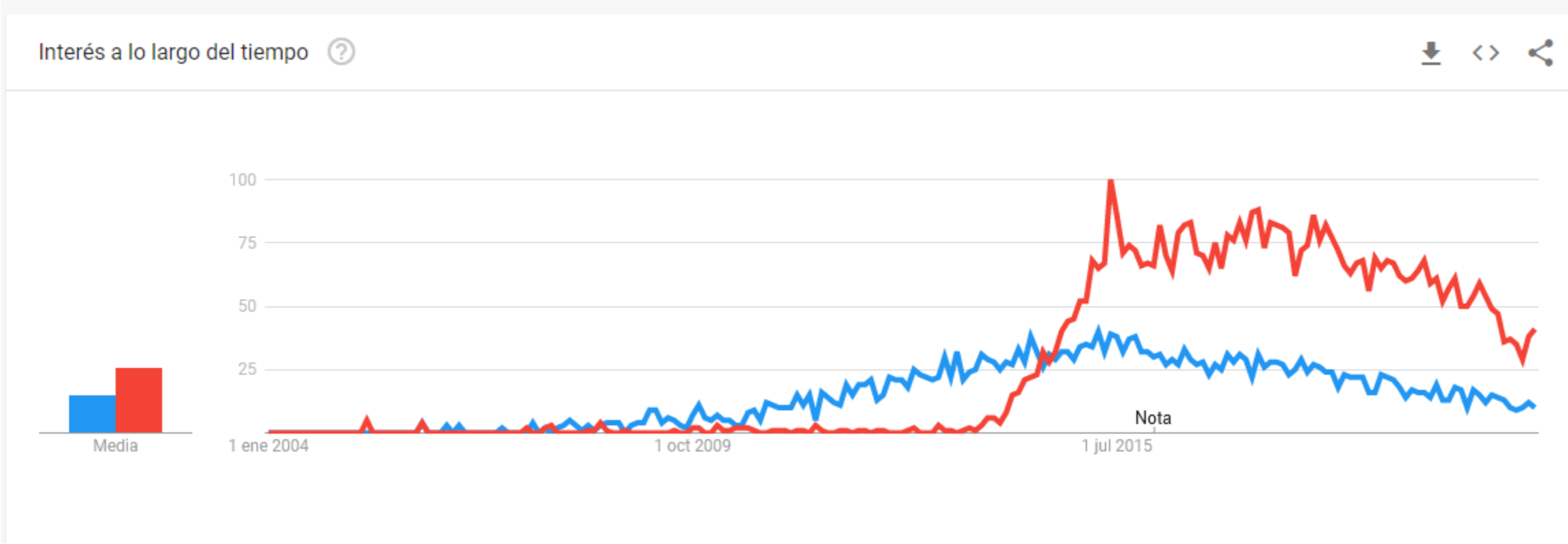
Fuente: Google Trends

HADOOP ES INVENCIBLE (?)



http://www.datanami.com/2014/03/06/apache_spark_3_real-world_use_cases/

HADOOP ES INVENCIBLE (?)



Fuente: Google Trends

APACHE SPARK

¿Qué es?

APACHE SPARK

FRAMEWORK DE PROCESADO DE GRANDES VOLÚMENES DE DATOS

- Es un framework para el procesamiento de grandes volúmenes de datos
- Creado en 2009 en la UC Berkeley, dentro del AMPLab como un proyecto de investigación
- Principales características:
  - Trabajo con colecciones distribuidas de objetos
  - Distribución de datos en memoria eficiente, en vez de distribución en disco
- Última version: Spark 3.1.1 (Marzo 2021)

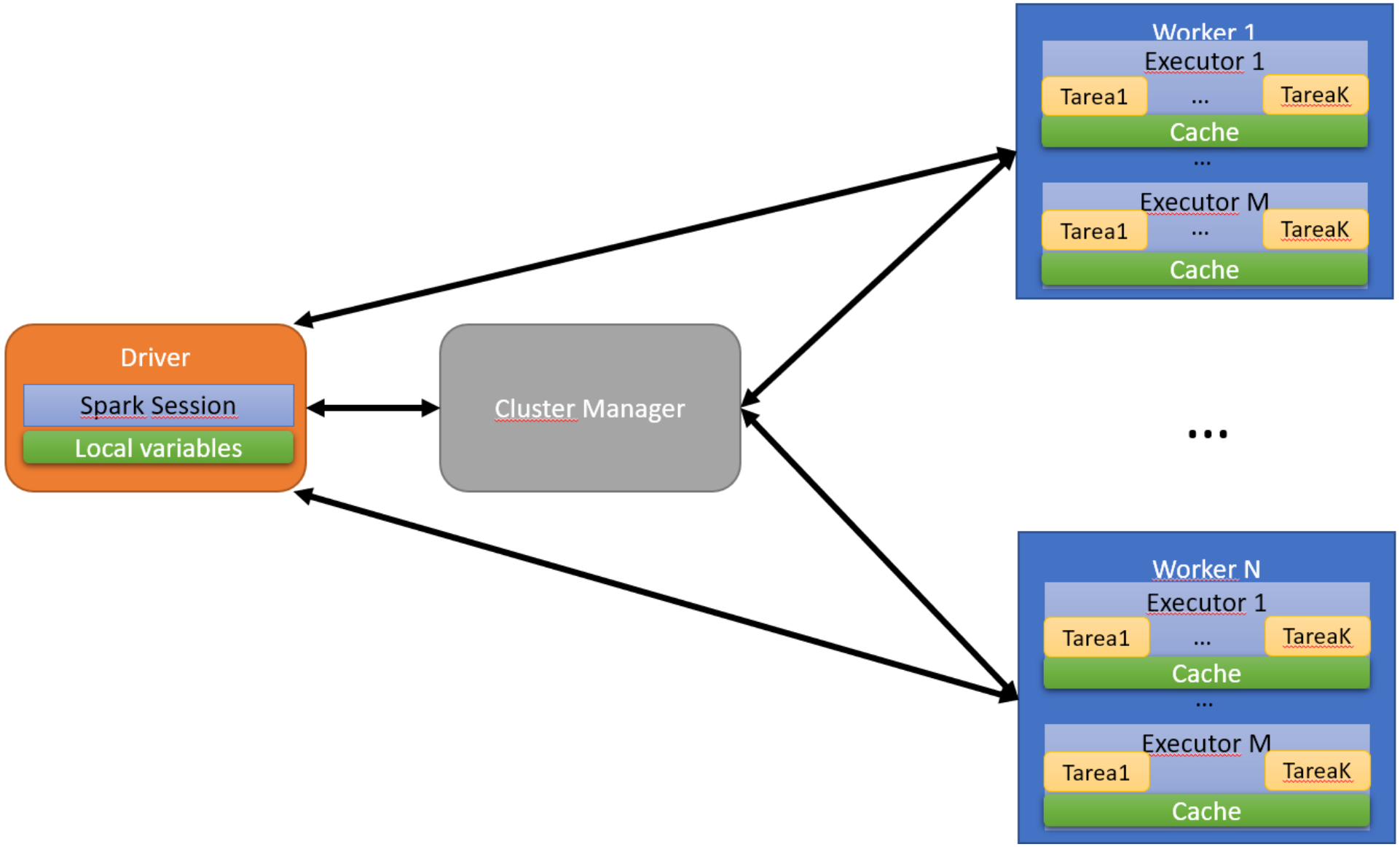
https://spark.apache.org

DISTRIBUCIÓN EN MEMORIA PRINCIPAL

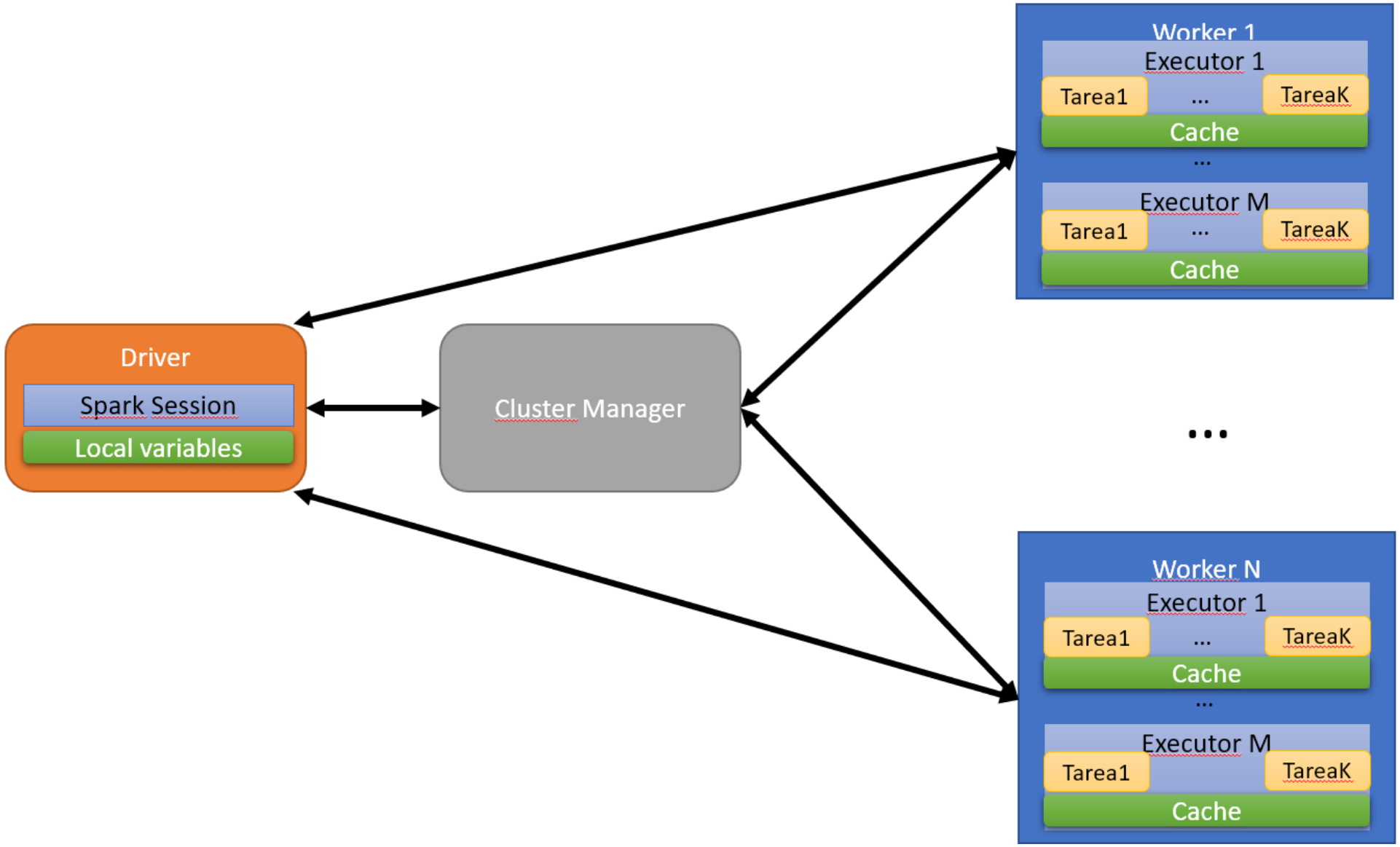
¿Por qué?

- En Spark los datos son almacenados (temporalmente) y cacheados en **memoria principal** (RAM)
- ¿Esto importa?
  - Velocidad escritura SSD más rápido ~ 3200MB/s
  - Velocidad escritura RAM más rápida ~ 48GB/s
- Desventajas memoria principal
  - **Coste de fabricación** → Ha ido descendiendo
  - **Volátil** → Almacenamiento de datos temporales
- ¿Donde es interesante almacenar de forma temporal en memoria? → Cálculo **iterativo** sobre un mismo conjunto de datos

ARQUITECTURA DE APACHE SPARK

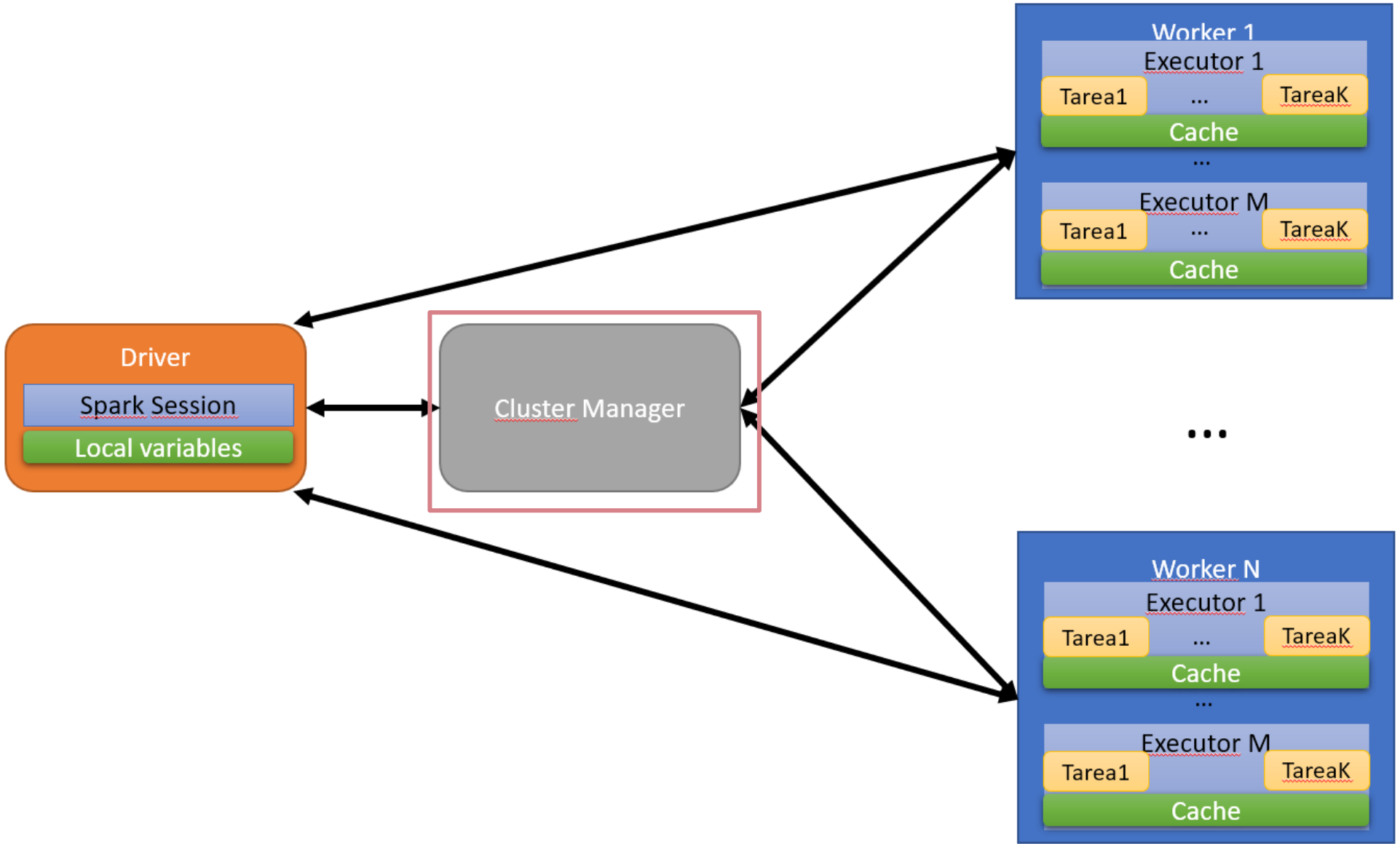


ARQUITECTURA DE APACHE SPARK



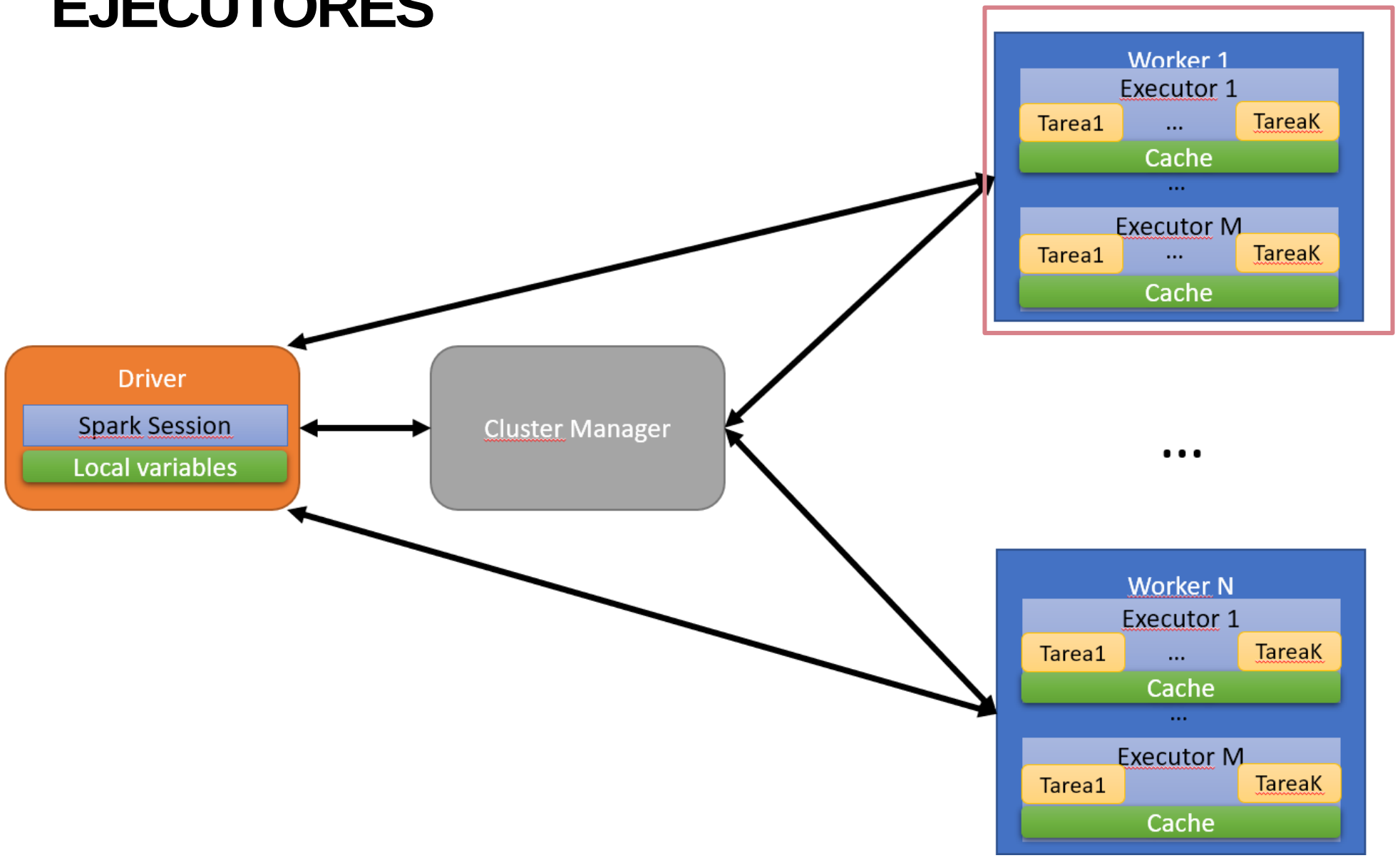
- Spark puede trabajar con diversos tipos de gestores de clúster (e.g., YARN, Mesos, propio)
- Sigue una arquitectura de tipo maestro-esclavo (driver y workers)

ARQUITECTURA DE APACHE SPARK: MANAGER



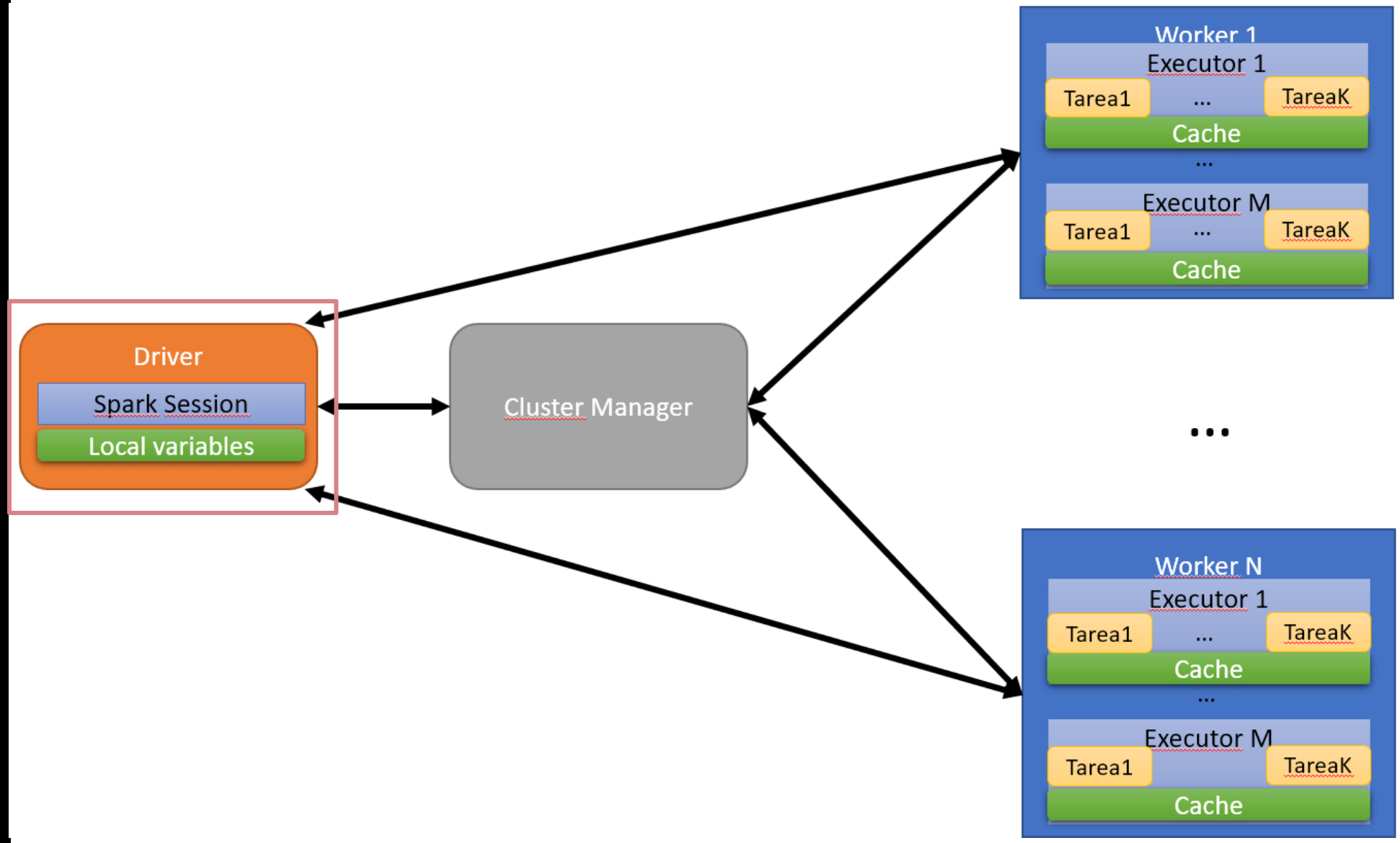
- Conoce donde se encuentra cada uno de los nodos trabajadores. Se ejecuta en un nodo
- Coordina y envía trabajos a los trabajadores
- Gestiona los recursos puestos a disposición por los trabajadores
- Tipos:
  - Standalone
  - Mesos
  - YARN
  - Kubernetes

ARQUITECTURA DE APACHE SPARK: WORKERS Y EJECUTORES



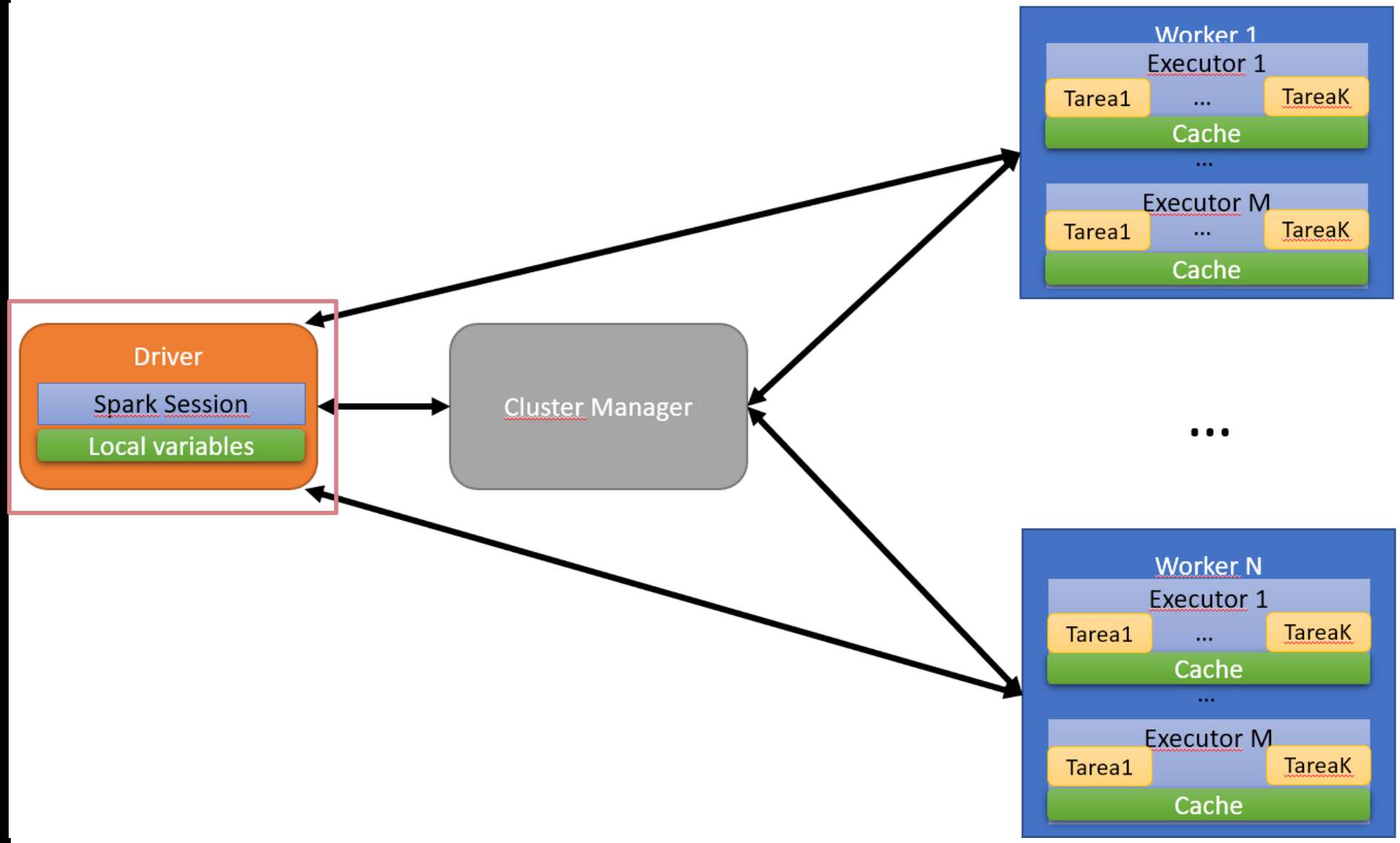
- Un worker es un nodo físico destinado al procesamiento de trabajos y datos dentro de la infraestructura de Spark
- Las aplicaciones de Spark crean Ejecutores → Procesos en diferentes JVMs que se encargan de ejecutar tareas para las aplicaciones. Tienen recursos asignados
- Pueden haber muchos workers, y muchos ejecutores dentro de un worker
- Son coordinados por el driver y envían mensajes recurrentes en el tiempo sobre diagnóstico tareas

ARQUITECTURA DE APACHE SPARK: DRIVER



- Actúa como maestro en la arquitectura
- Es un proceso que recibe la aplicación Spark del usuario y que se ejecuta en una JVM
- Funciones:
  - Recoge programa del usuario
  - Crea un objeto SparkContext para comunicación con el clúster
  - Negocia recursos con el manager
  - El programa es analizado y descompuesto en tareas locales y tareas distribuidas

ARQUITECTURA DE APACHE SPARK: DRIVER

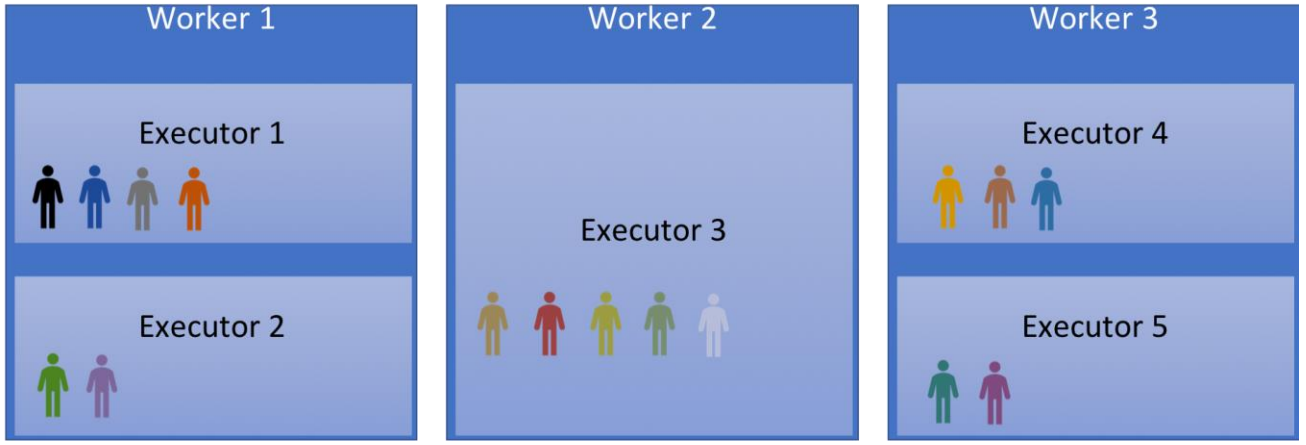


- Funciones (cont.):
  - Planificación de las tareas distribuidas entre los ejecutores
  - Envía las tareas distribuidas a los ejecutores y ejecuta las tareas locales
  - La aplicación de Spark está activa mientras lo esté el clúster y sus ejecutores

COLECCIONES DISTRIBUIDAS DE OBJETOS

ABSTRACCIÓN DE DATOS

- En Spark, los datos son distribuidos mediante la abstracción de colección distribuida de objetos, conocida como *Resilient Distributed Dataset* o RDD
- Un objeto es cualquier valor representable por el lenguaje de programación (e.g., string, números, imágenes, object, etc.)
- Un RDD puede estar compuesto por un gran número de objetos. Cada ejecutor típicamente albergará una parte de los objetos de dicho RDD



COLECCIONES DISTRIBUIDAS DE OBJETOS

CARACTERÍSTICAS RDDs

- Son **tolerantes a fallos** → Cuando se pierde un ejecutor, el driver almacena un plan de tareas para recuperar los objetos que se han perdido
- Un RDD reside en **memoria principal** siempre que sea posible. Cuando son demasiado grandes, parte del RDD puede ser **serializado** a disco
- Son **inmutables**
- Son evaluados de forma **perezosa**
- Se opera con ellos en **paralelo**

PROGRAMACIÓN CON SPARK

¿Cómo trabajamos con Spark?

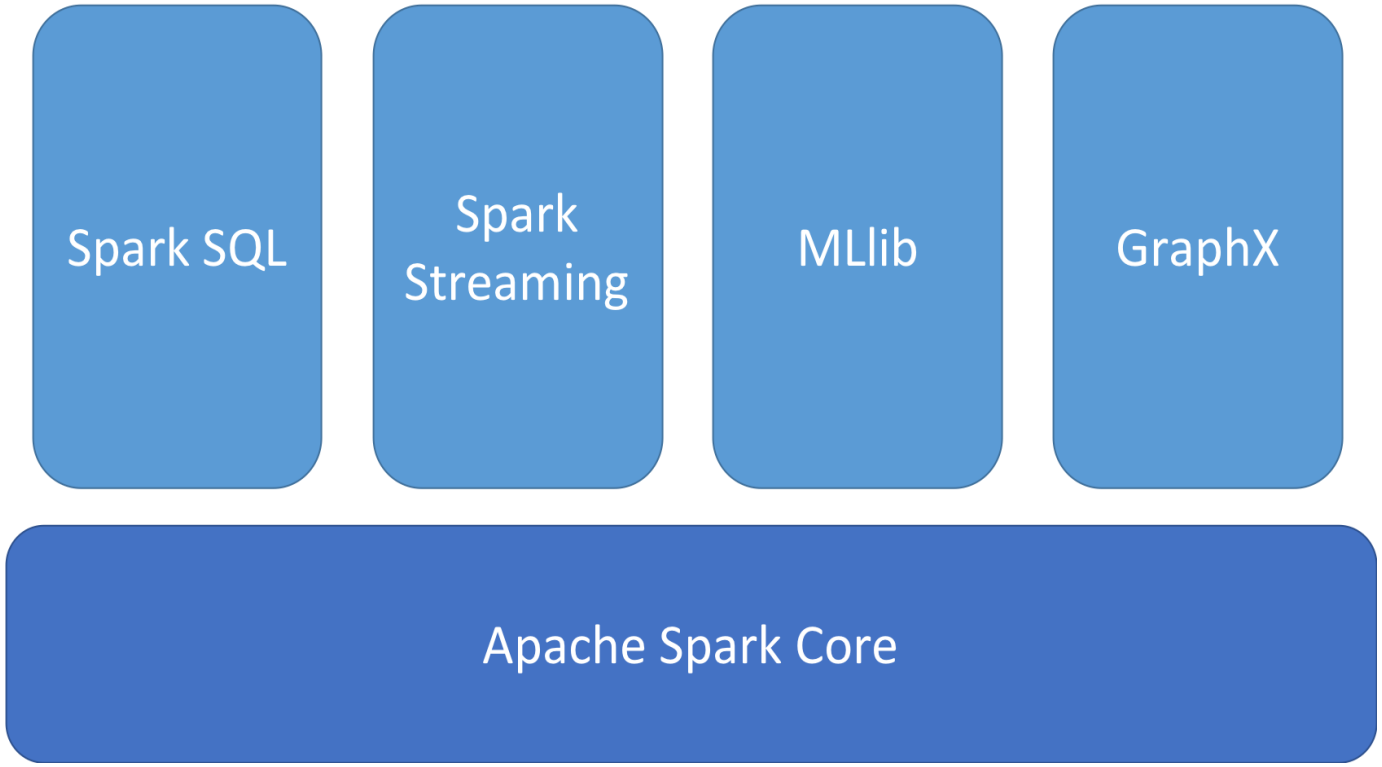
APLICACIONES SPARK

APIs generales disponibles

- Una aplicación en Spark es una colección de instrucciones locales e instrucciones de trabajo sobre RDDs (paralelas)
- La aplicación es recibida por el driver, analizada, y ejecutada entre el driver y ejecutores
- Para trabajar con el núcleo de Spark podemos emplear:
  - Java: Spark está programado en este lenguaje
  - Scala: Lenguaje de scripting compatible con la JVM y Java
  - Python
  - R
  - Otros addons nos oficiales para trabajar con JS, C# o Julia

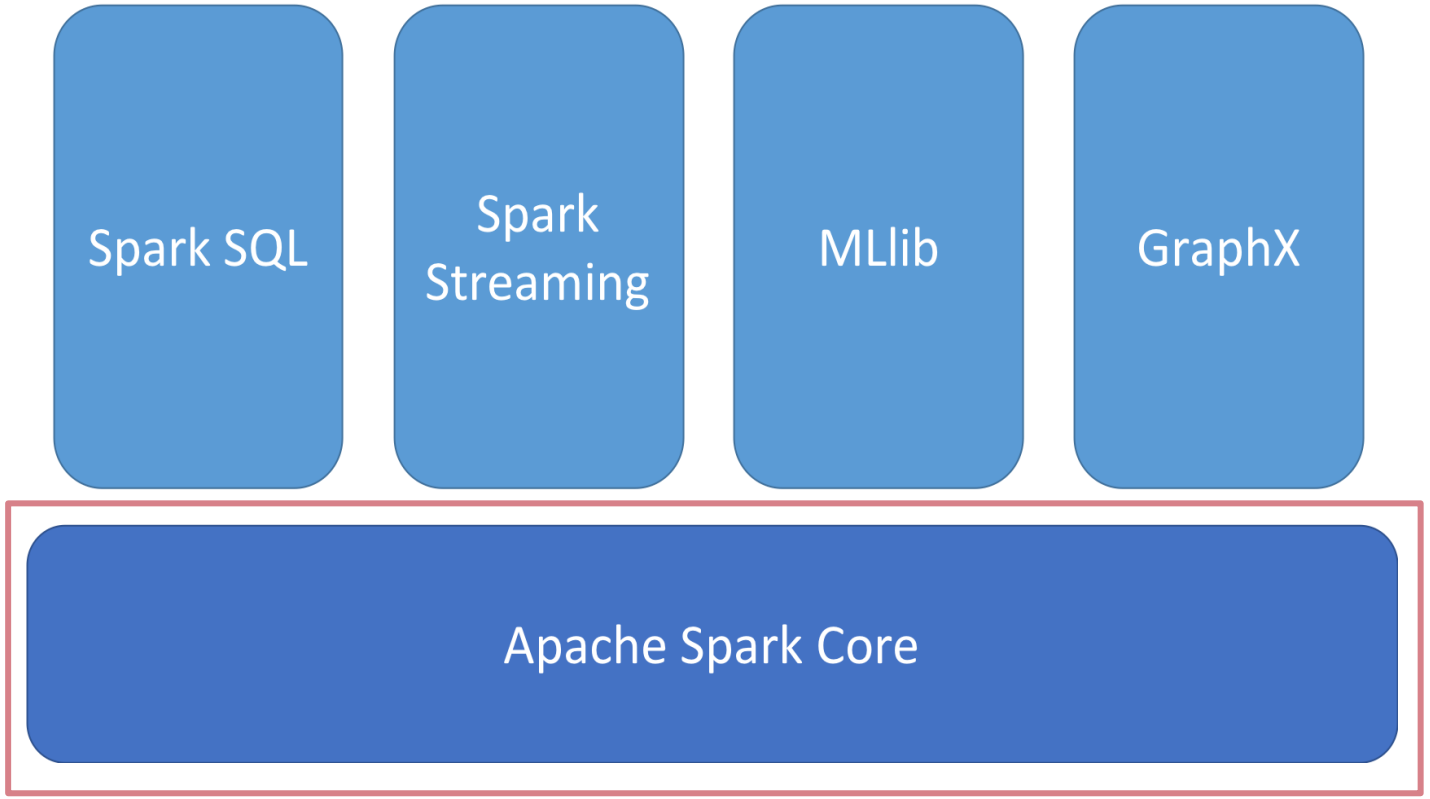
APIS DISPONIBLES EN SPARK

APIs generales disponibles



APIS DISPONIBLES EN SPARK

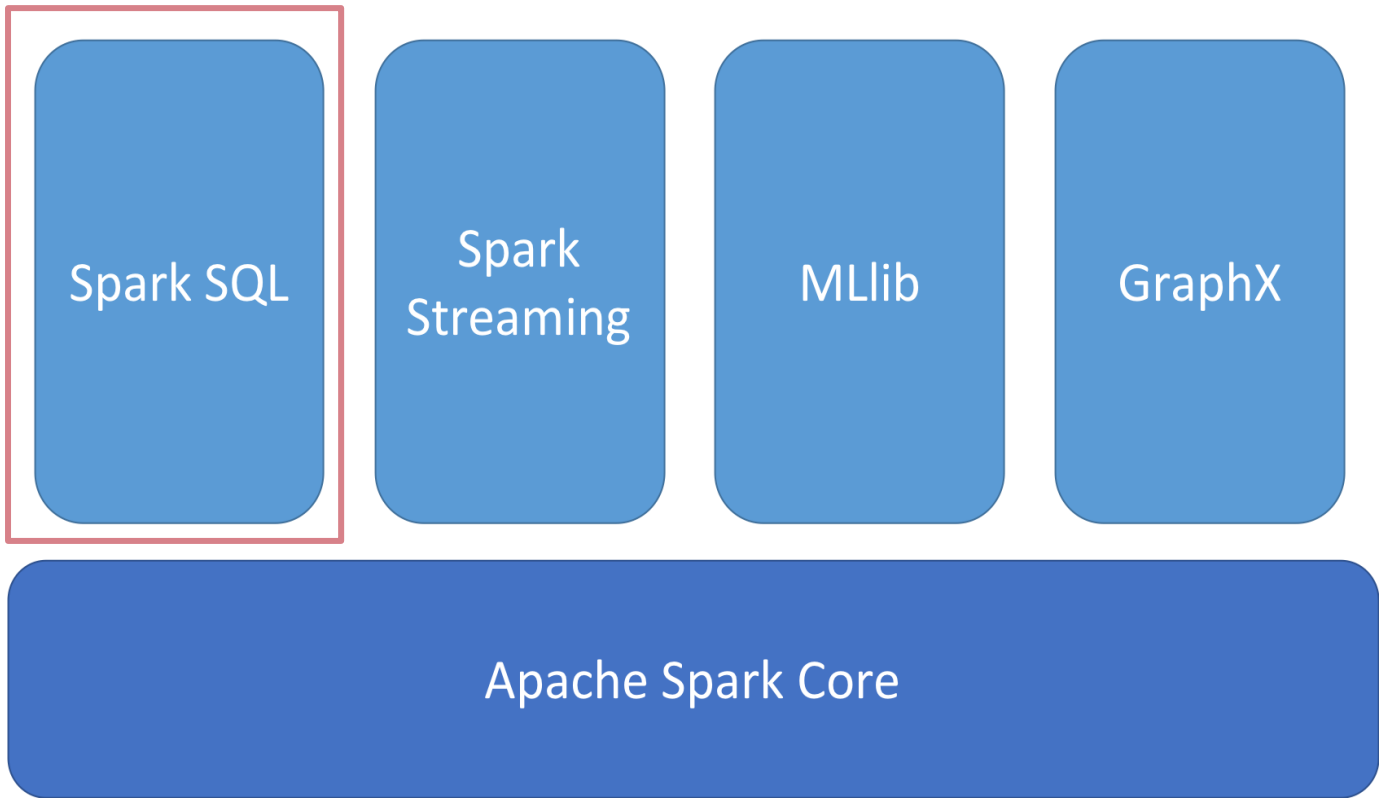
APIs generales disponibles



- Es la API de bajo nivel y más básica de interacción directa con los RDDs.
- Nos permite trabajar leer, cargar, y transformar RDDs.
- Se usa principalmente para trabajar con datos no estructurados o semiestructurados
- Disponible en Java, Scala y Python

APIS DISPONIBLES EN SPARK

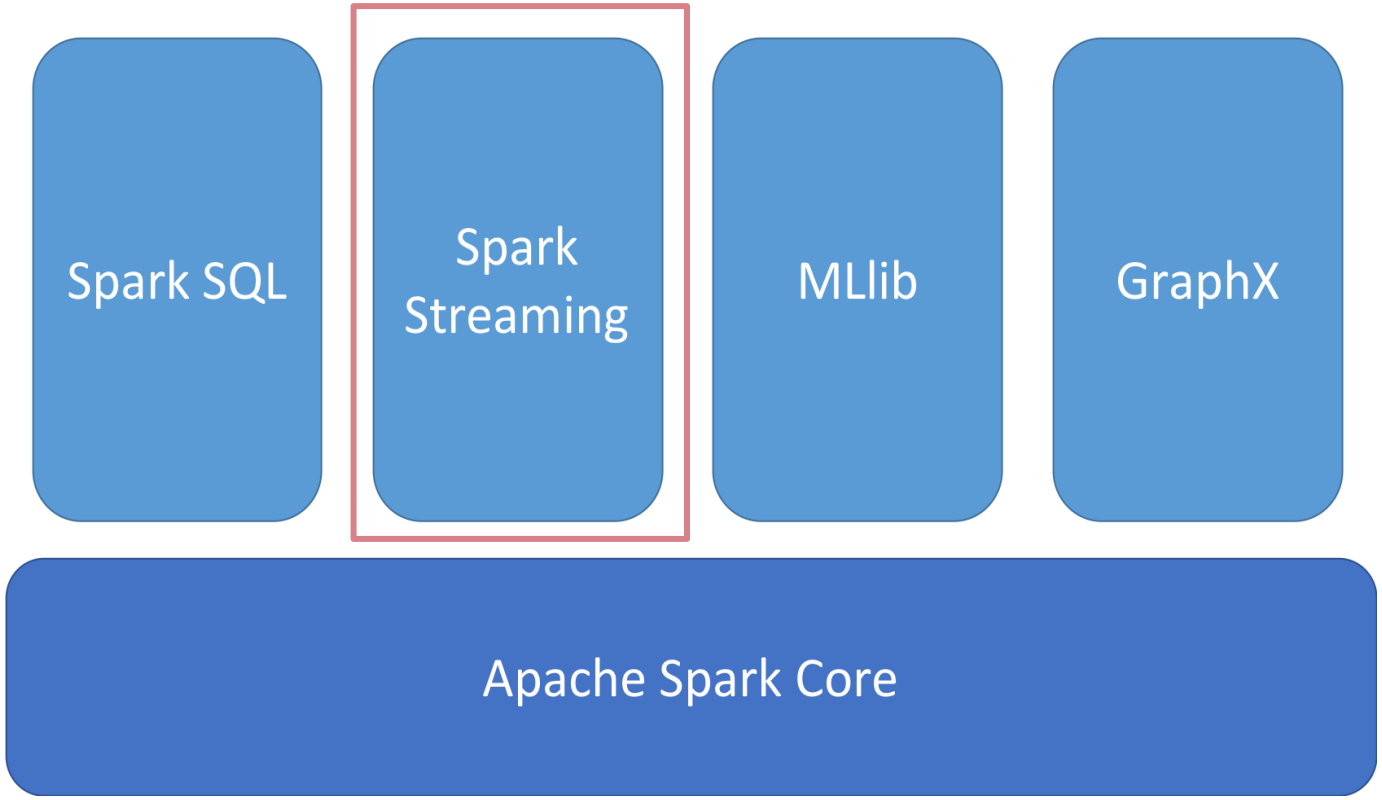
APIs generales disponibles



- Realmente hace referencia a la API para trabajar con datos estructurados
- Incluye tanto la posibilidad de trabajar con DataFrames como con consultas SQL
- Disponible en Java, Scala, Python y R

APIS DISPONIBLES EN SPARK

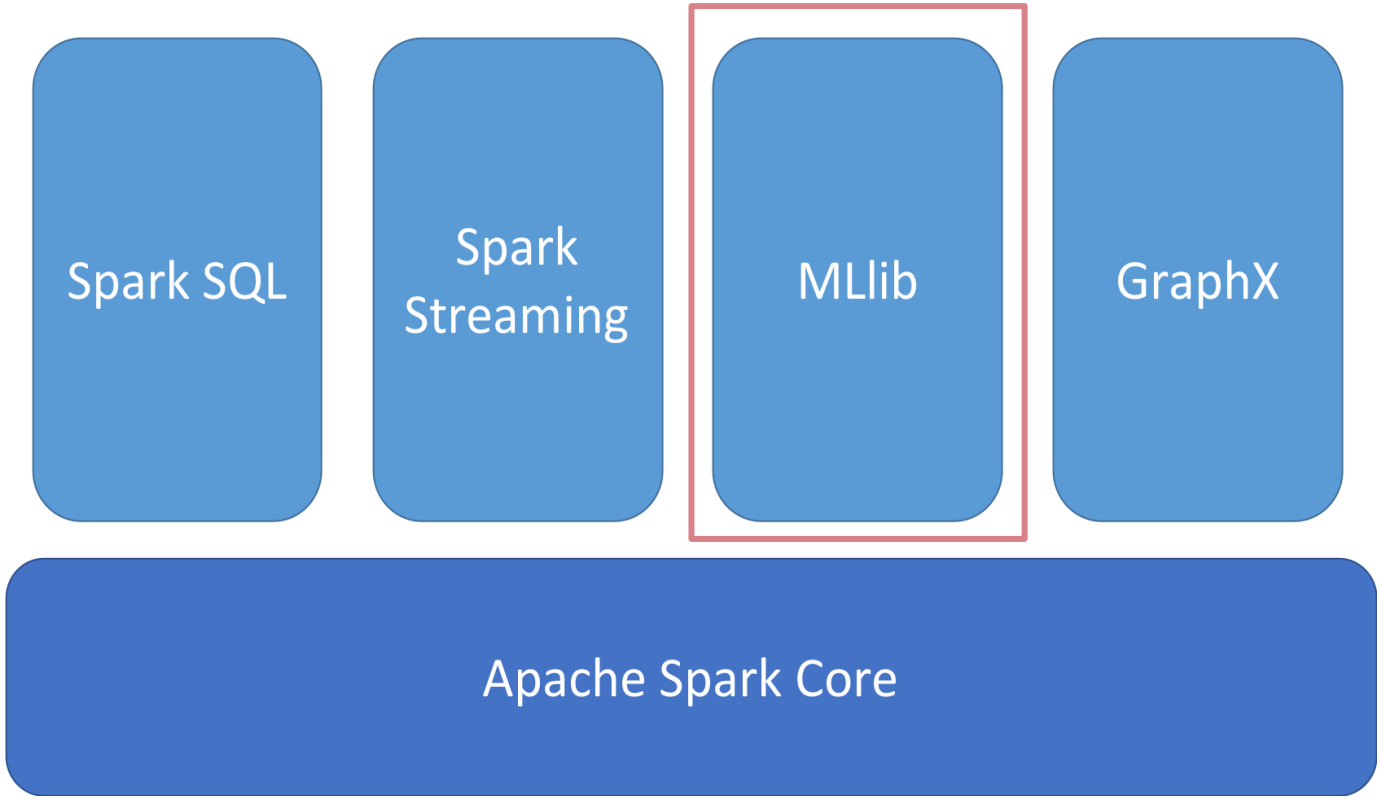
APIs generales disponibles



- Hay conjuntos de datos que no son obtenidos de golpe, sino que pequeños paquetes se van obteniendo a lo largo del tiempo
- El conjunto de datos que analizamos va cambiando en tiempo real
- Esta API se centra en el procesado de conjuntos de datos en tiempo real
- Se encuentra en Java, Scala y Python

APIS DISPONIBLES EN SPARK

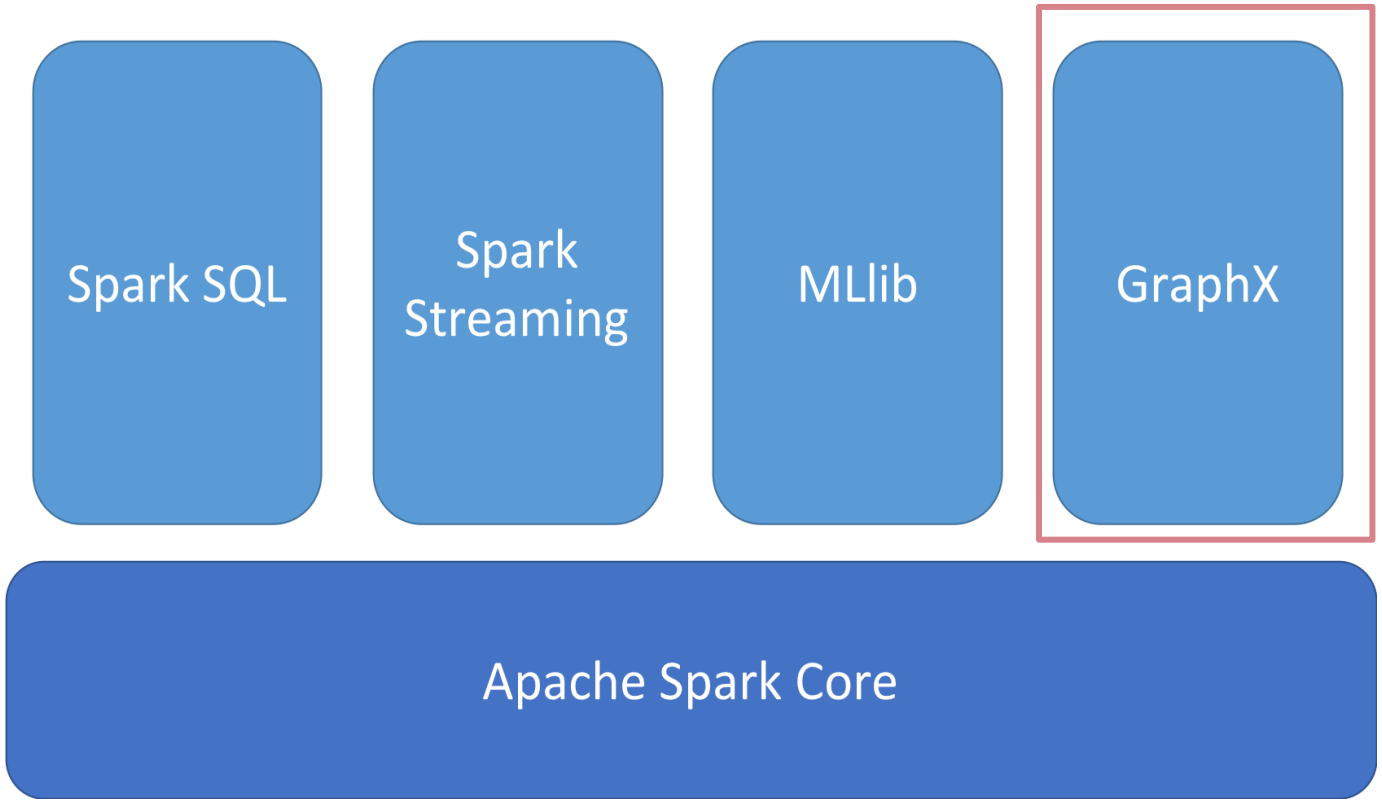
APIs generales disponibles



- MLLib es la API centrada en tareas de aprendizaje automático
- Permite la construcción de grandes modelos en base a colecciones muy grandes de datos
- Se encuentra en Java, Scala y Python

APIS DISPONIBLES EN SPARK

APIs generales disponibles



- Es una API especializada en el trabajo con grafos y en la ejecución de algoritmos para grafos
- Únicamente disponible en Scala
- Todavía se encuentra en alfa (lleva ya mucho tiempo así)

COP
UPV

CONTINUARÁ …

Víctor Sánchez Anguix ✉ vicsana1@upv.es

DSIC
DEPARTAMENT DE SISTEMES INFORMÀTICS I COMPUTACIÓ

DEIOAC
VALENCIA

DEPARTAMENTO DE ESTADÍSTICA E INV. OPERAT. APLICADAS Y CALIDAD

UNIVERSITAT POLITÈCNICA DE VALÈNCIA