APACHE SPARK
PART I

QUE ES APACHE SPARK

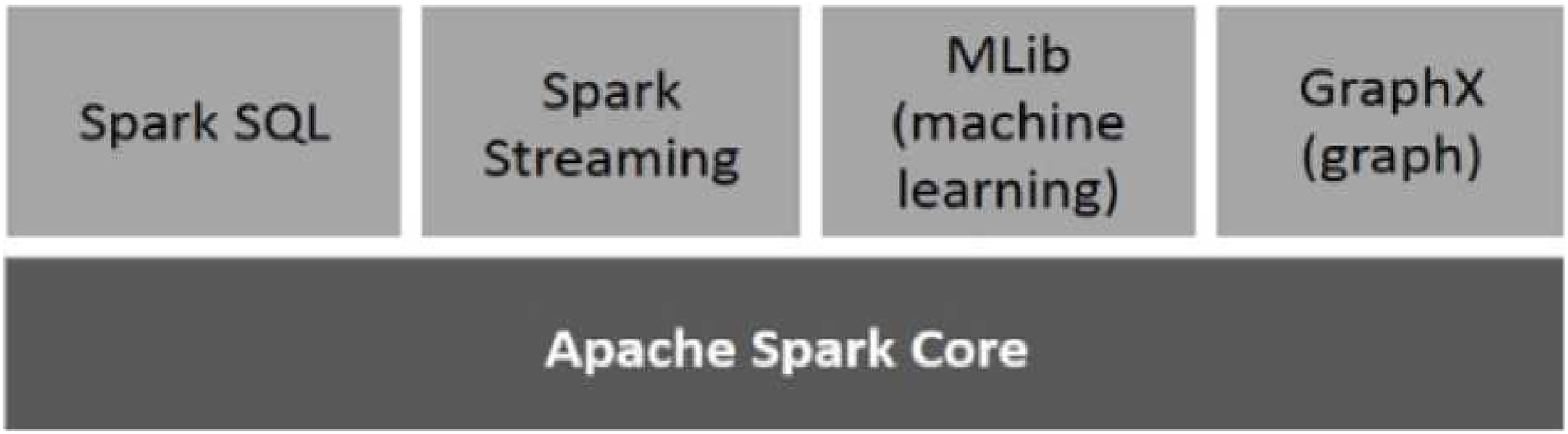
- Spark es una plataforma de computacion para clusters
- Es de proposito general.
- Desarrollo simplificado
- Trabaja en memoria
- Rapido
- Permite trabajo interactive, streaming..

ARQUITECTURA SPARK

Arquitectura

- Muy versatil. Puede trabajar:
  - Standalone.
  - Sobre la nube de Amazon
  - Sobre Hadoop
- Fuentes de datos:
  - Ficheros locales
  - HDFS
  - Cassandra
  - MongoDB
  - Hive
  - postgresQL, mySQL
  - S3 amazon
  -

COMPONENTES SPARK



SPARK CORE

Spark Core:

Es el corazón de sprk responsible de gestionar las funciones como la programacion de las tareas .

SPARK SQL

SPARK SQL :

Es un modulo ubicado en la parte superior del nucleo de Spark, que introduce una **nueva idea de datos llamada *SchemaRDD***, el cual proporciona soporte para datos estructurados y semi-estructurados.

Gracias a este componente, se permite **combinar consultas SQL con programas de Spark** y ademas, se permite la consulta de datos estructurados utilizando lenguaje SQL o con la API que nos ofrece Apache Spark.

SPARK STREAMING

Spark es capaz de realizar analisis de streaming sin problemas, gracias a la gran velocidad de programacion de su nucleo. Ademas, gracias a la API

Permite crear aplicaciones escalables e intolerantes a fallos de streaming.

Otra ventaja que posee Spark, es que es capaz de procesar grandes datos en tiempo real.

Los datos son analizados conforme entran, sin tiempo de latencia y a traves de un proceso de gestion en continuo transito.

SPARK MLIB

Es un framework de aprendizaje ubicado en la parte superior del nucleo de Spark, que tiene como finalidad hacer practico, escalable y facil el “machine learning”.

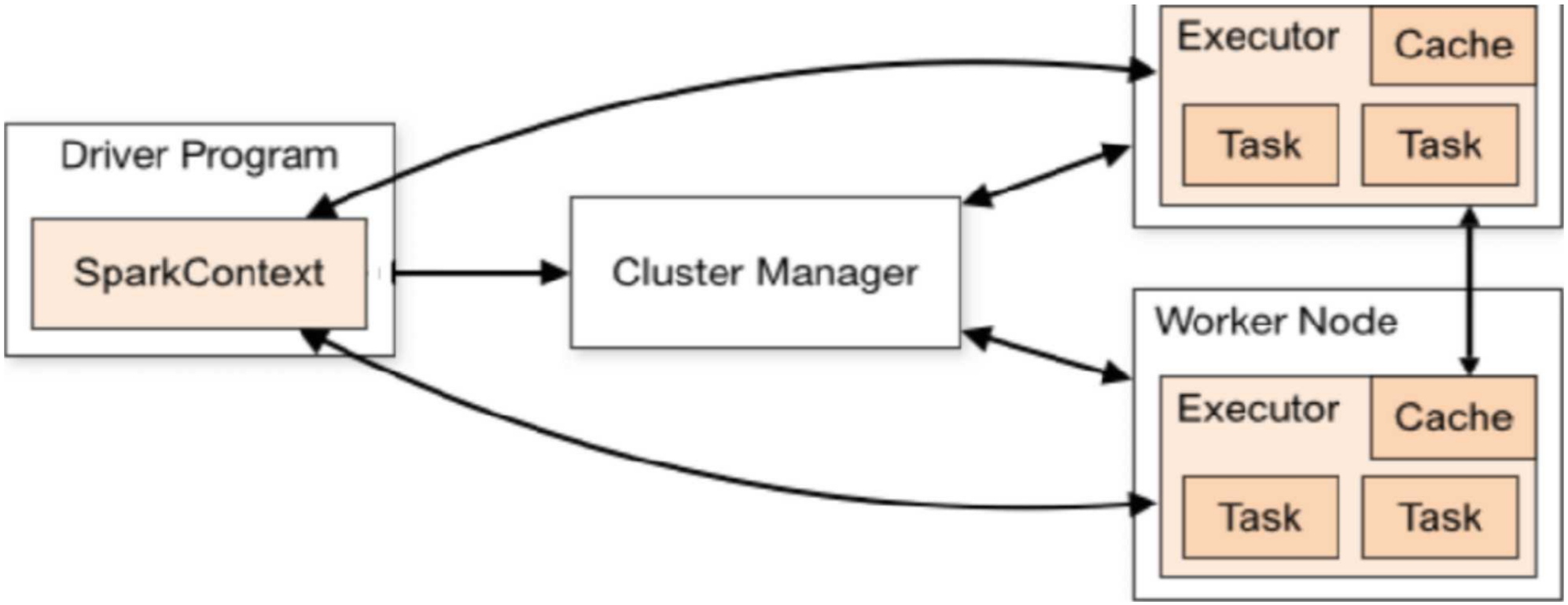
Este framework posee un conjunto de algoritmos y utilidades, como por ejemplo:

- Clasificacion,
- Regresion,
- Clustering,
- Filtrado colaborativo
- Reduccion de dimensionalidad

SPARK GRAPHX

Es una entorno de procesamiento grafico ubicado en la parte superior de Spark, el cual proporciona una API para graficos y calculo grafico en paralelo.

EJECUCIÓN DE SPARK MODO LOCAL



SPARK

VOCABULARIO CONCEPTOS:

1. Programa de usuario de aplicación basado en Spark. Consta de un programa de controlador y ejecutores en el clúster

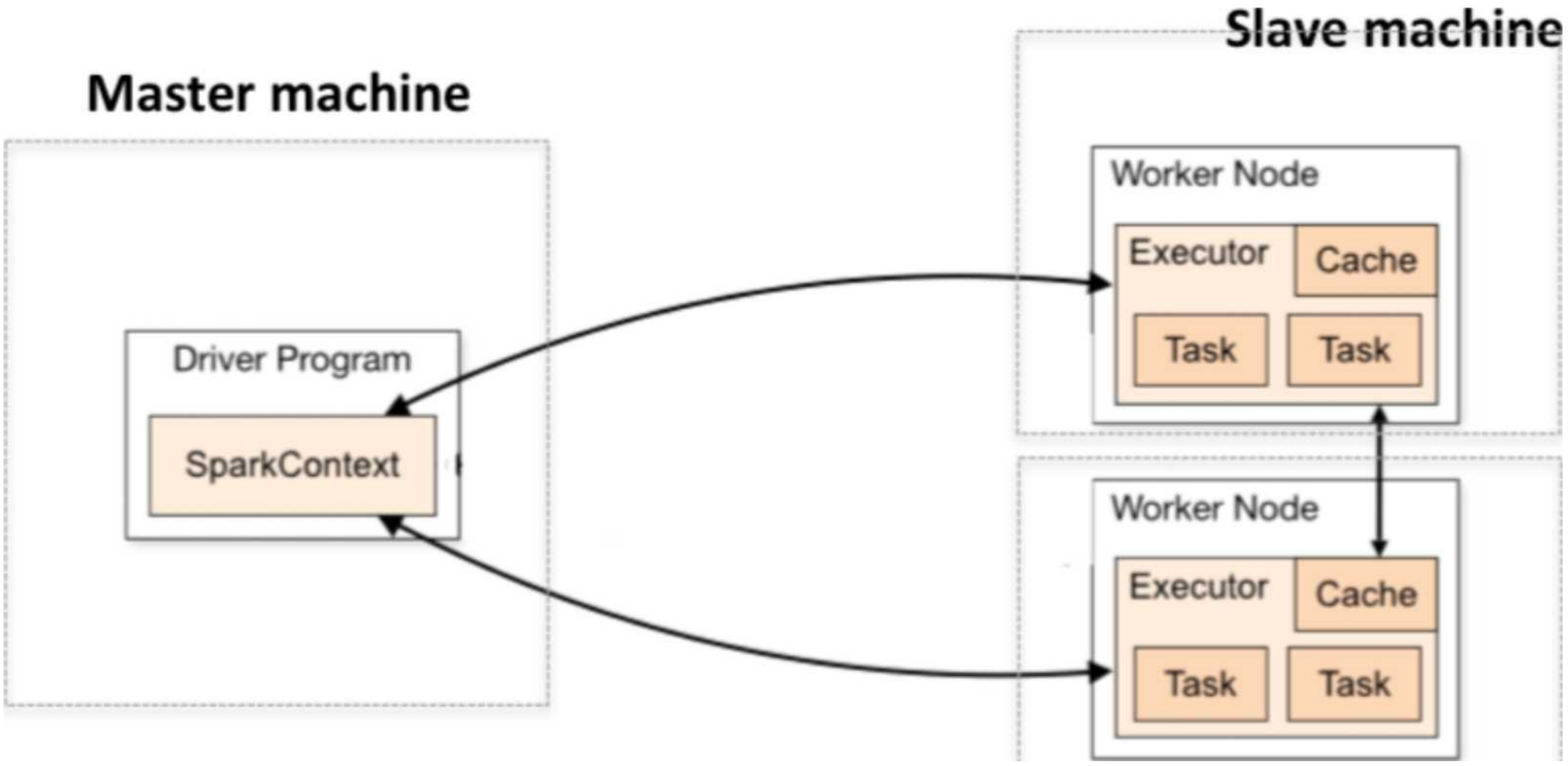
2. DRIVER PROGRAM : El proceso que ejecuta la función main() de la aplicación y crea El SparkContext.

3. **CLUSTER MANAGER :** Gestor de clústeres Un servicio externo para adquirir recursos en el clúster (por ejemplo, administrador independiente, Mesos, YARN)

4.WORKER NODE: Cualquier nodo que pueda ejecutar código de aplicación en el clúster.

5. EXECUTOR: Un proceso iniciado para una aplicación en un nodo de trabajo, que ejecuta tareas y mantiene los datos en memoria o almacenamiento en disco a través de ellos

EJECUCIÓN DE SPARK EN UN CLUSTER



MAQUINA ESCLAVA

API DE SPARK

SparkContext: es la parte principal de la API de Apache Spark, donde realizaremos todas las operaciones. Gracias a SparkContext se facilita la conexion de la aplicación con un cluster Spark. Permite que la aplicación spark acceda al clúster de Spark con la ayuda de Resource Manager.

SparkSession: es un punto de entrada a la funcionalidad subyacente de Spark. Todas las funciones disponibles con SparkContext también están disponibles en SparkSession. Además, proporciona API para trabajar en DataFrames y Datasets. . Incluye :

- Spark Context,
- SQL Context,
- Streaming Context,
- Hive Context.

Driver Manager: El Administrador de clústeres es el proceso maestro en el modo independiente de Spark. SparkContext puede conectarse a varios tipos de Driver Managers (ya sea el propio administrador de **clústeres independiente de Spark(Stanalone)** o Mesos/YARN), que asignan recursos entre aplicaciones

API SPARK

El sistema admite actualmente tres gestores de clústeres:

• Stanalone (Independiente): un administrador de clústeres simple incluido con Spark que facilita la configuración de un clúster.

•Apache Mesos – un administrador general de clústeres que también puede ejecutar Hadoop MapReduce y aplicaciones de servicio.

•Hadoop YARN – el administrador de recursos en Hadoop 2.

RDD

Existen varias **formas de generar RDD's**:
➔Obtener datos de un *fichero*.
➔Obtener datos almacenados en *memoria*.
➔Obtener datos de otro *RDD*.

OPERACIONES RDD

➔*Acciones*: devuelven un valor a la aplicacion. Algunos ejemplos sobre acciones sobre RDD son: contar los elementos que posee el mismo (***count()***), devolver un array con todos los elementos de un RDD (***collect()***).

➔ *Tranformaciones*: consiste en obtener un nuevo RDD tras transformar el original. Algunos ejemplos sobre tranformaciones sobre RDD son: crear un nuevo RDD en base a una funcion (map(funcion) / filter(funcion)).

RDD

Resilient Distributed Datasets (RDDs)

> Trabajaremos sobre colecciones de datos denominadas RDD:

> Es el concepto basico de trabajo en Spark

> Son inmutables. Es decir una vez creados no se pueden modificar.

> Se pueden transformar para crear nuevos RDDs o realizar acciones sobre ellos pero no modificar.

> Se guarda la secuencia de transformaciones para poder recuperar RDDs de forma eficiente si alguna maquina se cae

> Estan distribuidos en el cluster en los nodos workers

CARACTERISTICAS DE LOS RDD

RDD SON DISTRIBUIDOS: Cada RDD se divide en varias partes denominadas particiones, y estas particiones se dividen entre los clústeres.
Spark realiza automáticamente este proceso de particionamiento, por lo que no tiene que preocuparse por todos los detalles de cómo se particionan los datos en todo el clúster.

RDD SON INMUTABLES : No se puede cambiar una vez creados.
La inmutabilidad hace que sea seguro compartir RDD entre procesos.

RDD son RESISTENTES (RESILIENT): En caso de que cualquier nodo del clúster se deshaga, Spark puede recuperar las partes de los RDD de la entrada y recoger desde donde lo dejó.
Spark hace el trabajo pesado para que usted se asegure de que los RDD son tolerantes a errores.

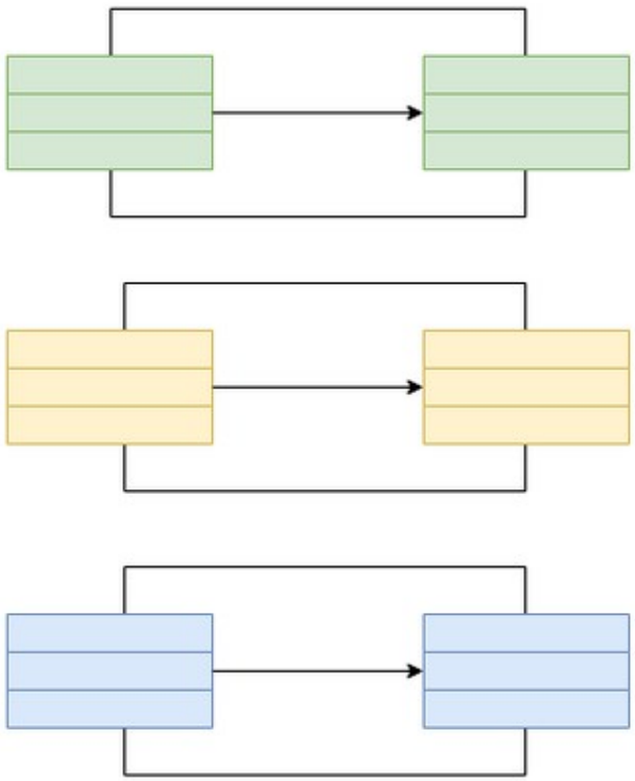
RDD

Tipos de transformaciones de un RDD
Existen dos tipos de operaciones de transformacion, ya que posiblemente los datos a procesar se encuentren en diferentes RDD's:

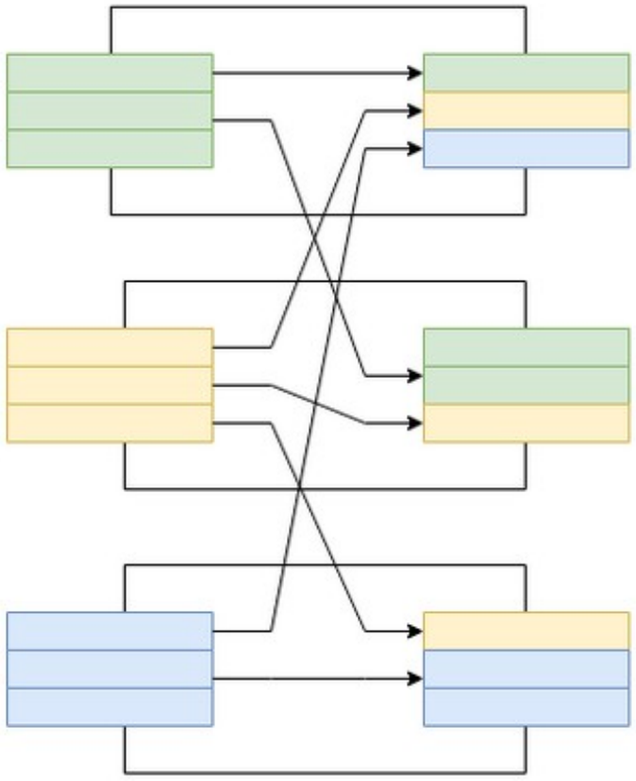
➢ **Narrow**: utilizado cuando los datos a tratar estan ubicados en la misma particion del RDD y no hace falta mezclar dichos datos. Algunos ejemplos de funciones para este tipo son: **filter(),sample(), map() o flatMap().**

➢ **Wide**: utilizado cuando los datos a tratar estan ubicados en diferentes particiones de un RDD y es necesario mezclar dichas particiones. Algunos ejemplos de funciones para este tipo son: **groupByKey() o reduceByKey().**

RDD



Transformación narrow



Transformación wide

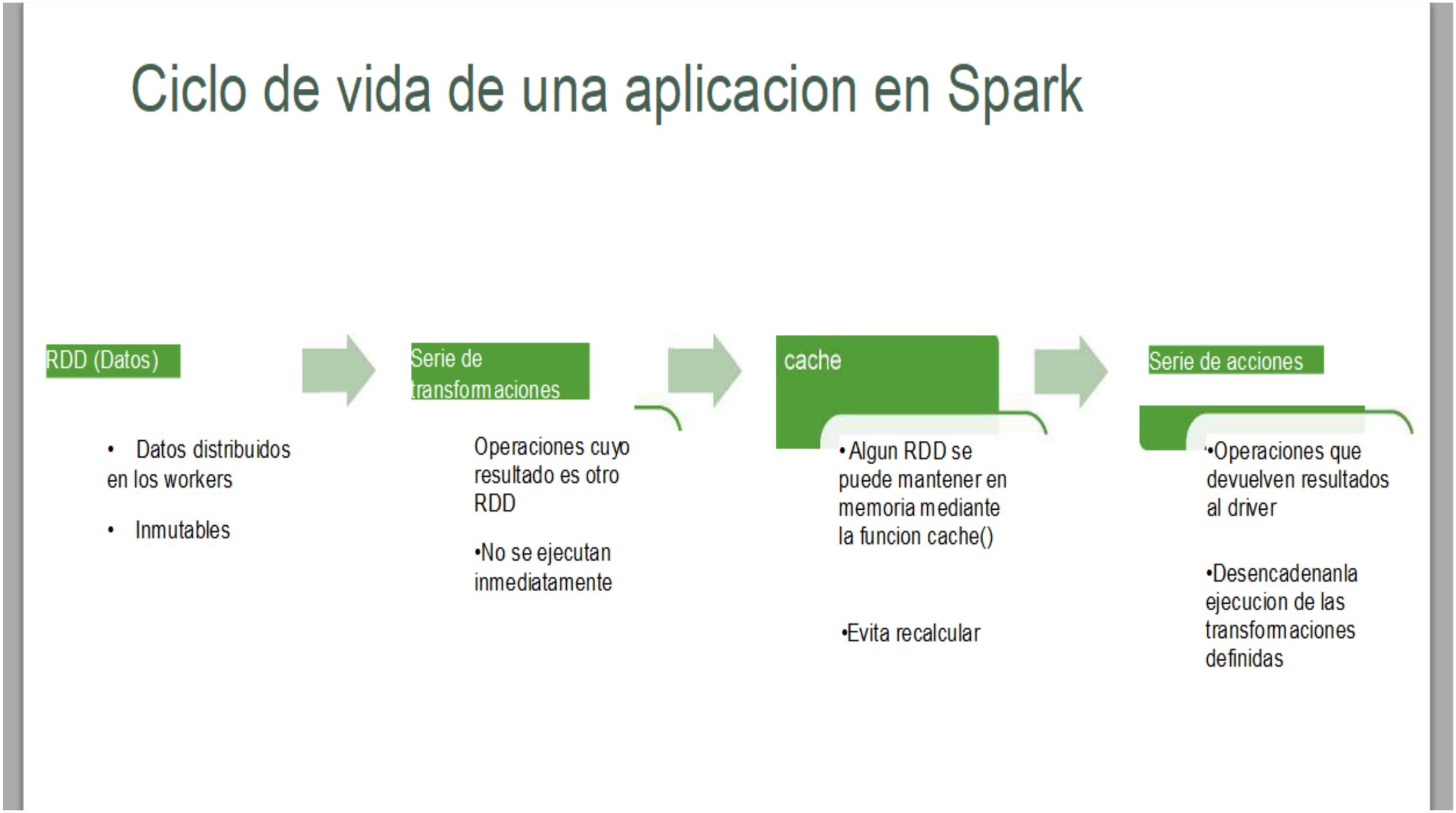
QUE ES UN DATASET

Un DATASET es básicamente una colección de datos; puede ser una lista de cadenas, una lista de enteros o incluso un número de filas en una base de datos relacional.

FLUJO GENERAL SPARK RDD

- Generar RDD iniciales a partir de datos externos.
- Aplicar transformaciones
- Lanzar acciones

FLUJO GENERAL SPARK RDD. CICLO DE VIDA



TRANSFORMACIONS filte() y map()

FILTER()
Toma una función y devuelve un RDD formado por la selección de los elementos que pasan la función de filtro.
Se puede usar para quitar algunas filas no válidas para limpiar el RDD de entrada o simplemente obtener un subconjunto del RDD de entrada basado en la función de filtro

```
val cleanedLines = lines.filter(line => !line.isEmpty)
```

MAP()
Toma una función y pasa cada elemento del RDD de entrada a través de la función, con el resultado de que la función es el nuevo valor de cada elemento en el RDD resultante.
Se puede utilizar para realizar solicitudes HTTP a cada URL en nuestro RDD de entrada, o se puede utilizar para calcular la raíz cuadrada de cada número.

```
val lines= sc.textFileC’in/uppercase.text")
val lengths = lines.map(line => line.length)
```

ACCIONES

Las acciones son las operaciones que devolverán un valor final al programa de controlador o conservarán los datos en un sistema de almacenamiento externo.

Las acciones forzarán la evaluación de las transformaciones necesarias para el RDD al que se les pidió.

ACCIONES COMUNES EN SPARK

- Collect
- Count
- CountByValue
- Take
- SaveAsTextFile
- reduce

EJEMPLO DE REDUCE