



# INTRODUCCIÓN

## ¿ Qué es ?

Apache Spark es un **framework** de **procesamiento distribuido.**

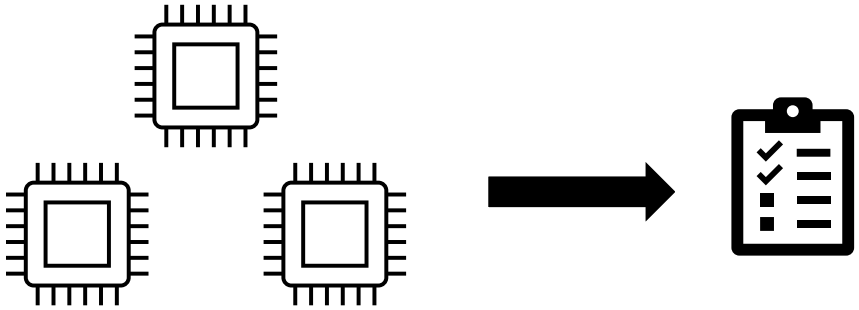
Provee APIs en de alto nivel para Java Scala , Python y R.

(1) Apache Spark para principiantes - BI Geek Blog (bi-geek.com)

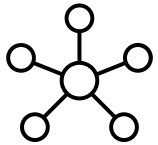
## ¿ Qué es ?

- **Framework**

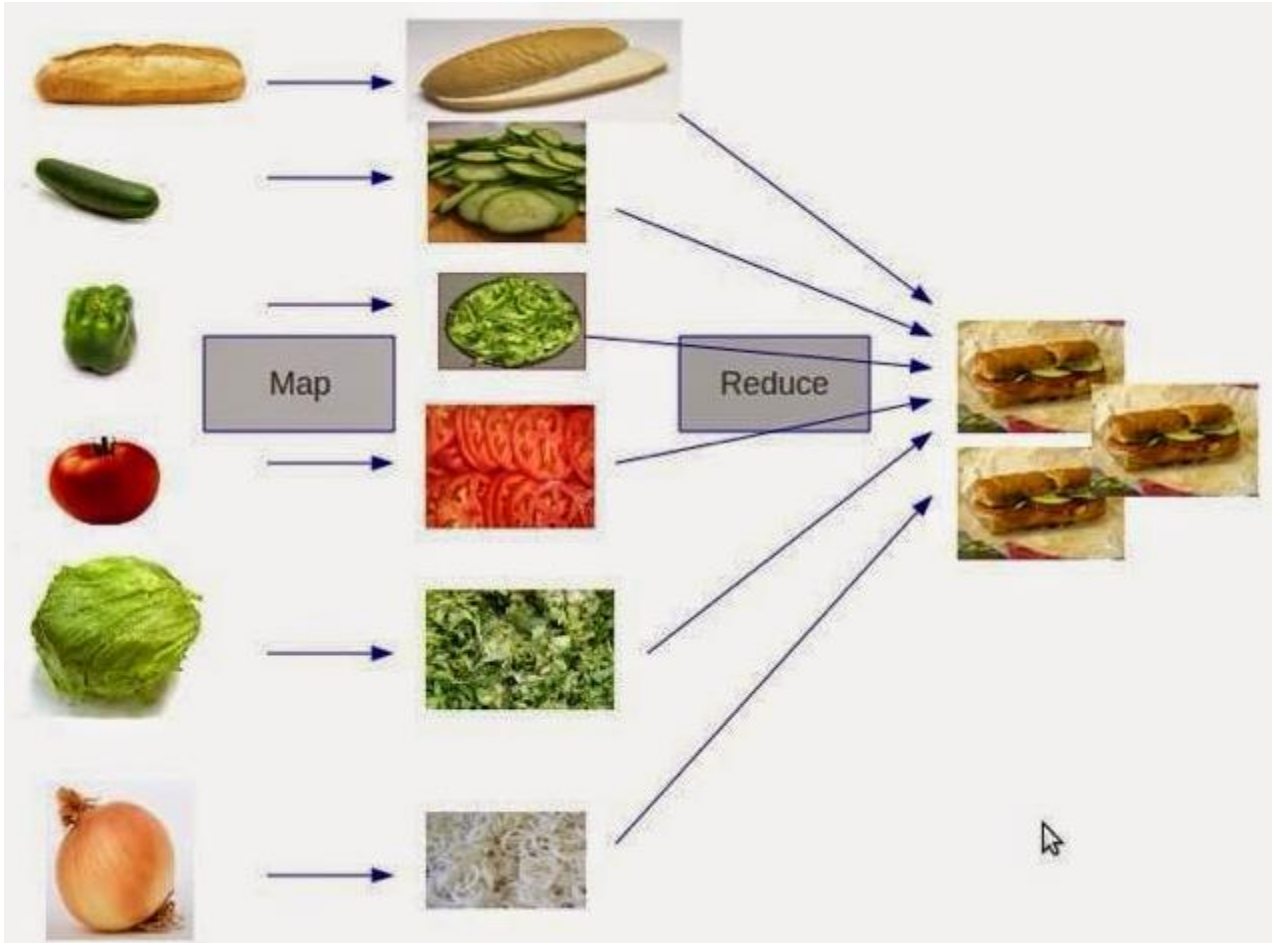
- **Procesamiento distribuido**



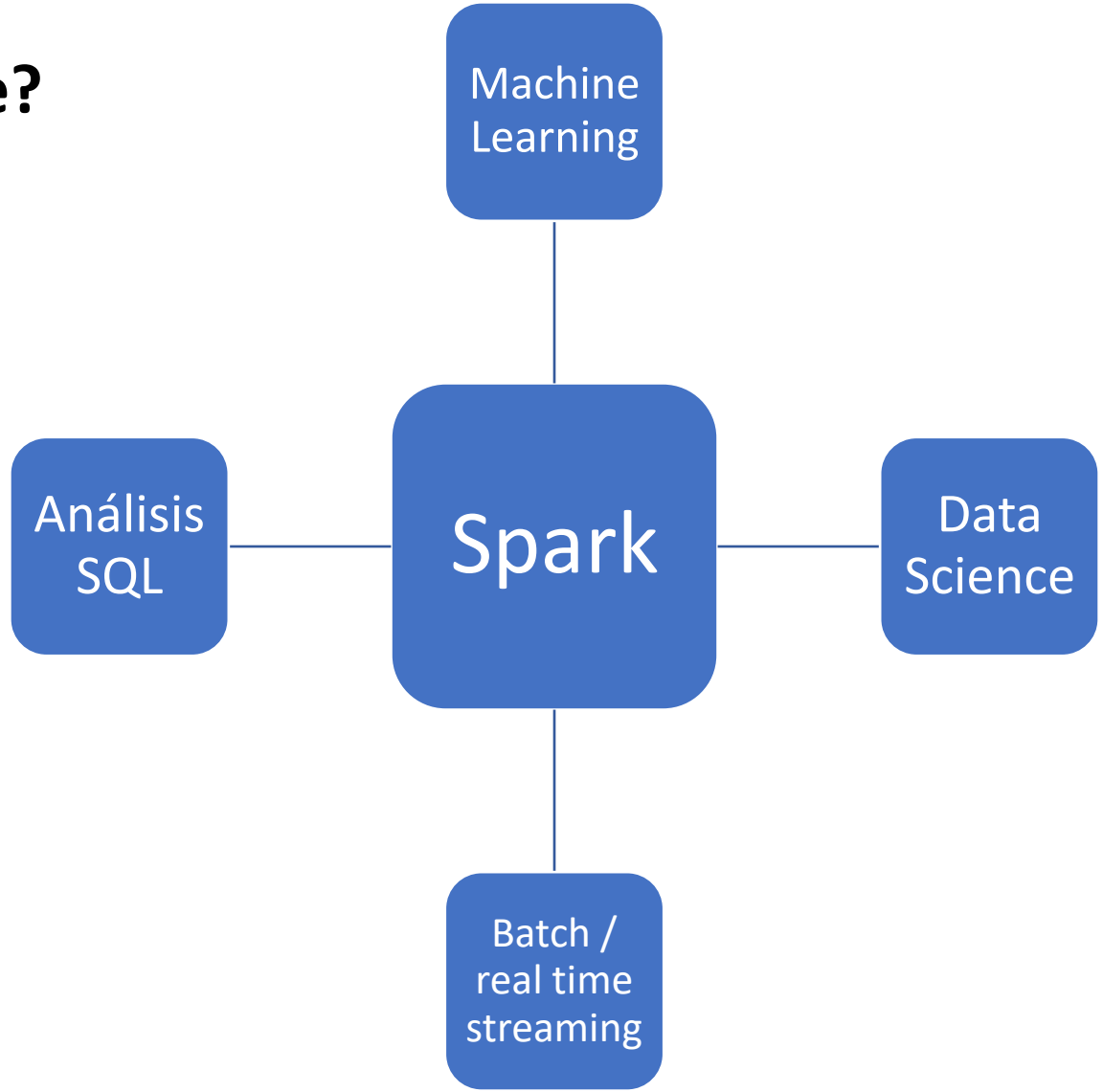
- Varios nodos



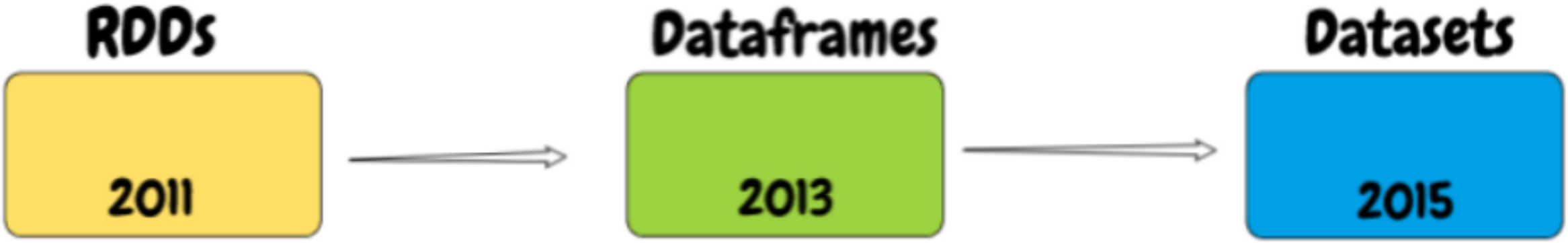
- MapReduce



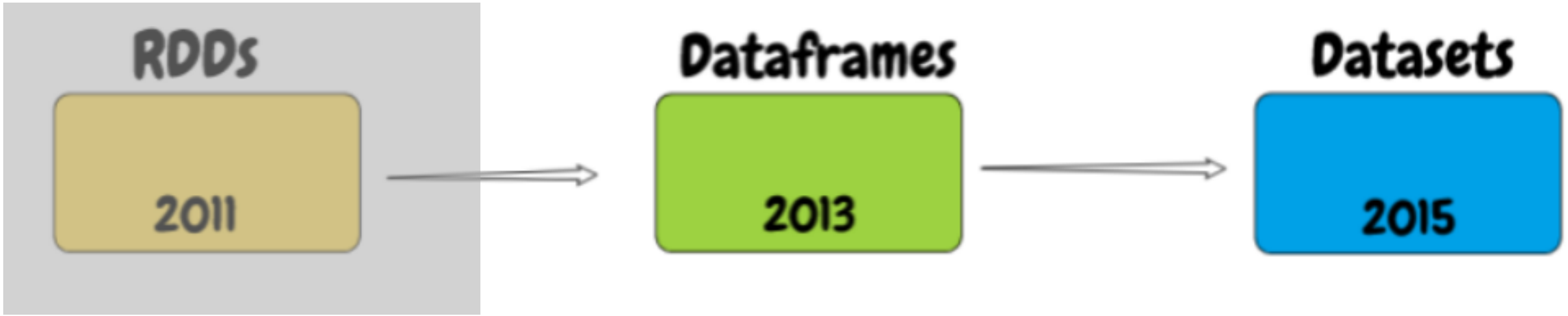
# ¿Para qué sirve?



## ¿Qué estructuras de datos maneja?



# ¿Qué estructuras de datos maneja?

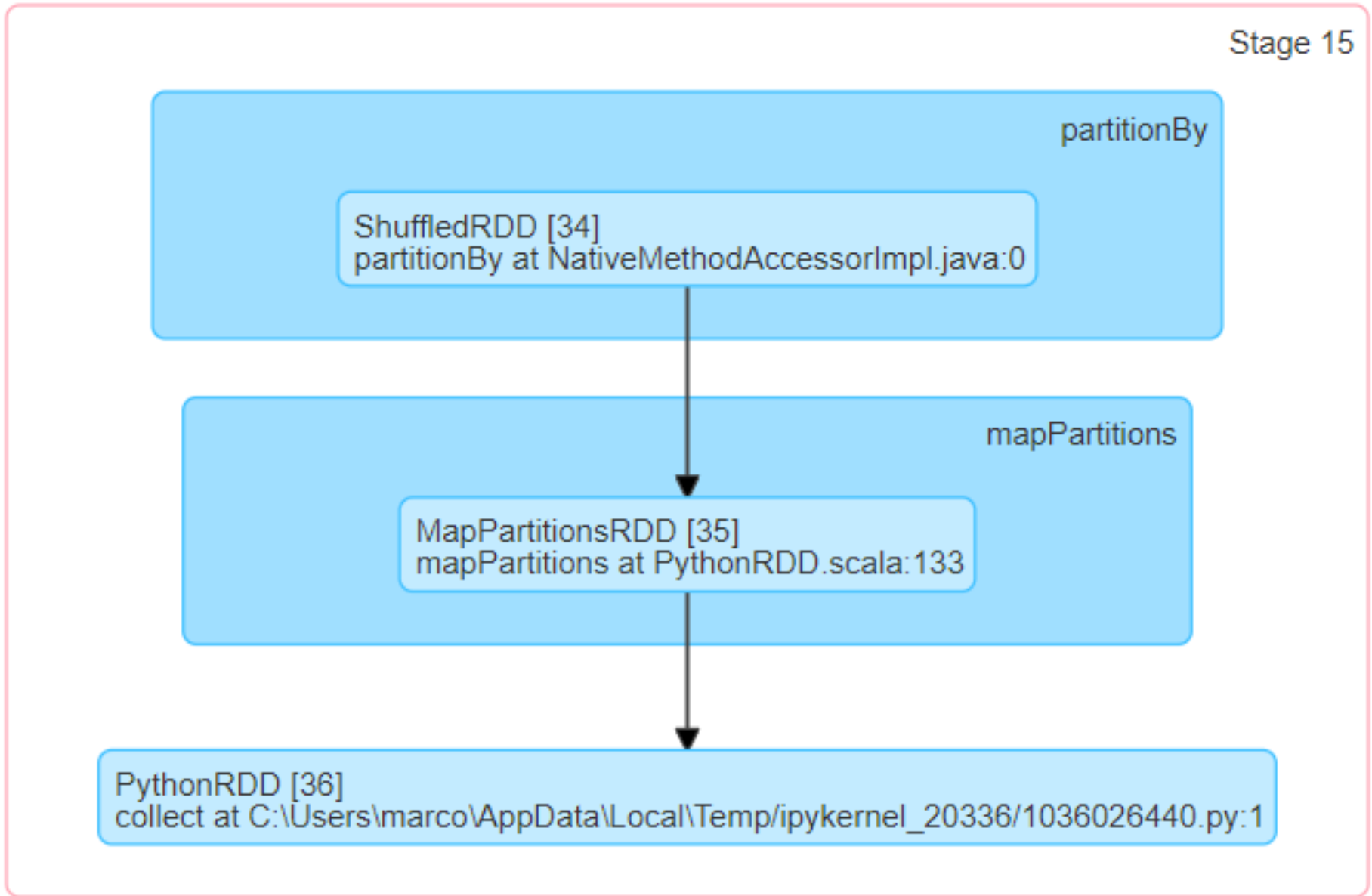


```
miRDD = sc.textFile("inversiones.csv")

miRDD.map(lambda x : (x.split(",")[3],1)).collect()
```

[31] ✓ 0.4s

```
... Output exceeds the size limit. Open the full output data in a text editor
[('2018-10-24', 1),
 ('2018-10-24', 1),
 ('2018-10-24', 1),
 ('2018-10-24', 1),
 ('2018-10-24', 1),
 ('2018-10-24', 1),
 ('2018-10-24', 1),
 ('2018-10-24', 1),
 ('2018-10-24', 1),
```


Stage 15
partitionBy
ShuffledRDD [34]
partitionBy at NativeMethodAccessorImpl.java:0
mapPartitions
MapPartitionsRDD [35]
mapPartitions at PythonRDD.scala:133
PythonRDD [36]
collect at C:\Users\marco\AppData\Local\Temp/ipykernel_20336/1036026440.py:1

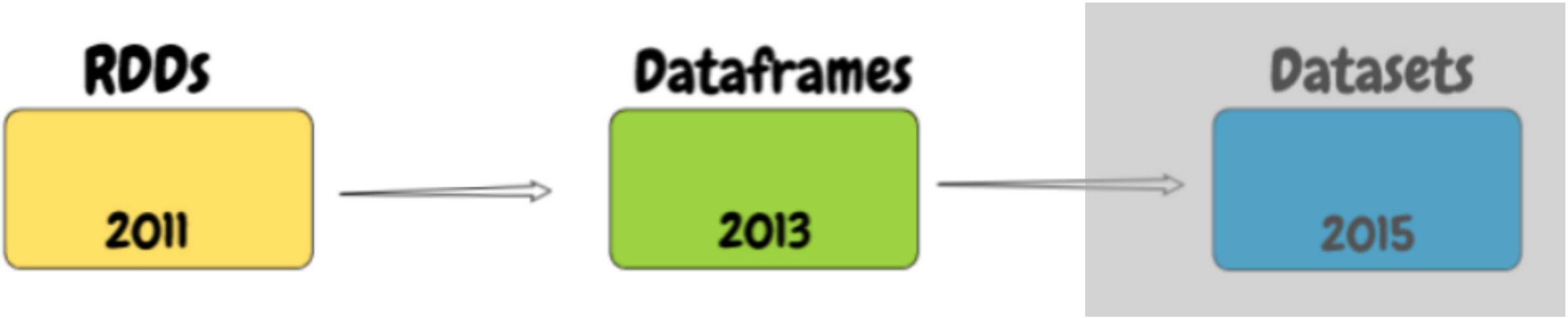
# ¿Qué estructuras de datos maneja?

```
newRDD.toDF(schema=["dates","qty"]).show(2)
```

✓ 1.4s

```
+----------+---+
|     dates|qty|
+----------+---+
|2018-10-24| 25|
|2019-01-02| 16|
+----------+---+
only showing top 2 rows
```

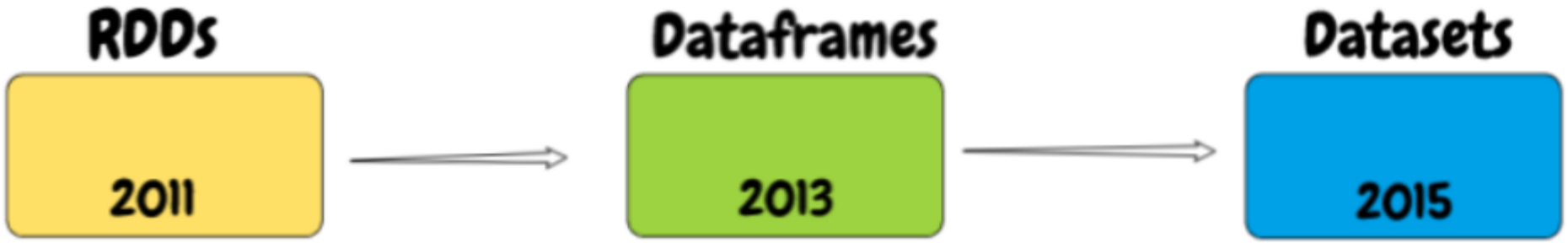
## ¿Qué estructuras de datos maneja?



```
// define a case class that represents the device data.
case class DeviceIoTData (
  battery_level: Long,
  c02_level: Long,
  cca2: String,
  cca3: String,
  cn: String,
  device_id: Long,
  device_name: String,
  humidity: Long,
  ip: String,
  latitude: Double,
  longitude: Double,
  scale: String,
  temp: Long,
  timestamp: Long
)

// read the JSON file and create the Dataset from the ``case class`` DeviceIoTData
// ds is now a collection of JVM Scala objects DeviceIoTData
val ds = spark.read.json("/databricks-datasets/iot/iot_devices.json").as[DeviceIoTData]
```

## ¿Qué estructuras de datos maneja?



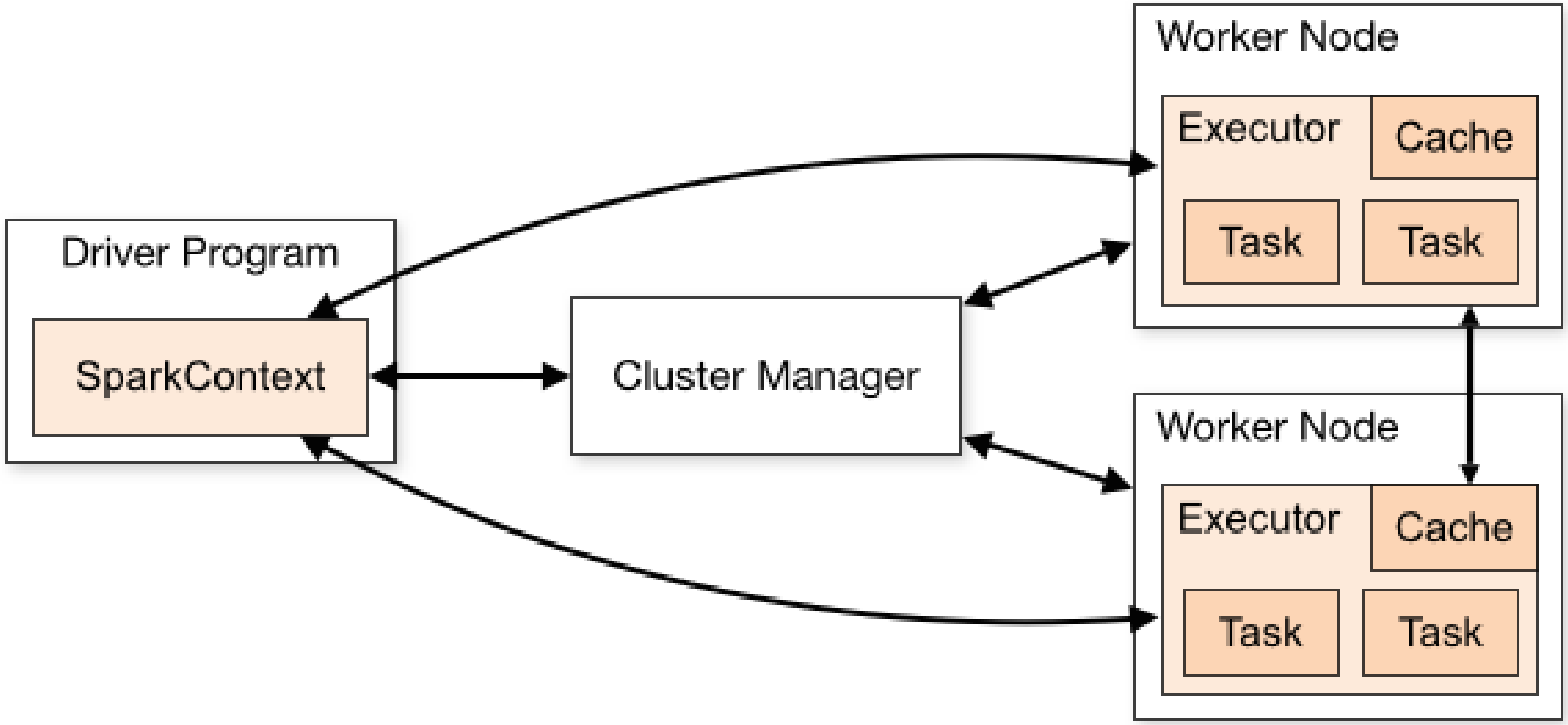

## ¿Cuándo utilizar cuál?

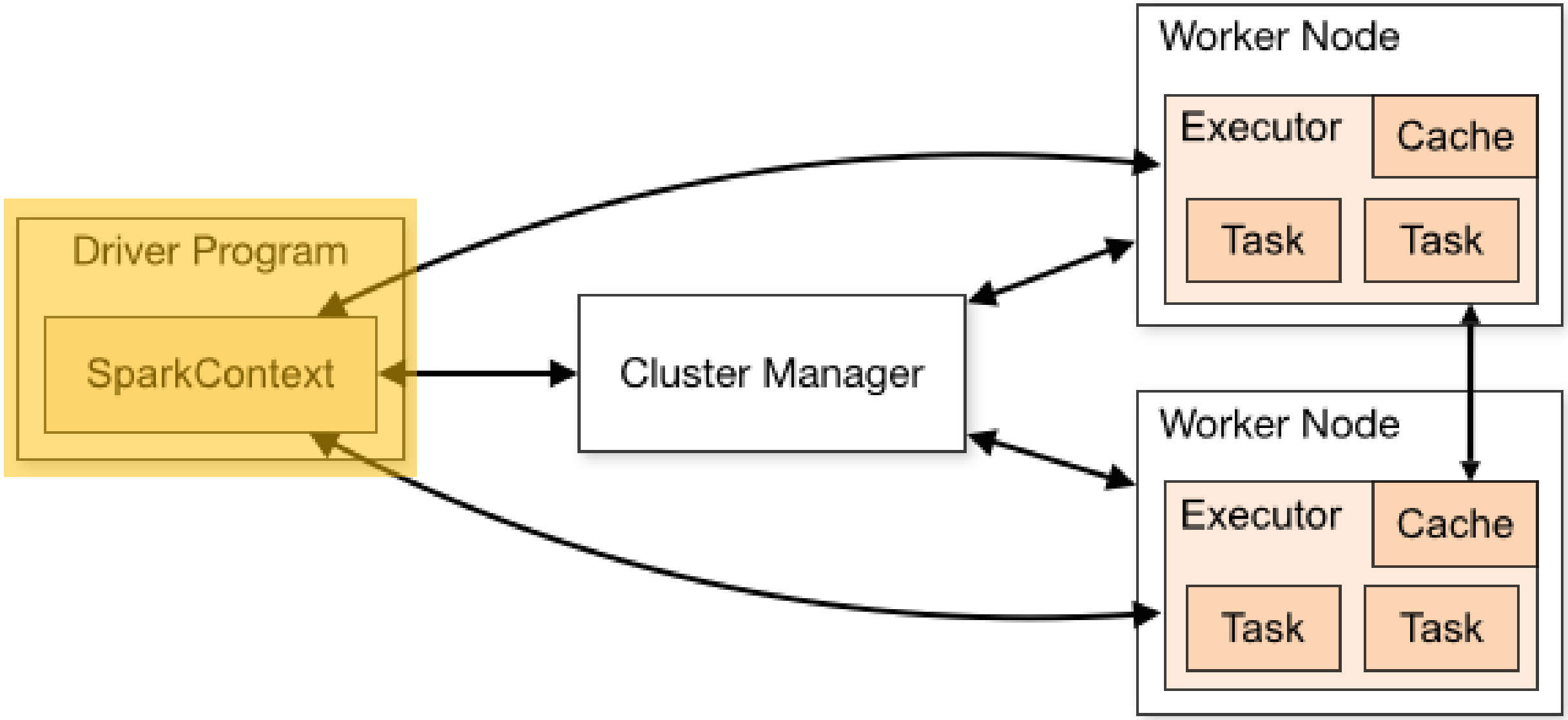
Si quieres una semántica rica, abstracciones de alto nivel, segura, entonces ve a por los Dataframes o Datasets. Si necesitas más control sobre la parte de preprocesamiento usa los RDDs.

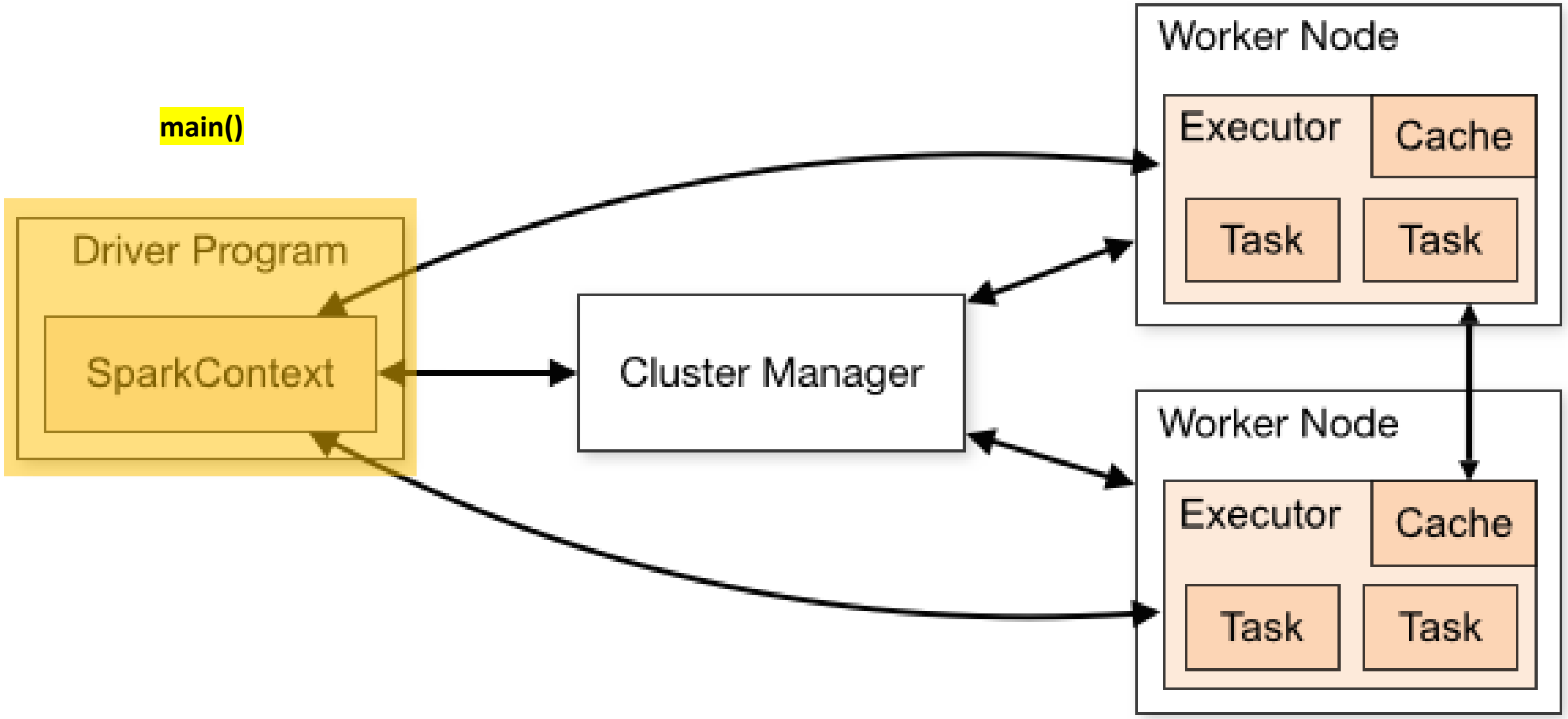
| | RDD | Dataframe | Datasets |
|---|---|---|---|
| Representación de Datos | RDD es una colección distribuida de datos. | Una colección distribuida de datos organizada por columnas | Es una extensión de dataframe con más características de seguridad de tipos de dato y una interfaz orientada a objetos. |
| Optimización | No tiene un motor de optmización embebido. El desarrollador tiene que optimizar a través del código. | Usa un catalyst para propósitos de optimización | Usa un catalyst para propósitos de optimización |
| Esquema | No hay un esquema inferido, el desarrollador debe inputar manualmente. | Automáticamente permite inferir el esquema. | Automáticamente infiere el esquema usando el motor SQL. |
| Operaciones de agregación | Es el menos rápido a comparación que los otros dos. | Posee un API para realizar operaciones de agregación. Lo hace más rápido que RDD y Dataset. | Un Dataset es más rápido que un RDD pero menos que los Dataframes. |

# Arquitectura Spark

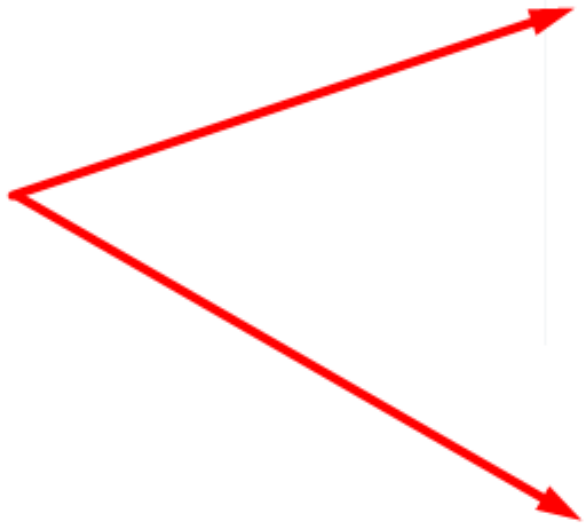
Componentes


Worker Node
Executor
Cache
Task
Task
Driver Program
SparkContext
Cluster Manager
Worker Node
Executor
Cache
Task
Task


Worker Node
Executor
Cache
Task
Task
Driver Program
SparkContext
Cluster Manager
Worker Node
Executor
Cache
Task
Task


main()
Driver Program
SparkContext
Cluster Manager
Worker Node
Executor
Cache
Task
Task
Worker Node
Executor
Cache
Task
Task

ENTRY POINTS

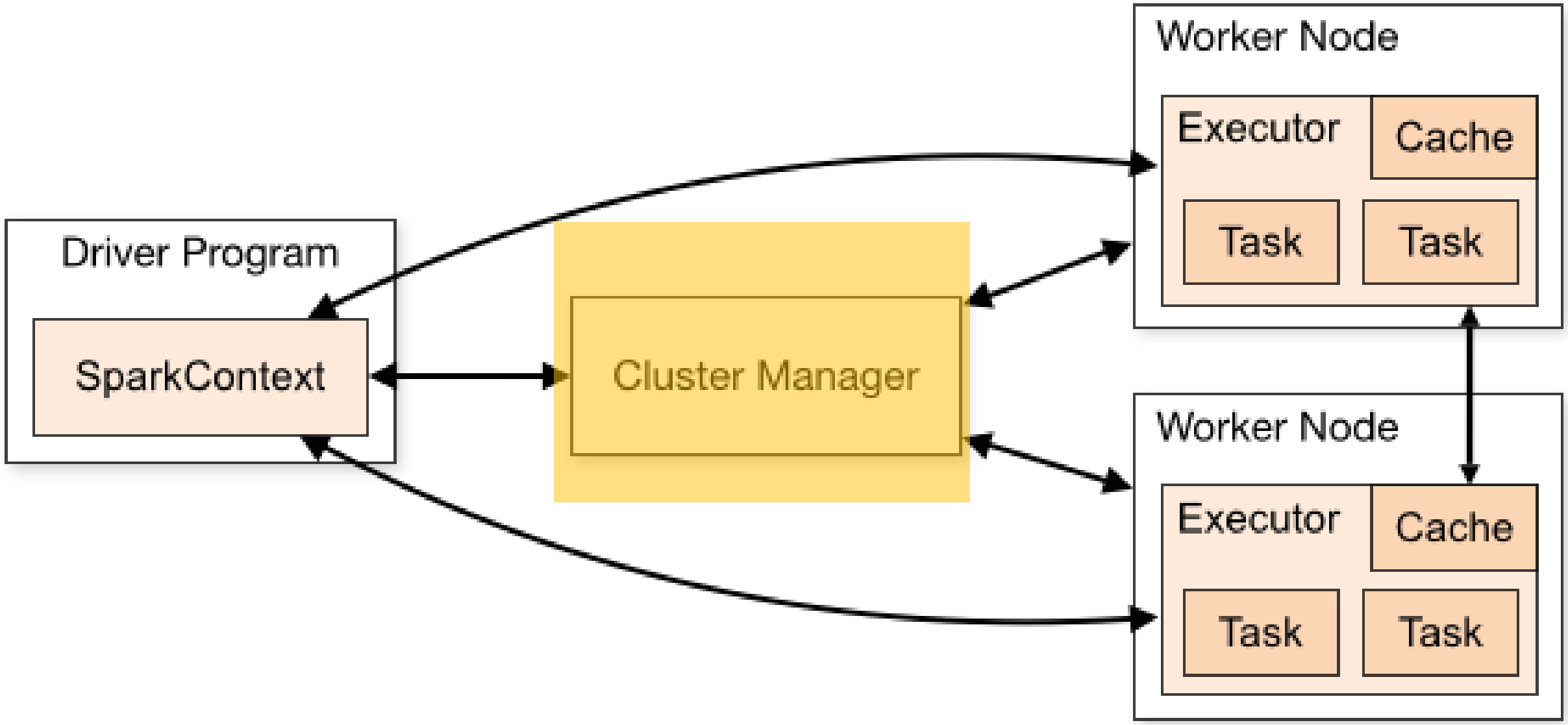


## SparkContext

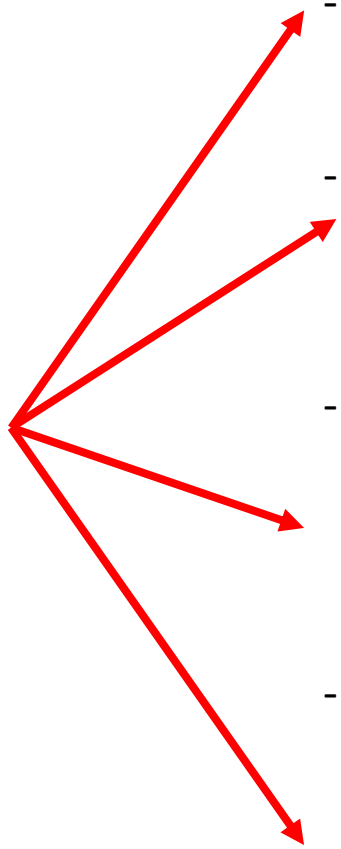
Spark SparkContext is an entry point to Spark and defined in `org.apache.spark` package since 1.x and used to programmatically create Spark RDD, accumulators and broadcast variables on the cluster

## SparkSession

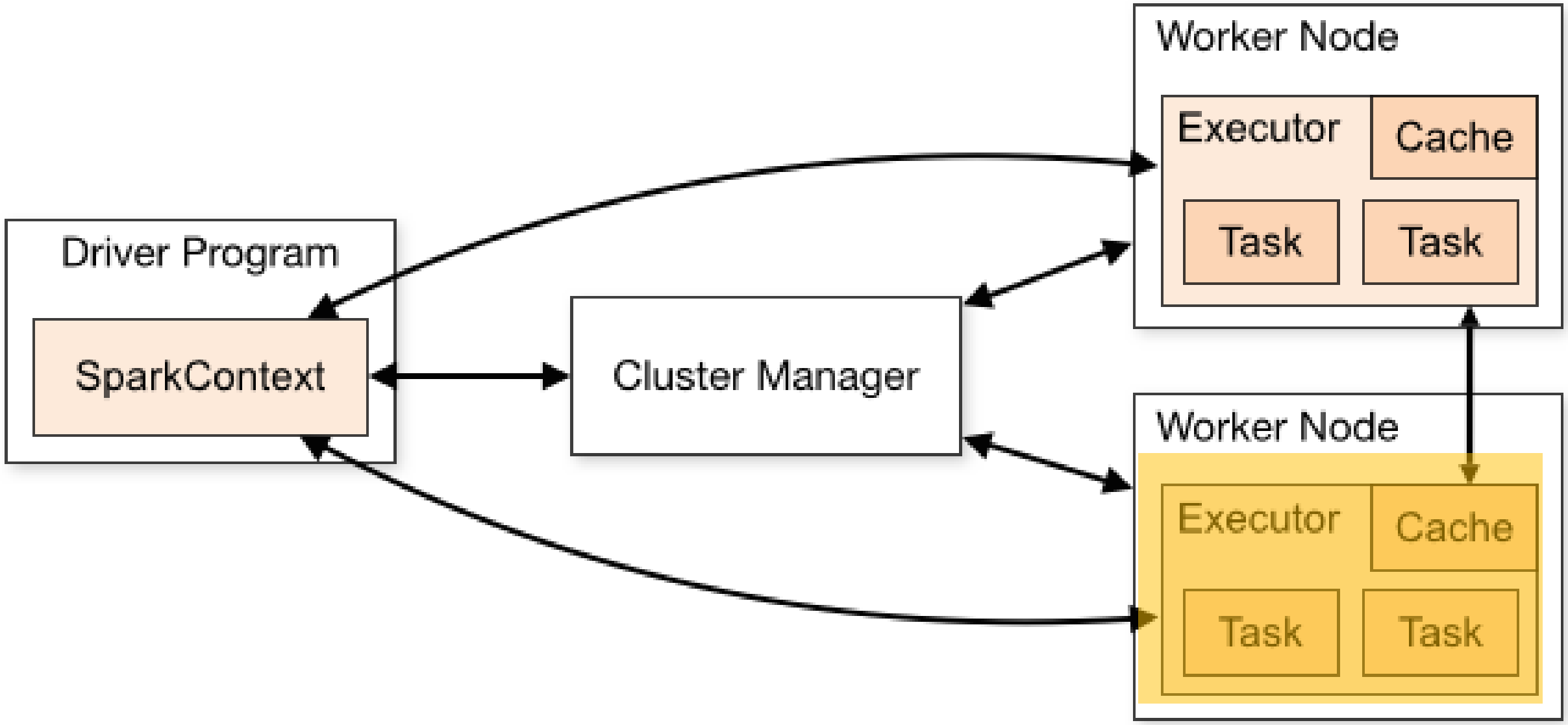
SparkSession introduced in version 2.0 and is an entry point to underlying Spark functionality in order to programmatically create Spark RDD, DataFrame and DataSet.


Worker Node
Executor
Cache
Task
Task
Driver Program
SparkContext
Cluster Manager
Worker Node
Executor
Cache
Task
Task

Cluster Manager



- **Standalone**: Gestor de clústeres simple, de características limitadas, incorporado con Spark.
- **Apache Mesos**: Un cluster-manager de código abierto que en su día fue muy popular para cargas de trabajo de big data (no sólo Spark) pero que está en declive en los últimos años.
- **Hadoop YARN**: El cluster-manager basado en JVM de hadoop lanzado en 2012 y más utilizado hasta la fecha, tanto para despliegues on-premise (por ejemplo, Cloudera, MapR) como en la nube (por ejemplo, EMR, Dataproc, HDInsight).
- **Kubernetes**: Spark se ejecuta de forma nativa en Kubernetes desde la versión Spark 2.3 (2018). Este modo de despliegue está ganando tracción rápidamente, así como respaldo empresarial (Google, Palantir, Red Hat, Bloomberg, Lyft). Sin embargo, a partir de junio de 2020 su soporte sigue marcado como experimental.

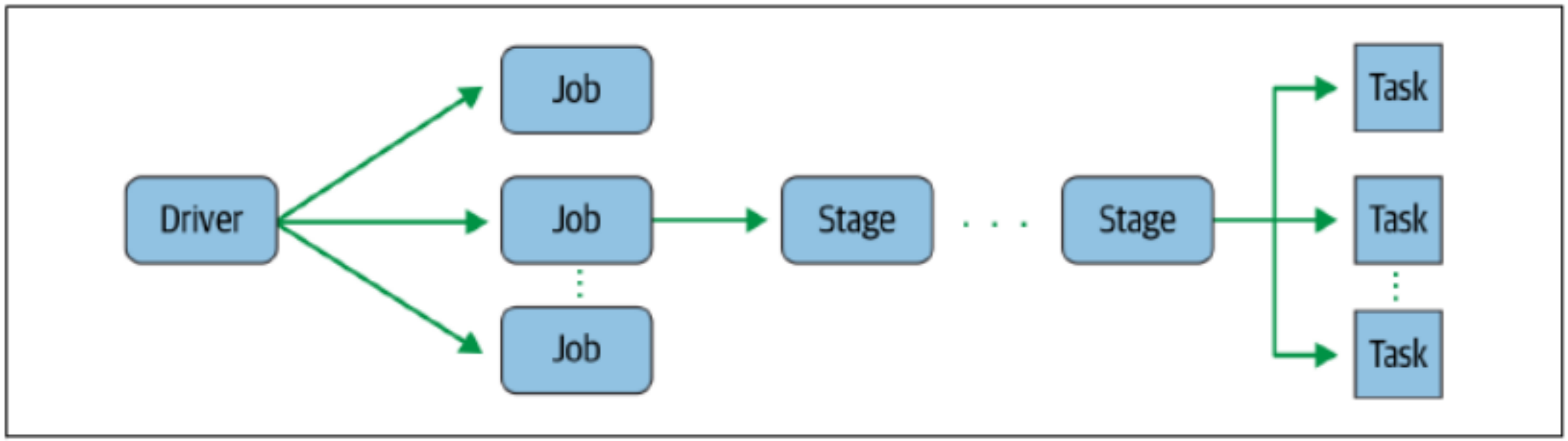

Worker Node
Executor
Cache
Task
Task
Driver Program
SparkContext
Cluster Manager
Worker Node
Executor
Cache
Task
Task

**Job** - Un cálculo paralelo que consiste en múltiples tareas que se generan en respuesta a una acción de Spark (por ejemplo, save(), collect()).

**Stage** - Cada trabajo se divide en conjuntos más pequeños de tareas llamadas etapas que dependen unas de otras. Como parte de los nodos DAG, las etapas se crean en función de las operaciones que pueden realizarse en serie o en paralelo.

**Task** - Una sola unidad de trabajo o ejecución que se enviará a un ejecutor de Spark.

# Job -> Stage -> Task



# Términos de la arquitectura Spark

| Term | Significado |
|---|---|
| Application | Programa de usuario construido sobre Spark. Consiste en un programa controlador y ejecutores en el clúster. |
| Driver program | El proceso que ejecuta la función main() de la aplicación y crea el SparkContext |
| Cluster manager | Un servicio externo para adquirir recursos en el clúster (por ejemplo, un gestor independiente, Mesos, YARN, Kubernetes) |
| Deploy mode | Distingue dónde se ejecuta el proceso del controlador. En el modo "cluster", el framework lanza el driver dentro del cluster. En el modo "cliente", el remitente lanza el controlador fuera del clúster. |
| Worker node | Cualquier nodo que pueda ejecutar código de aplicación en el clúster |
| Executor | Un proceso lanzado para una aplicación en un nodo trabajador, que ejecuta tareas y mantiene datos en memoria o almacenamiento en disco a través de ellas. Cada aplicación tiene sus propios ejecutores. |
| Task | Una unidad de trabajo que se enviará a un ejecutor |
| Job | Una computación paralela que consiste en múltiples tareas que se generan en respuesta a una acción de Spark (por ejemplo, save, collect); verás este término utilizado en los registros del controlador. |
| Stage | Cada trabajo se divide en conjuntos más pequeños de tareas denominadas etapas que dependen unas de otras (similar a las etapas map y reduce en MapReduce); verás este término utilizado en los registros del controlador. |