# Análisis de Datos y Big Data

## Sesión 5 : Procesamiento de datos en tiempo real con PySpark


Presentan:

Dr. Ulises Olivares Pinto

Joshelyn Yanori Mendoza Alfaro

René Delgado Servín


BLOQUE
Innovación, Tecnología y Creatividad.

# Contenido

1. Aplicación y ejercicios de análisis exploratorio de datos

2. Reducción de dimensionalidad

3. Procesamiento con Hadoop

# Contenido

1. FUNDAMENTOS DE PYSPARK

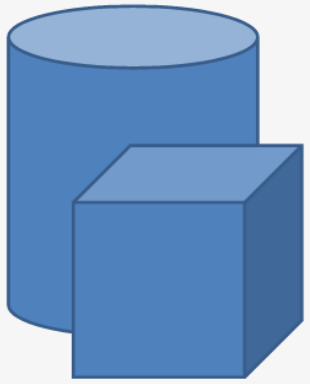
2. SPARK DATAFRAMES Y RDDS EN PYTHON

3. SPARK STREAMING Y PROCESAMIENTO EN TIEMPO REAL

4. OPTIMIZACIÓN DE CONSULTAS CON PYSPARK

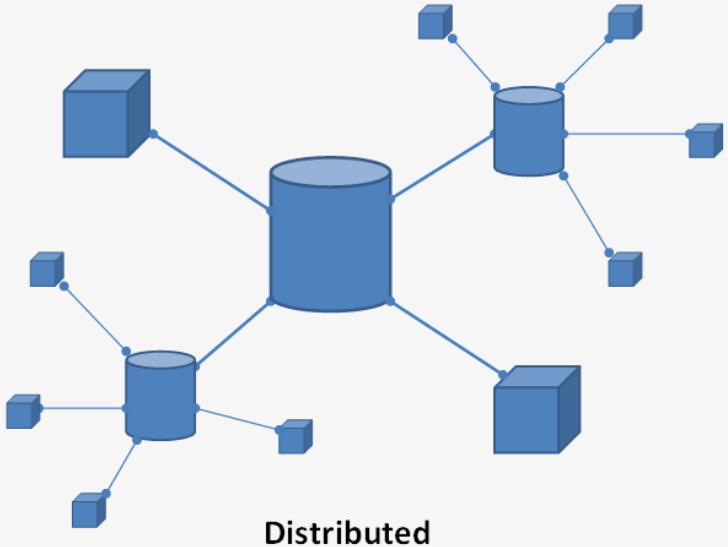
5. DEMO Y EJERCICIOS

6. DISCUSIÓN Y PREGUNTAS



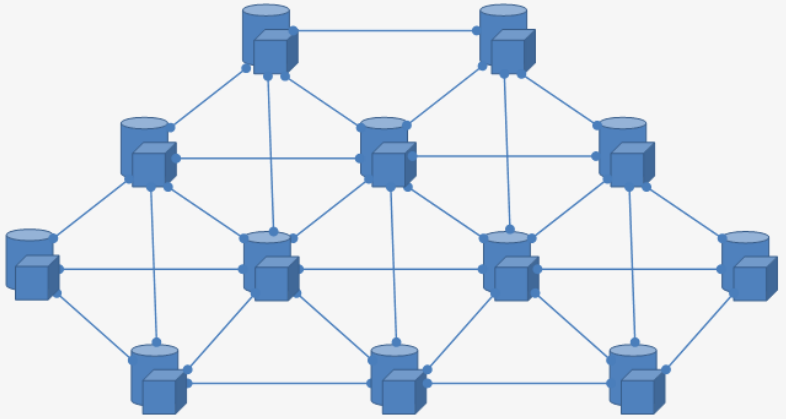
**Centralized**

*one node does everything*



**Distributed**

*nodes distribute work to sub-nodes*



**Decentralized**

*nodes are only connected to peers*

# Arquitecturas de Memoria Compartida vs Distribuida

# Memoria Compartida

En una arquitectura de memoria compartida, todos los procesadores en un sistema tienen acceso a una **memoria común**.

Cada procesador puede **leer y escribir** en una única dirección de memoria, y todos los cambios realizados en esa memoria son visibles para todos los procesadores.

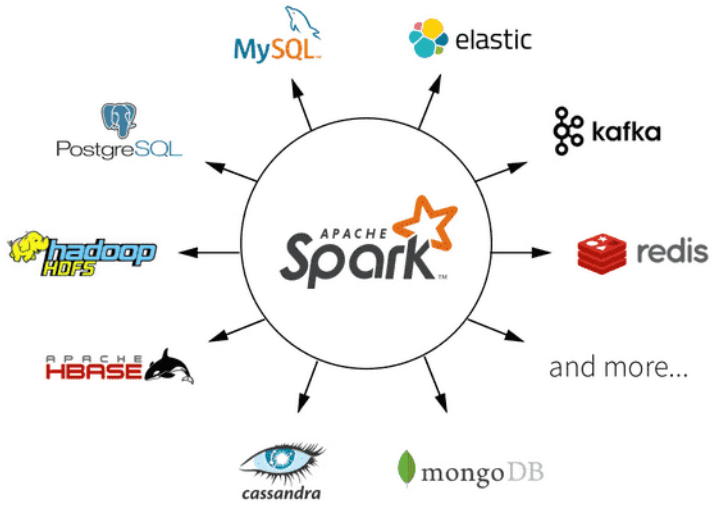
# Memoria Distribuida

En una arquitectura de memoria distribuida, cada procesador tiene su propia **memoria local**, y los procesadores se comunican entre sí **enviando mensajes**.

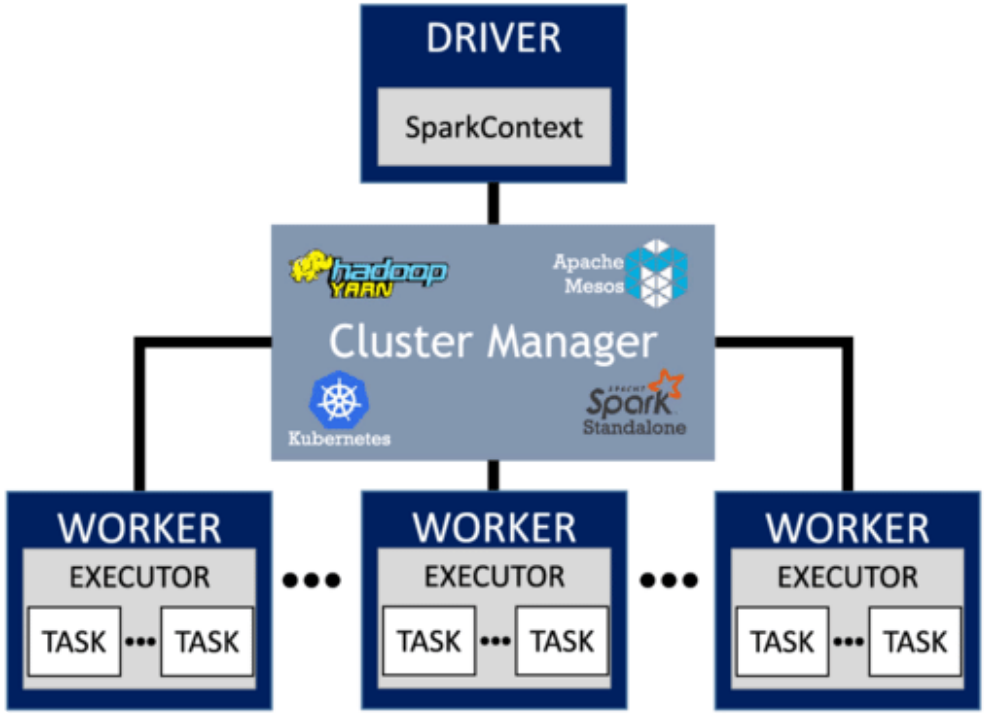
**No existe un espacio de memoria común** a todos los procesadores.

# Introducción a Apache Spark

Apache Spark es un motor de procesamiento de datos de **código abierto**, **rápido** y de **propósito general** que permite el **procesamiento distribuido** de grandes volúmenes de datos.

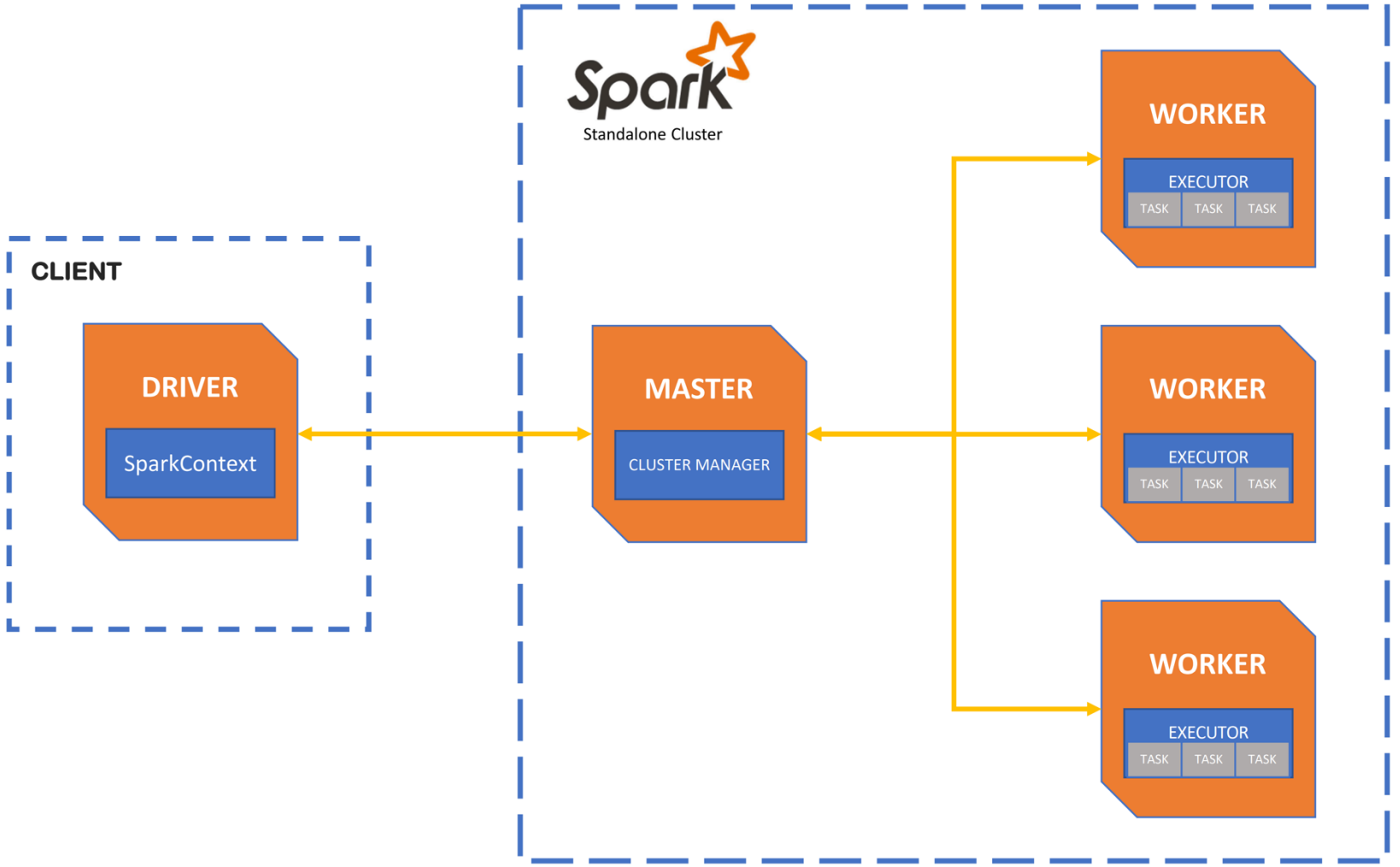


# Arquitectura de Spark



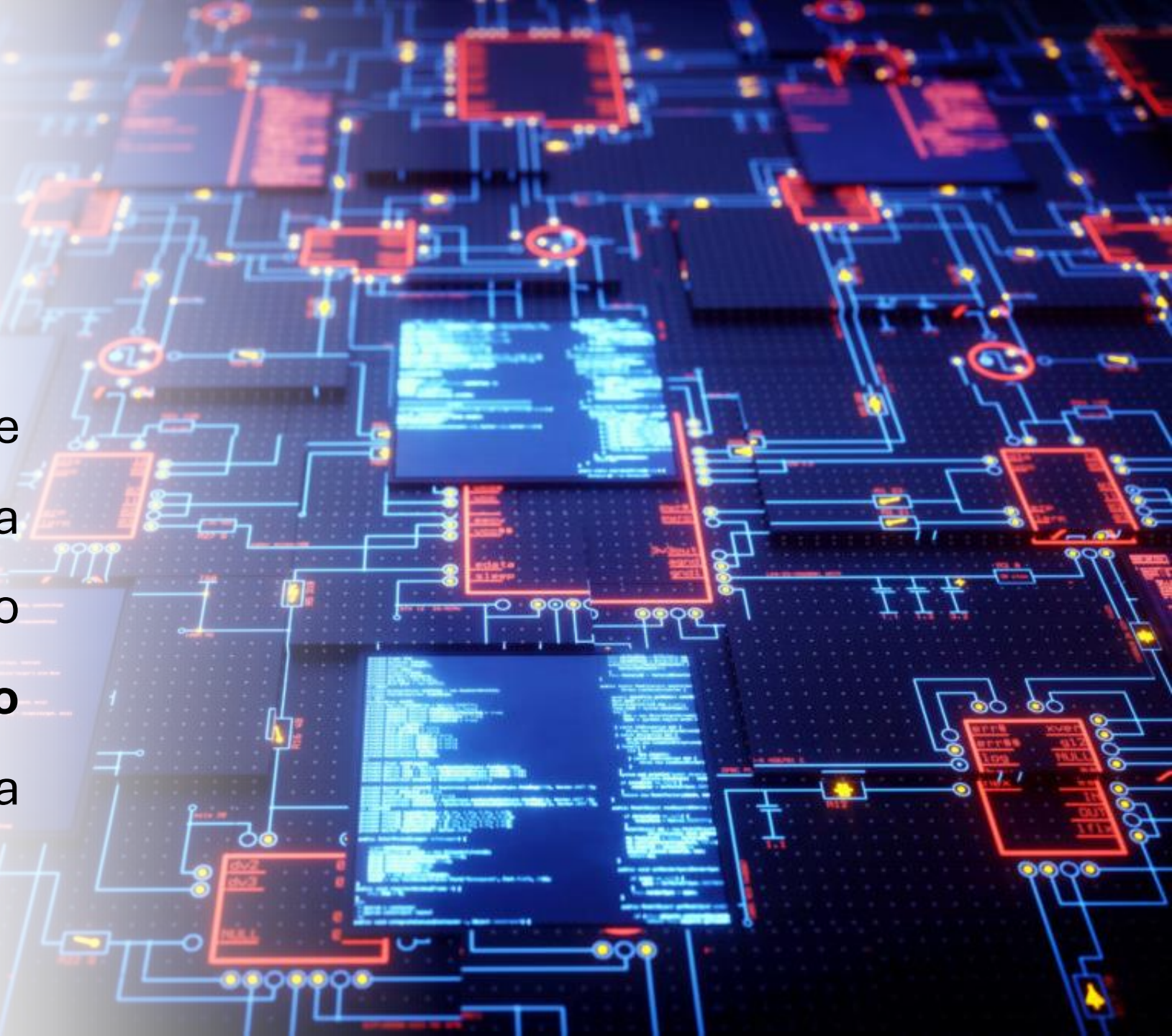

- La arquitectura de Spark se basa en un modelo **maestro-esclavo**, donde el nodo maestro coordina la ejecución de tareas en los nodos esclavos.
- **Driver:** Coordina la ejecución.
- **Workers:** Ejecutan las tareas.

# Diagrama de la Arquitectura de Spark



# Modo Local vs. Modo Distribuido

Spark puede ejecutarse en **modo local** para pruebas y desarrollo, o en **modo distribuido** para procesamiento a gran escala.



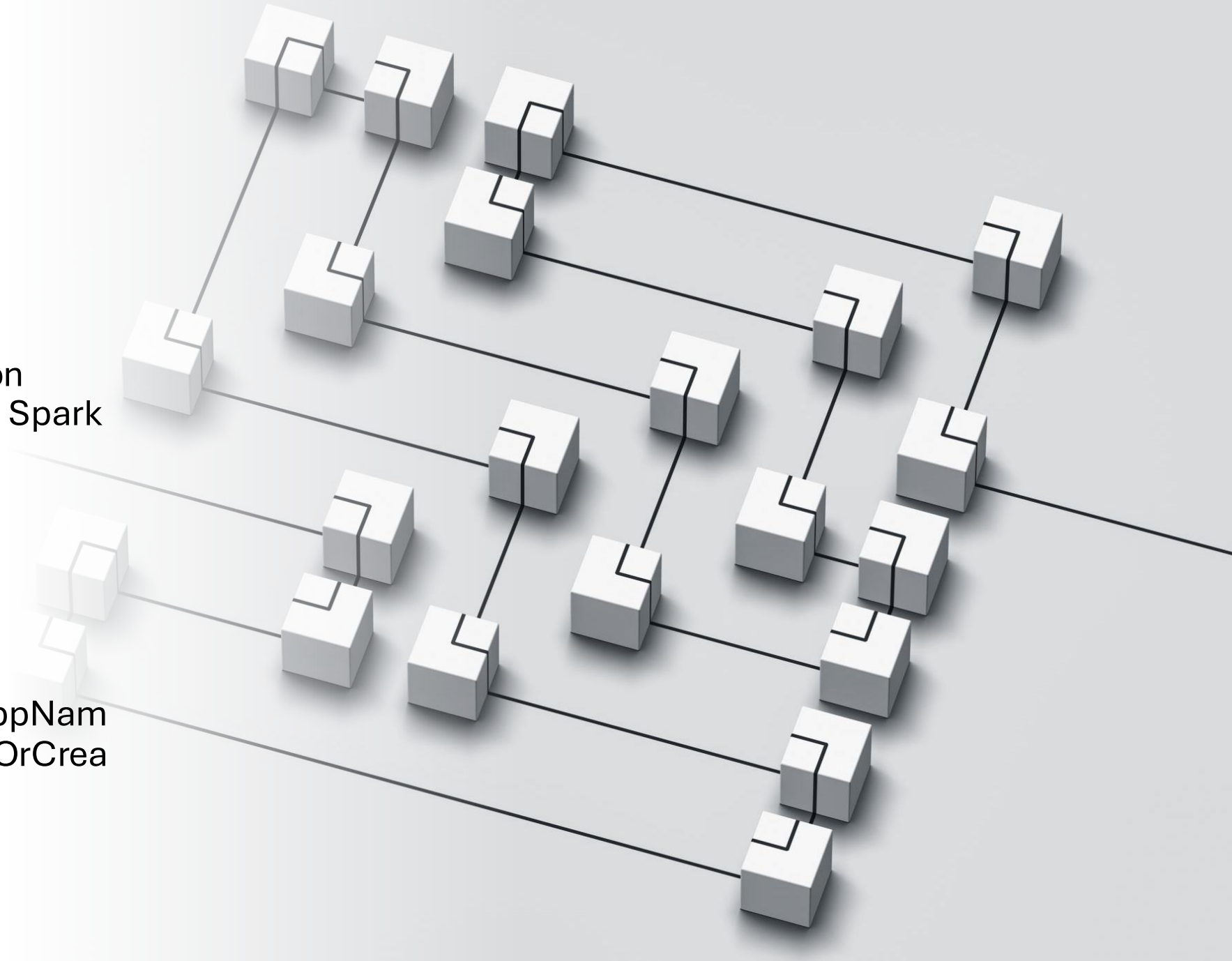
# Configuración de PySpark en Diferentes Entornos

- PySpark puede configurarse en entornos locales o en clústeres distribuidos.

# Configuración Básica de SparkSession

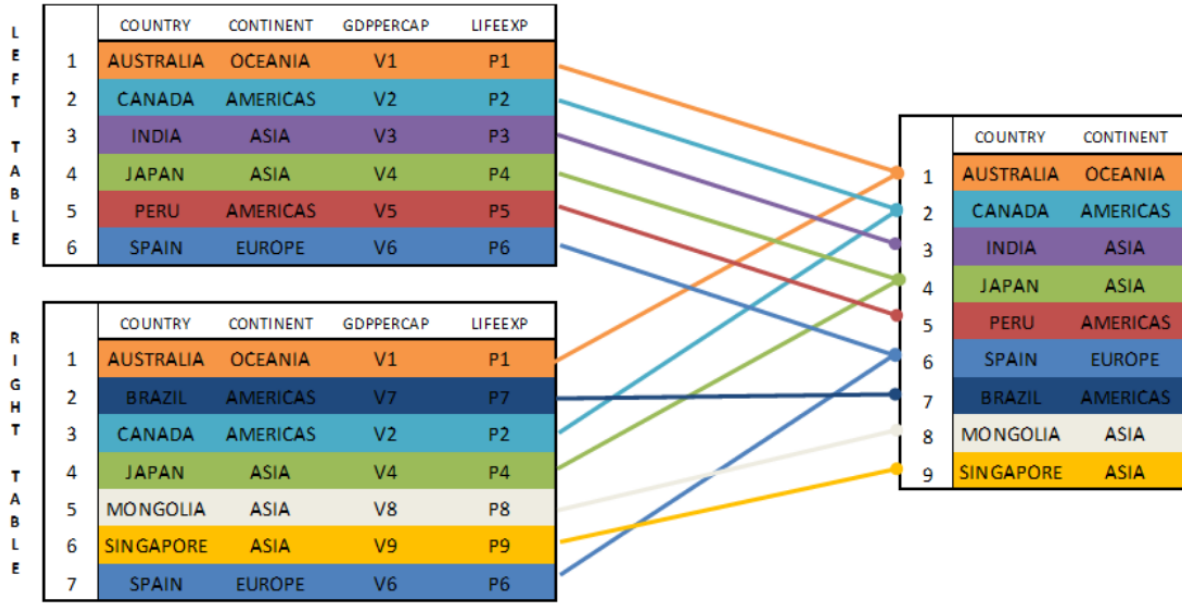
- Ejemplo de configuración básica de una sesión de Spark en PySpark.
- ```python
  from pyspark.sql import SparkSession

  spark = SparkSession.builder.appName('PySparkSession').getOrCreate()
  ```


  ```

# Operaciones Avanzadas en DataFrames

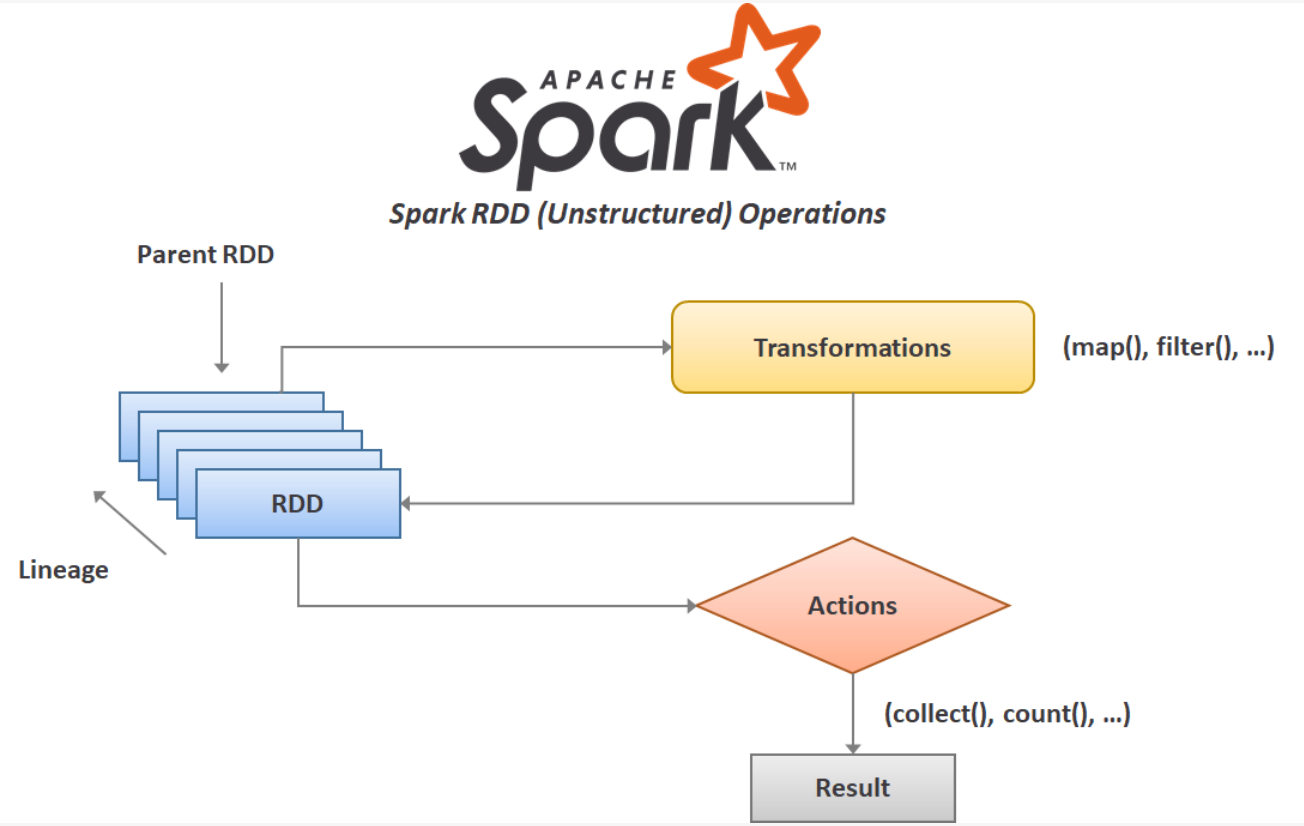
Los DataFrames permiten realizar operaciones complejas como **joins**, **agrupaciones** y **agregaciones**.




| | COUNTRY | CONTINENT | GDPPERCAP | LIFEEXP |
|---|---|---|---|---|
| 1 | AUSTRALIA | OCEANIA | V1 | P1 |
| 2 | CANADA | AMERICAS | V2 | P2 |
| 3 | INDIA | ASIA | V3 | P3 |
| 4 | JAPAN | ASIA | V4 | P4 |
| 5 | PERU | AMERICAS | V5 | P5 |
| 6 | SPAIN | EUROPE | V6 | P6 |

| | COUNTRY | CONTINENT | GDPPERCAP | LIFEEXP |
|---|---|---|---|---|
| 1 | AUSTRALIA | OCEANIA | V1 | P1 |
| 2 | BRAZIL | AMERICAS | V7 | P7 |
| 3 | CANADA | AMERICAS | V2 | P2 |
| 4 | JAPAN | ASIA | V4 | P4 |
| 5 | MONGOLIA | ASIA | V8 | P8 |
| 6 | SINGAPORE | ASIA | V9 | P9 |
| 7 | SPAIN | EUROPE | V6 | P6 |

| | COUNTRY | CONTINENT |
|---|---|---|
| 1 | AUSTRALIA | OCEANIA |
| 2 | CANADA | AMERICAS |
| 3 | INDIA | ASIA |
| 4 | JAPAN | ASIA |
| 5 | PERU | AMERICAS |
| 6 | SPAIN | EUROPE |
| 7 | BRAZIL | AMERICAS |
| 8 | MONGOLIA | ASIA |
| 9 | SINGAPORE | ASIA |



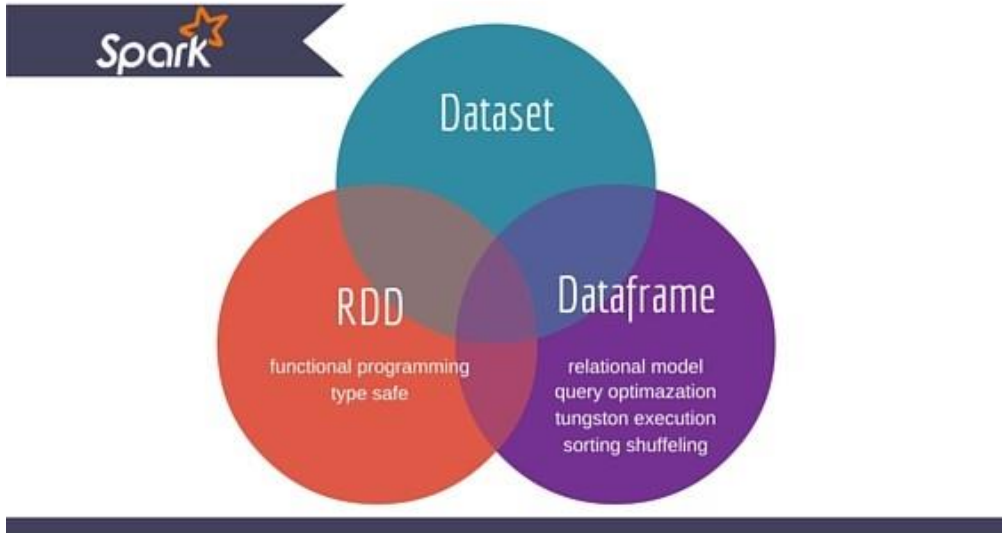

# Introducción a RDDs

Los **Resilient Distributed Datasets** (RDDs) son la **abstracción** principal de **Spark**, permitiendo operaciones distribuidas y resilientes sobre colecciones de objetos.

# Comparación entre DataFrames y RDDs

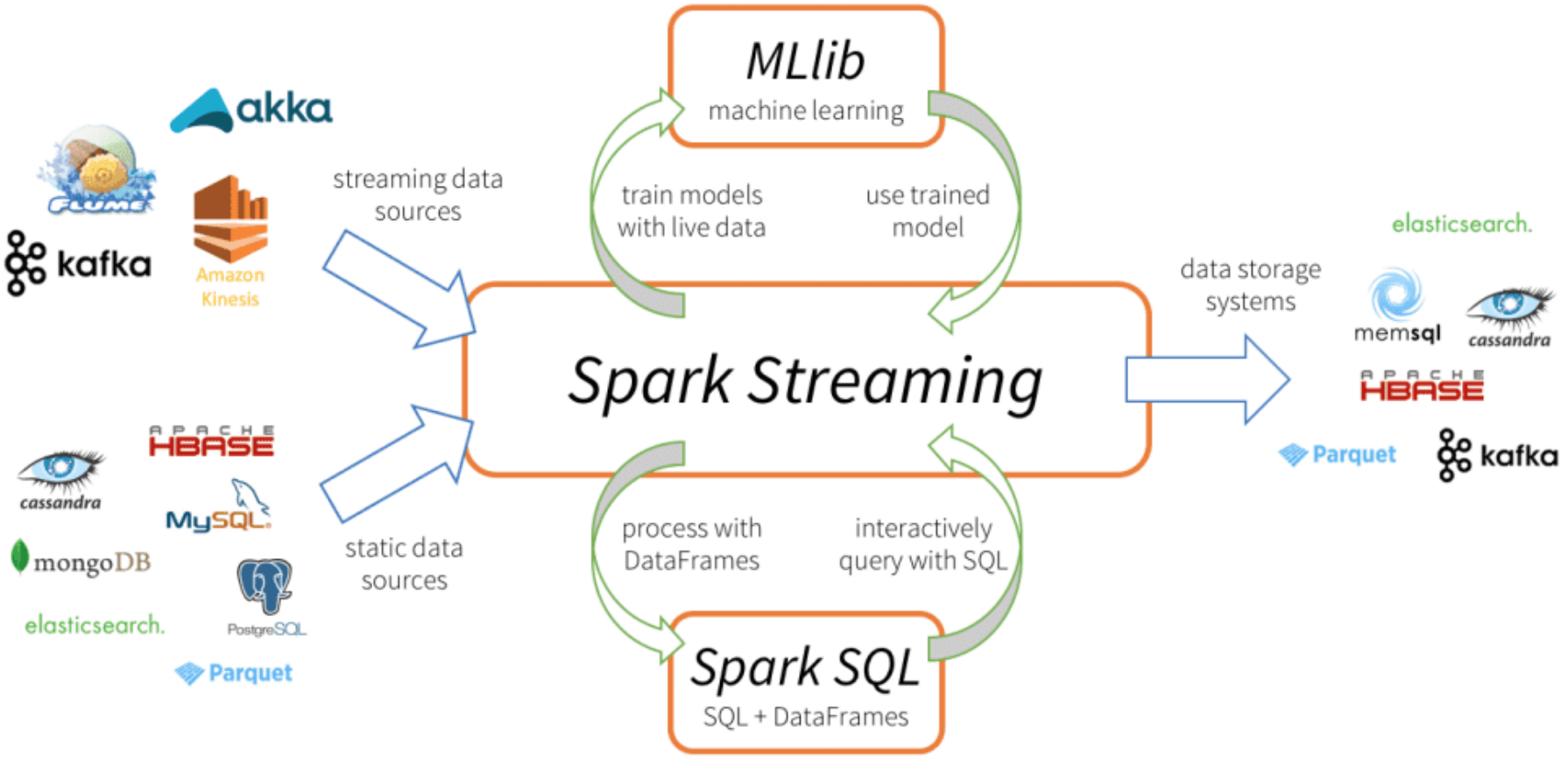
- DataFrames y RDDs ofrecen diferentes niveles de abstracción y optimización, cada uno con su propio conjunto de operaciones.

- **RDD**: Es ideal para aplicaciones donde se requiere un **control detallado** sobre las operaciones de datos y cuando se trabaja con **datos no estructurados** o **complejos** que no encajan bien en un formato tabular.
- **DataFrame**: Es adecuado para la mayoría de los casos de uso, especialmente cuando se trabaja con datos **estructurados o semiestructurados**. Es preferido cuando se desean optimizaciones automáticas y una API más simple.



# Introducción a Spark Streaming

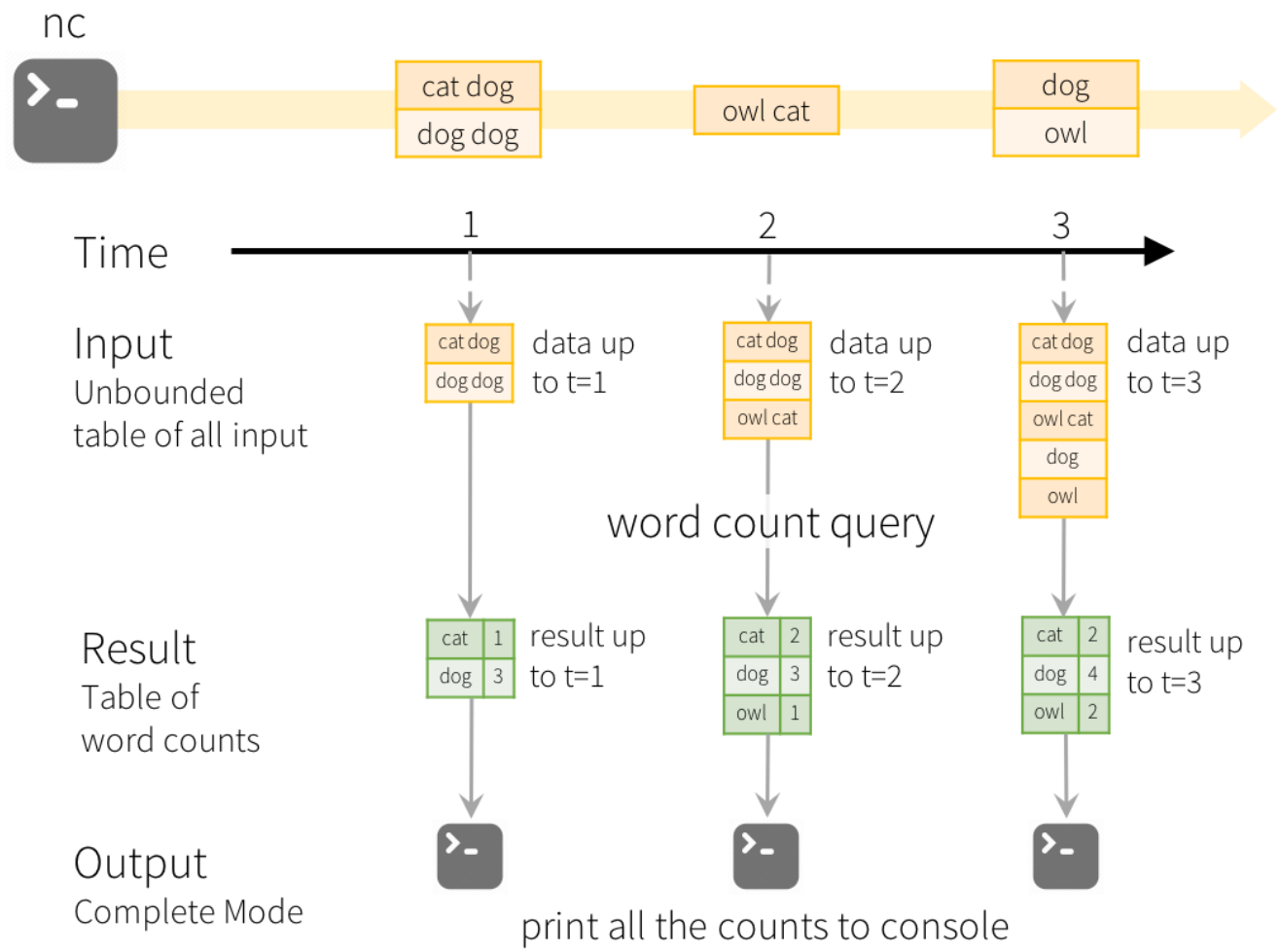
Spark Streaming permite procesar flujos de datos en tiempo real a través de micro-lotes.



# Micro-lotes en Spark Streaming

Los **microlotes (micro-batches)** en Spark Streaming son una técnica fundamental para procesar datos en tiempo real.

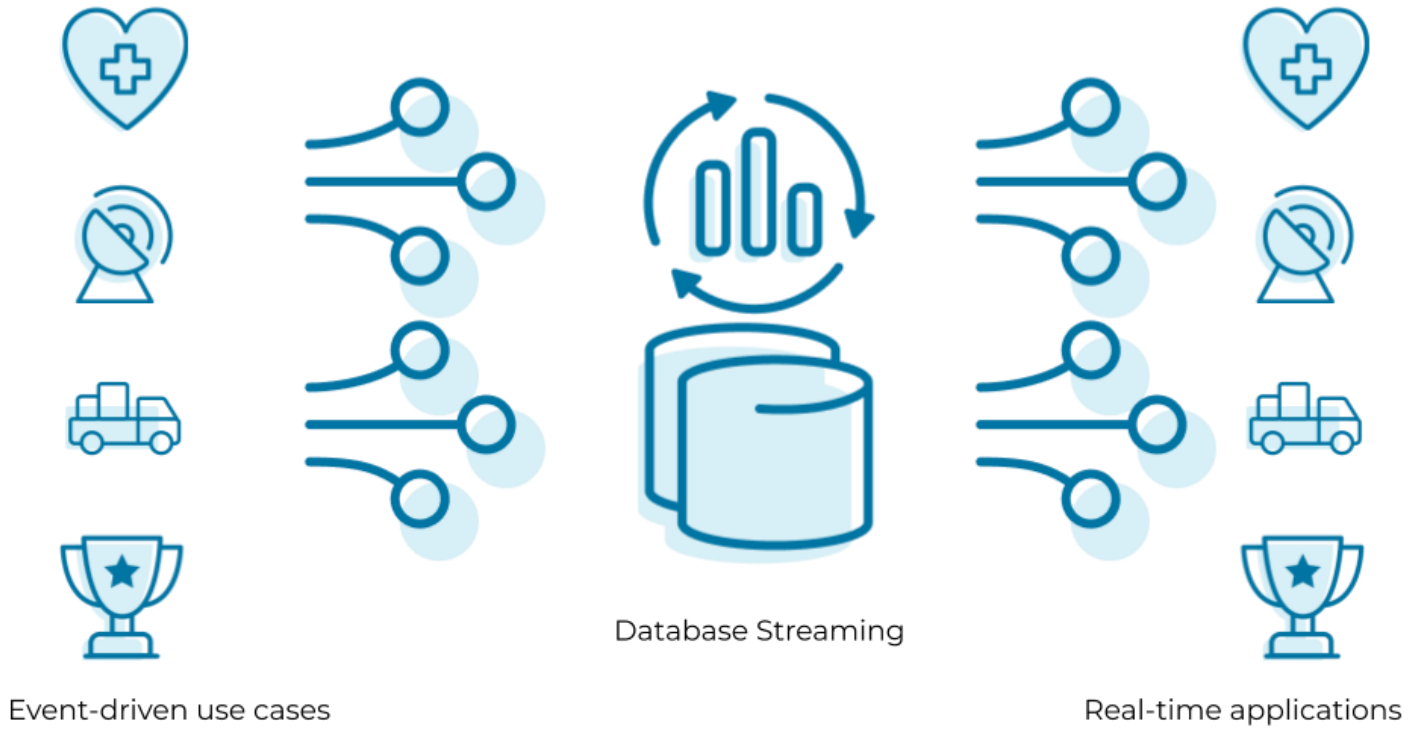
Spark Streaming utiliza un enfoque de procesamiento en tiempo real basado en microlotes, lo que significa que los datos de entrada se dividen en pequeños lotes (o "microlotes") a intervalos regulares y luego se procesan cada lote de **forma secuencial**.

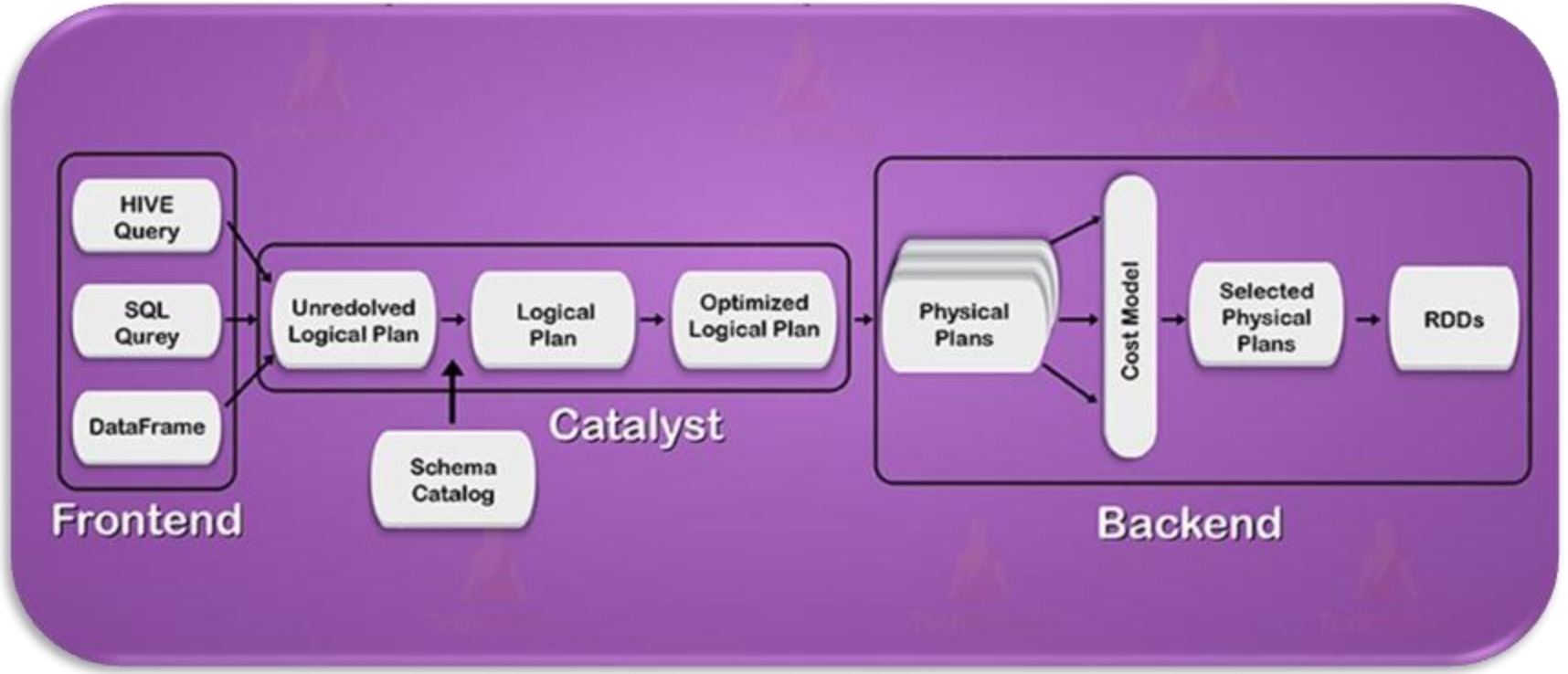



Model of the Quick Example

# Casos de Uso de Spark Streaming

Spark Streaming es utilizado en múltiples industrias para procesamiento de logs, monitoreo de redes sociales, y más.

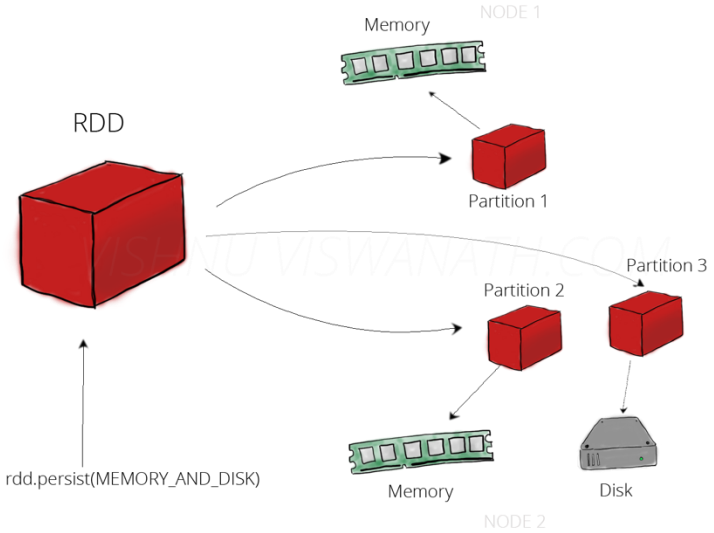


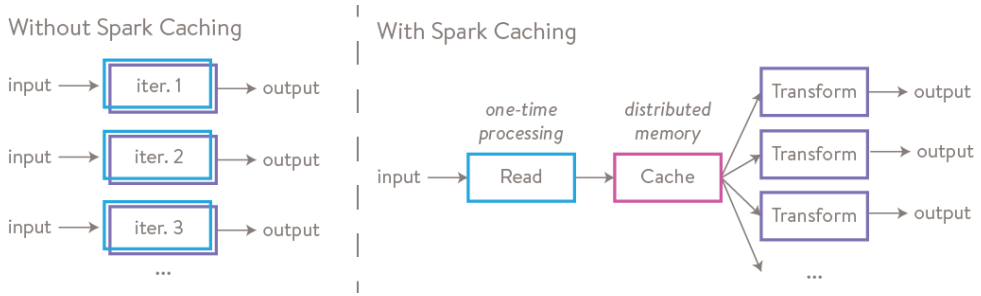



Catalyst Optimizer es un optimizador de consultas basado en reglas que mejora el rendimiento de las operaciones en Spark.

# Introducción a Catalyst Optimizer

# Uso de Caché y Persistencia




El uso de caché y persistencia permite almacenar en memoria los resultados intermedios para mejorar el rendimiento.

# Spark Casos de aplicación y Demos