Ingeniería para el Procesado Masivo de Datos

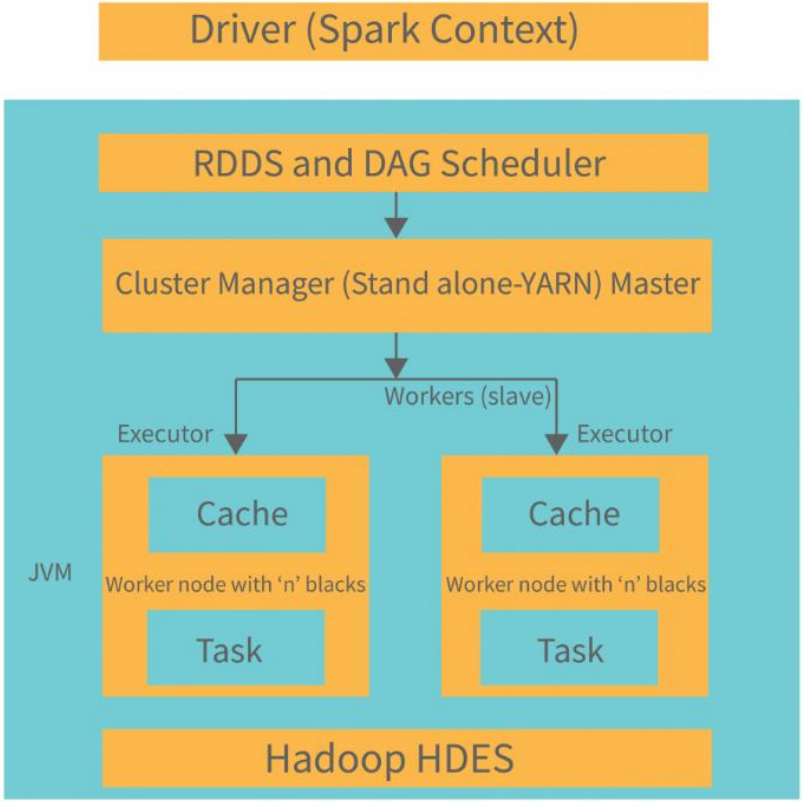
Dra. Ana Beatriz Medina Ruiz

SPARK I

Tema 3: **SPARK I**

- 3.1. Introducción y objetivos Apache Spark
- 3.2. Componentes de Spark
- 3.3. Arquitectura de Spark
- 3.4. Resilient distributed datasets (RDD)
- 3.5. Transformaciones y acciones
- 3.6. Jobs, stages y tasks
- 3.7. Ejemplo completo con RDD

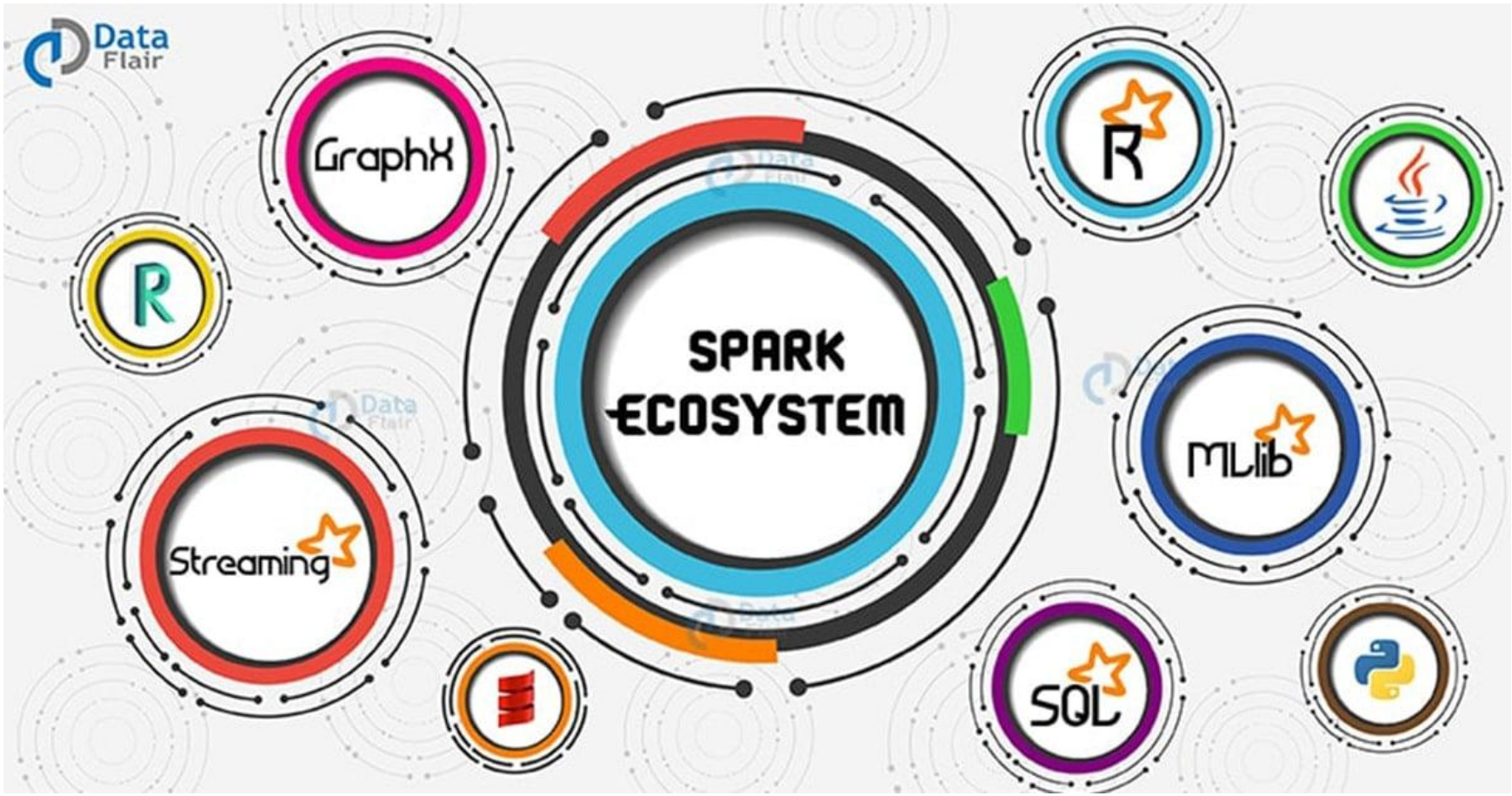
Tema 3: Introducción y Objetivos **SPARK I**



Es un **motor de procesamiento de datos distribuido, rápido y de propósito general**. Fue diseñado para abordar limitaciones de Hadoop MapReduce al ejecutar cálculos en memoria y optimizar flujos de trabajo complejos mediante gráficos de ejecución optimizados (DAG)

- Comprender su importancia en el cambio de las Ciencias de Datos
- Entender porque Spark es útil para procesar grandes volúmenes de datos

Tema 3: **Componentes de Spark**



Tema 3: **Componentes de Spark**

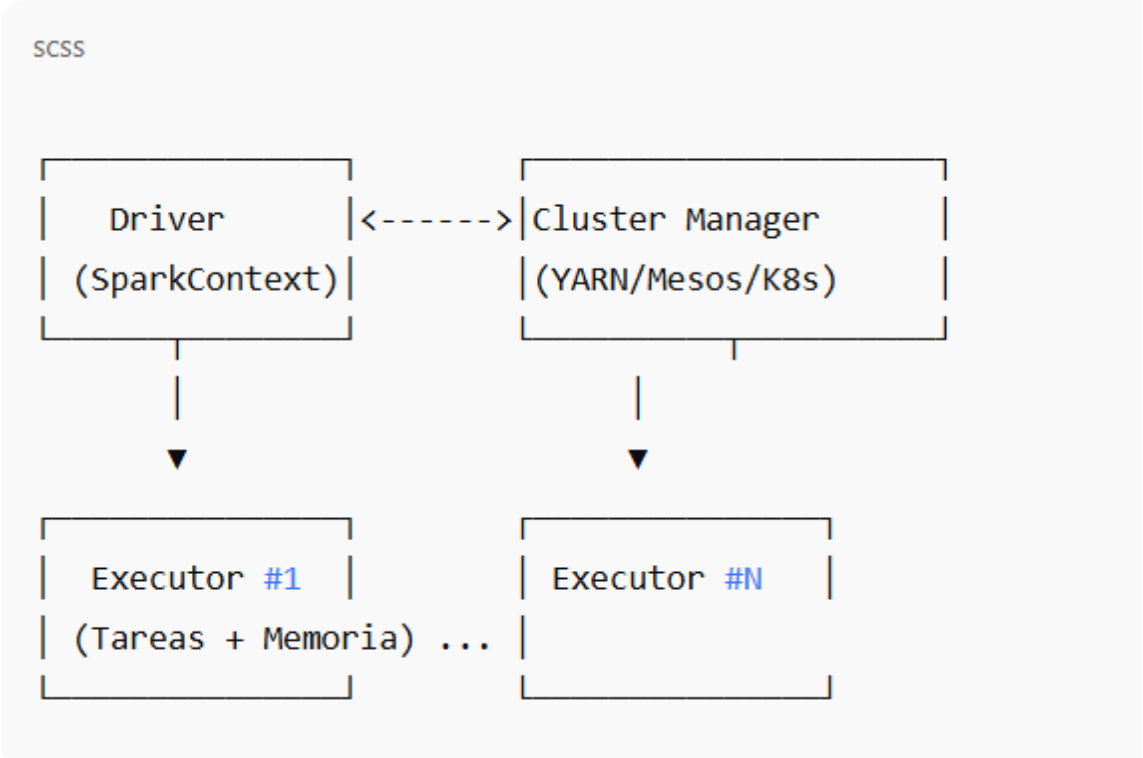
•**Ingesta de Datos**: Se pueden consumir datos desde diversas fuentes como HDFS, S3, Kafka, entre otros.

•**Procesamiento**: Utilizando Spark SQL o Spark Streaming, los datos se procesan en tiempo real o por lotes.

•**Análisis**: Con MLlib, se pueden aplicar modelos de aprendizaje automático para obtener insights.

•**Visualización y Acción**: Los resultados pueden ser visualizados o utilizados para tomar decisiones informadas.

Tema 3: **Arquitectura Spark**



- **Definición del Driver**: El programa principal (Driver) define las tareas y coordina su ejecución.
- **Distribución de Tareas**: Las tareas se distribuyen entre los ejecutores (Executors) en los nodos del clúster.
- **Ejecución Paralela**: Los ejecutores procesan las tareas en paralelo, aprovechando el procesamiento distribuido.
- **Agregación de Resultados**: Los resultados se recopilan y se presentan al usuario final.

Tema 3: **Resilient distributed datasets (RDD)**

RDD es una serie de transformaciones aplicadas a los RDD para crear nuevos RDD. En lugar de replicar datos entre nodos, Spark mantiene información sobre cómo recrear un RDD a partir de sus datos de origen mediante información de linaje.

Linaje: Spark crea un gráfico de linaje de transformaciones para realizar un seguimiento de las dependencias entre RDD. Este gráfico ayuda a Spark a volver a calcular solo los datos perdidos en lugar de volver a ejecutar todo el trabajo.

Tema 3: **Transformaciones y acciones**

Se realizan dos tipos principales de operaciones sobre los RDDS:

Transformaciones: No se ejecuta hasta que no haya una acción que las descencadene. Algunas de las transformaciones más comunes incluyen:

map(): Aplica una función a cada elemento del RDD y devuelve un nuevo RDD con los resultados.

filter(): Filtra los elementos del RDD según una condición dada.

flatMap(): Similar a map(), pero permite devolver un número variable de elementos por entrada.

distinct(): Devuelve un nuevo RDD con elementos únicos.

union(): Combina dos RDDs.

Tema 3: **Transformaciones y acciones**

union(): Combina dos RDDs.

join(): Une dos RDDs de acuerdo a una clave común.

groupBy(): Agrupa los elementos del RDD según una clave dada.

reduceByKey(): Realiza una operación de reducción por clave.

Acciones: Son operaciones que desencadenan la ejecución del proceso de transformación y devuelven un valor o un resultado final. Algunas de las acciones más comunes incluyen:

collect(): Recupera todos los elementos del RDD y los devuelve como una lista en el driver (en la máquina principal).

count(): Devuelve el número de elementos en el RDD.

reduce(): Realiza una operación de reducción en los elementos del RDD.

Tema 3: **Transformaciones y acciones**

union(): Combina dos RDDs.

join(): Une dos RDDs de acuerdo a una clave común.

groupBy(): Agrupa los elementos del RDD según una clave dada.

reduceByKey(): Realiza una operación de reducción por clave.

Acciones: Son operaciones que desencadenan la ejecución del proceso de transformación y devuelven un valor o un resultado final. Algunas de las acciones más comunes incluyen:

collect(): Recupera todos los elementos del RDD y los devuelve como una lista en el driver (en la máquina principal).

count(): Devuelve el número de elementos en el RDD.

reduce(): Realiza una operación de reducción en los elementos del RDD.

Tema 3: **Jobs, stages y tasks**

Job: Es el nivel más alto de procesamiento en Spark. Se inicia cuando ejecutas una **acción** (count(), collect(), save(), etc.) sobre un RDD o DataFrame.

Stage es una etapa lógica compuesta por transformaciones que no requieren **shuffle** (narrow dependencies). Cuando aparece una transformación **wide** (como reduceByKey, groupBy, join), Spark termina el stage actual y comienza uno nuevo.

Cada Job se subdivide en múltiples stages, determinados por los puntos de shuffle en el grafo de ejecución (DAG).

Tema 1: Ejemplo Completo con RDD

Práctica para Iniciar con HDF

- Combina dos tecnologías para trabajar con Big Data y contenedorización
- RDD: Abstrae datos y permite realizar transformaciones y acciones de manera distribuida sobre grandes volúmenes de datos.
- Docker: Excelente opción para ejecutar aplicaciones en entornos de producción, por ser contenedores fáciles de distribuir, ligeros y portátiles.
- Docker se Integra con RDD: Una vez que Spark esta dentro de Docker, usan RDDs para procesar grandes volúmenes de datos distribuidos de manera eficiente desde la entrada hasta las salidas dentro de un clúster de contenedores Docket.

www.unir.net