Programa de formación
MACHINE LEARNING AND DATA SCIENCE MLDS

Facultad de INGENIERÍA

UNIVERSIDAD NACIONAL DE COLOMBIA

Módulo 3
Big Data

Unidad 5

Procesamiento *Distribuido* II

Clase sincrónica

Facultad de INGENIERÍA
UNIVERSIDAD NACIONAL DE COLOMBIA

Bienvenida

Jorge Eliécer Camargo Mendoza, PhD.

UNIVERSIDAD NACIONAL DE COLOMBIA

https://dis.unal.edu.co/~jecamargom/

jecamargom@unal.edu.co

Departamento de Ingeniería de Sistemas e Industrial

Facultad de Ingeniería

Universidad Nacional de Colombia

Sede Bogotá

Tabla de contenidos

Objetivos de aprendizaje

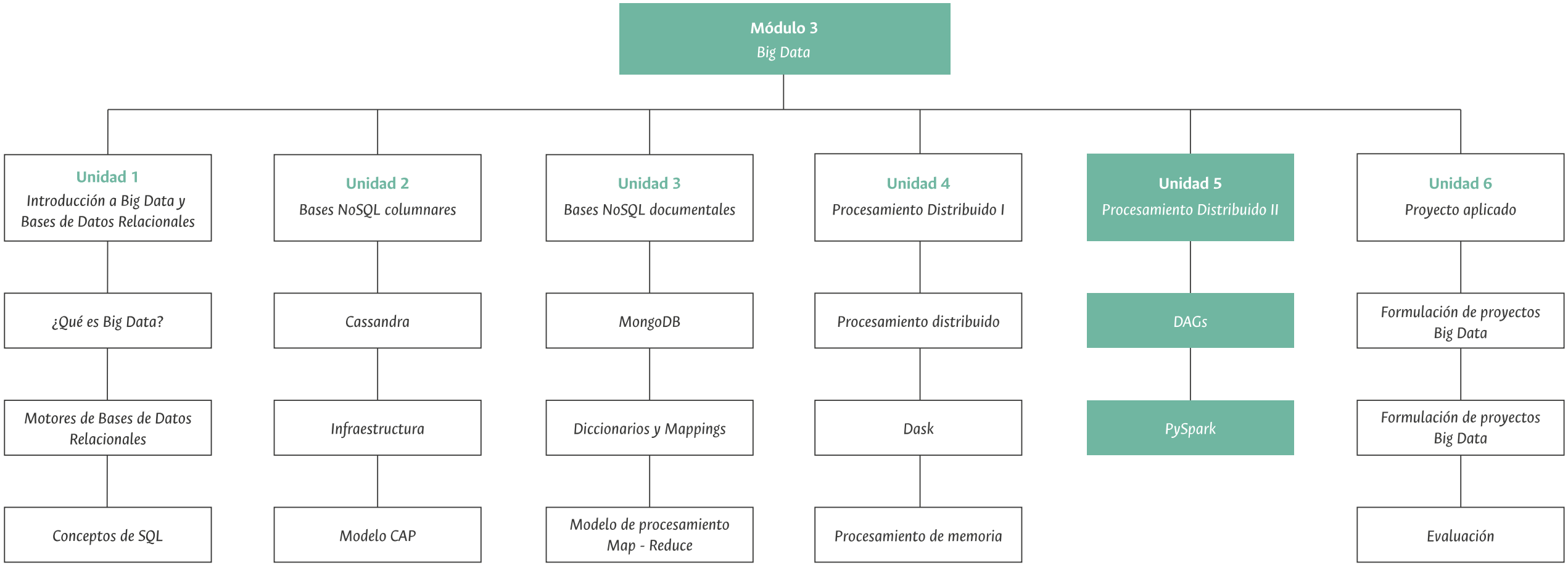
Unidad 5 – Procesamiento *Big Data* II

Al finalizar la unidad usted deberá ser capaz de:

Entender conceptos de computación distribuida con operaciones optimizadas automáticamente por medio de grafos dirigido acíclicos (DAGs).

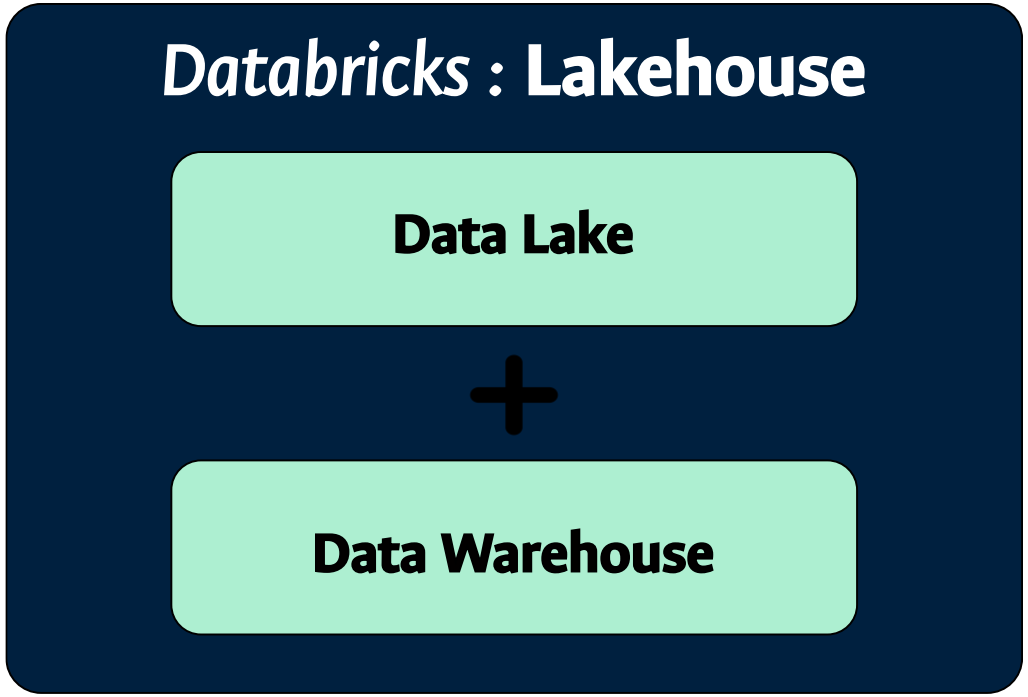
Utilizar dataframes distribuidos y optimizados desde la librería Spark para el manejo de grandes cantidades de datos tabulares.

Mapa de contenidos de la unidad



1 Databricks

- Una plataforma simple para **unificar** del cliente sus *datos*, *análisis* y trabajo desarrollado usando técnicas de *inteligencia artificial.*
- *Databricks* es uno de los más exitosos proyecto ***open-source*** en el *data space.*
- **Su popularidad se debe a la *unificación* de las herramientas de *Data Lake* y *Data Warehouse.***



Databricks

Importancia

databricks

5000+ clientes alrededor del mundo

30 millones+ descargas mensuales

450+ colaboradores alrededor del mundo como: ***AWS, Microsoft Azure*** y ***Google Cloud***

Databricks

Trabajo colaborativo

databricks

Analista de datos

Herramientas de trabajo colaborativo de cada uno de los roles en un mismo lugar

Ingeniero de datos

Científico de datos

Databricks provee herramientas colaborativas como:

1. **Notebooks**
2. **Dashboards**
3. **Modelos**
4. **Datasets**

APACHE Spark™

Con los *notebooks* es possible manejar ***Spark***

2 Apache Spark

Las organizaciones en la actualidad necesitan procesar grandes cantidades de datos (*Big data*), en diferentes formatos y de manera muy rápida. Esto es imposible aplicarlo con los métodos de procesamiento de datos tradicionales, como los que se dan de manera *centralizada*.

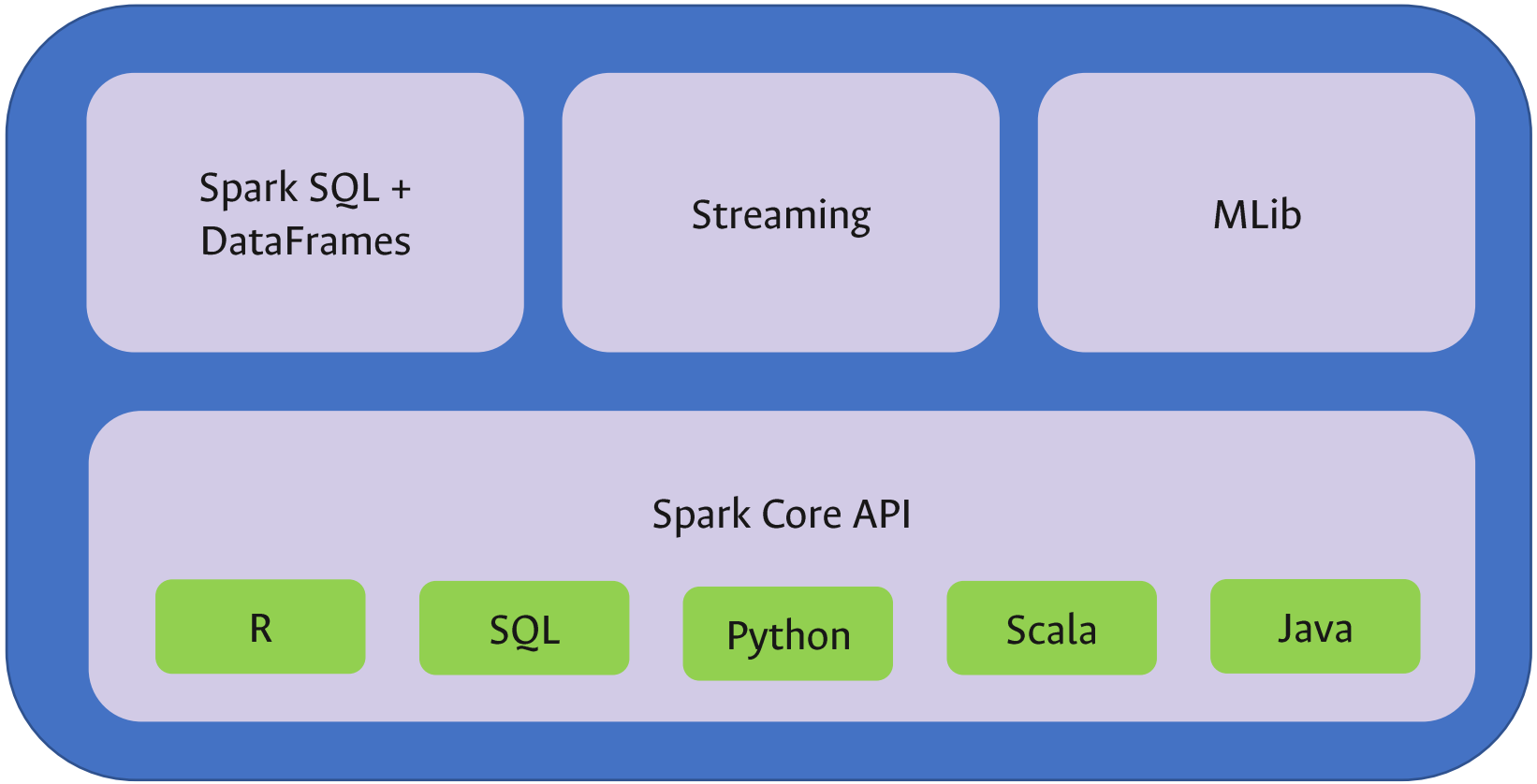
Este es el problema que aborda Spark, procesa **grandes cantidades de datos, en diferentes formatos, rápido y de manera *distribuida*.**

Apache Spark

Introducción

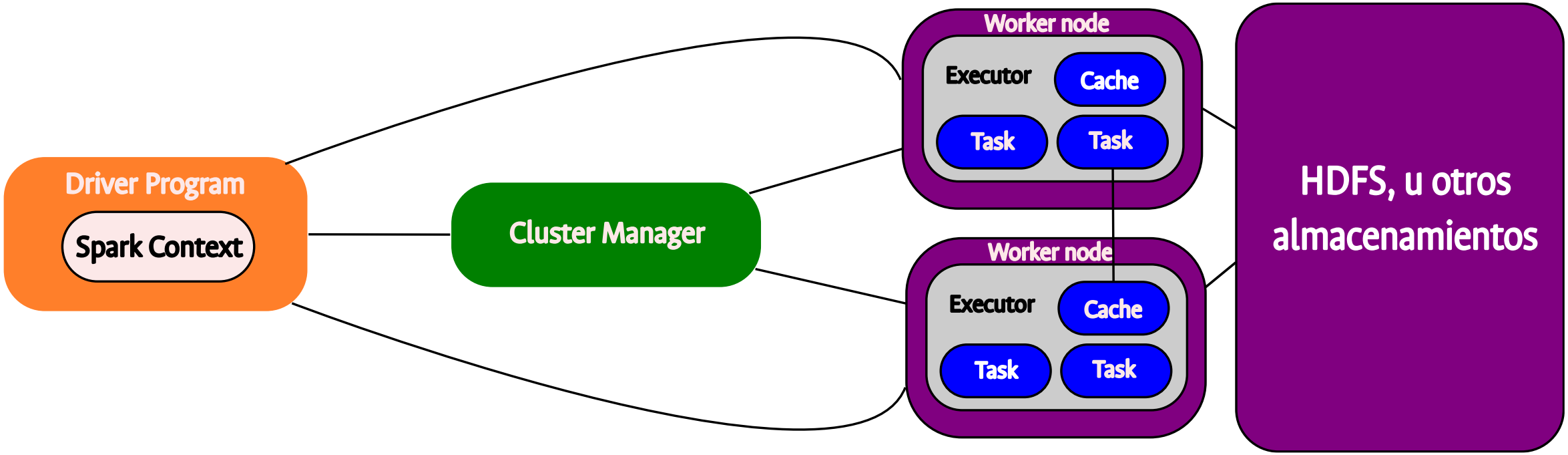
Apache Spark es un motor analítico unificado para procesamiento de datos a gran escala. Provee APIs de alto nivel en *Java*, *Scala*, *Python* y *R*, además de un motor optimizado que soporta un gran conjunto de herramientas de alto nivel, en el que se incluyen *Spark* SQL, para procesamiento de datos estructurados y SQL, *MLib* para tareas de aprendizaje de máquina; *GraphX*, para procesamiento de datos y *Structured Streaming* para tareas de computación incremental y procesamiento de flujos (***The Apache Software Foundation, s.f.***).

Spark API


Spark SQL + DataFrames
Streaming
MLib
Spark Core API
R
SQL
Python
Scala
Java

Apache Spark

Arquitectura de Spark



Arquitectura de Spark – Video

3 Spark SQL

Spark SQL es un módulo para procesamiento de datos estructurados con múltiples interfaces:

SQL

La sintaxis SQL es possible usarla directamente en los *Notebooks* con *Spark*

DataFrame API
Python, Scala, Java, R

La sintaxis de SQL puede implementarse también con DataFrames. Existen APIs para *Python*, *Scala*, *Java* y *R*.

Spark SQL

Queries

SQL

```sql
SELECT id, resultado
FROM examenes
WHERE resultado > 70
ORDER BY resultado
```

Comando SQL en una celda de un *Notebook* de *Databricks* usando *Spark*

Python DataFrame API (*PySpark*)

```python
spark.table("examenes")
    .select("id","resultado")
    .where("resultado > 70")
    .orderBy("resultado")
```

Comando de *PySpark* en una celda de un *Notebook* de *Databricks*

4 Python DataFrame API: PySpark

- Un *DataFrame* es una colección de datos distribuidos organizados en columnas nombradas.
- Los *DataFrames* son equivalentes a una tabla en una base de datos relacional o a los objetos *DataFrame* de la librería *Pandas* de *Python*.
- Provee una API de manipulación de alto nivel.
- Existen diferentes maneras de crear *DataFrames* (***The Apache Software Foundation. s.f.***).
  - Paralelizando colecciones en el *driver program*.
  - Desde archivos de texto con datos estructurados (CSV, JSON).
  - Tablas en *Hive* o bases de datos externas.
  - *RDDs* existentes.

Python DataFrame API: *PySpark*

Operaciones

Transformaciones

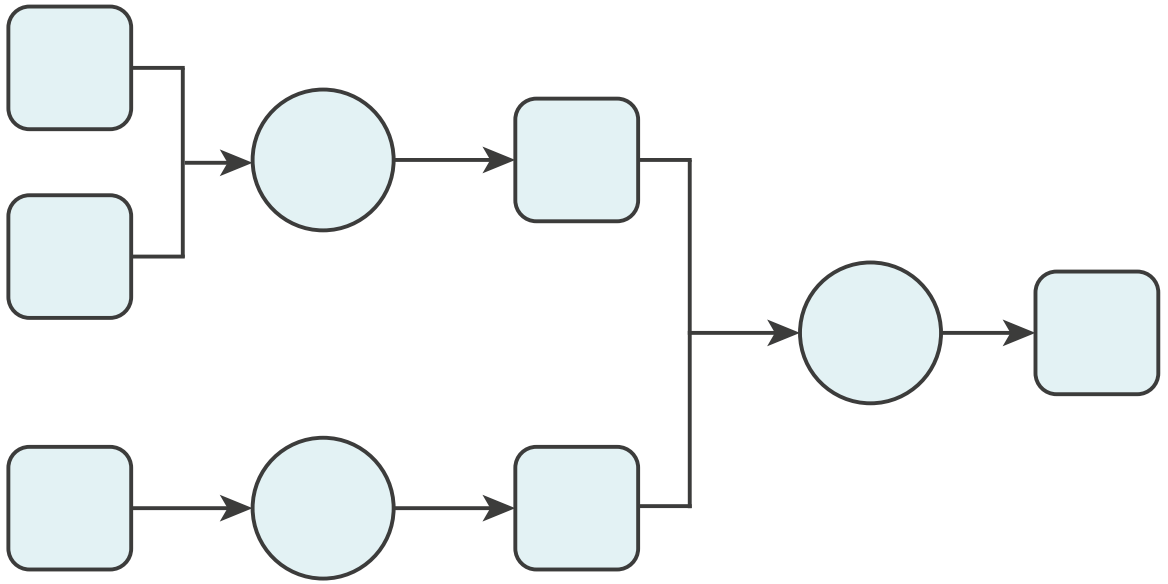
- Crean un nuevo conjunto de datos, que se basan en conjuntos de datos existentes.
- Algunas de las transformaciones más comunes son *map*, *filter*, *select*, *join*, *union*, *distinct*, *groupBy*, *orderBy*
- Se ejecutan de manera *Lazy* (perezosa), es decir, sus resultados no se computan al ser definidas. *Spark* recuerda las transformaciones y solo las ejecuta cuando una acción que se está ejecutando necesita el resultado de dicha transformación.

Acciones

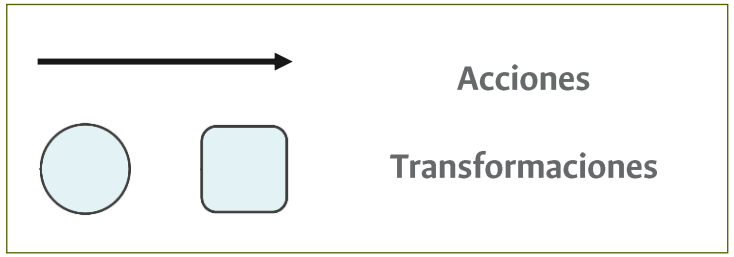
- Son comandos que se ejecutan sobre los conjuntos de datos y retornan un valor.
- Generalmente ejecutan tareas de agregación, análisis, conteos, etc.
- Algunas acciones comunes son *show*, *count*, *collect*, *reduce*, *save*, *avg*
- Se ejecutan inmediatamente y en primer momento ejecutan cualquier transformación que esté pendiente en el conjunto de datos (*The Apache Software Foundation*. s.f.).

Python DataFrame API: *PySpark*

Grafo Acíclico dirigido



Grafo Acíclico Dirigido



Python DataFrame API: *PySpark*

Ejemplo de creación

```python
datos = [["1", "Daniel", "Perez"],
         ["1", "Nicolas", "Roldan"],
         ["1", "Marcela", "Vargas"],
         ["2", "Leonardo", "Rodriguez"],
         ["3", "Luisa", "Cardona"],
         ["4", "Tani", "Forero"]]

schema = ['id', 'nombre', 'apellido']

#Creando el DataFrame
dataframe = spark.createDataFrame(datos, schema)
```

id	nombre	apellido
1	Daniel	Perez
1	Nicolas	Roldan
1	Marcela	Vargas
2	Leonardo	Rodriguez
3	Luisa	Cardona
4	Tani	Forero

Python DataFrame API: *PySpark*

Consulta: Todos los datos

```
dataframe.show()
```

Acción para mostrar el *DataFrame*. Debe ser usada después de cada *transformación*.

```
dataframe.select("*").show()
```

Pseudo SQL en los DataFrames de PySpark

id	nombre	apellido
1	Daniel	Perez
1	Nicolas	Roldan
1	Marcela	Vargas
2	Leonardo	Rodriguez
3	Luisa	Cardona
4	Tani	Forero

Python DataFrame API: *PySpark*

Consulta: Datos de dos columnas en específico

```
dataframe.select("id","nombre").show()
```

id	nombre	apellido
1	Daniel	Perez
1	Nicolas	Roldan
1	Marcela	Vargas
2	Leonardo	Rodriguez
3	Luisa	Cardona
4	Tani	Forero

Python DataFrame API: *PySpark*

Consulta: Una fila en específico

```
dataframe.select("*").where("id == 3").show()
```

id	nombre	apellido
1	Daniel	Perez
1	Nicolas	Roldan
1	Marcela	Vargas
2	Leonardo	Rodriguez
3	Luisa	Cardona
4	Tani	Forero

Python DataFrame API: *PySpark*

Consulta: Valores diferentes en una columna

```
dataframe.select("id").distinct().show()
```

id	nombre	apellido
1	Daniel	Perez
1	Nicolas	Roldan
1	Marcela	Vargas
2	Leonardo	Rodriguez
3	Luisa	Cardona
4	Tani	Forero

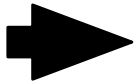
id
1
2
3
4

Python DataFrame API: *PySpark*

Actualizar: Todos los valores de una columna

```
dataframe_auxiliar = dataframe.withColumn("id",dataframe["id"]*10)
dataframe_auxiliar.show()
```

id	nombre	apellido
1	Daniel	Perez
1	Nicolas	Roldan
1	Marcela	Vargas
2	Leonardo	Rodriguez
3	Luisa	Cardona
4	Tani	Forero



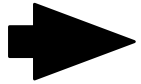
id	nombre	apellido
10	Daniel	Perez
10	Nicolas	Roldan
10	Marcela	Vargas
20	Leonardo	Rodriguez
30	Luisa	Cardona
40	Tani	Forero

Python DataFrame API: *PySpark*

Actualizar: Unas filas en específico

```
dataframe_auxiliar = dataframe.withColumn("id",when(dataframe["id"]==1,-1).otherwise(dataframe["id"]))
dataframe_auxiliar.show()
```

id	nombre	apellido
10	Daniel	Perez
10	Nicolas	Roldan
10	Marcela	Vargas
20	Leonardo	Rodriguez
30	Luisa	Cardona
40	Tani	Forero



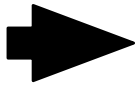
id	nombre	apellido
-1	Daniel	Perez
-1	Nicolas	Roldan
-1	Marcela	Vargas
20	Leonardo	Rodriguez
30	Luisa	Cardona
40	Tani	Forero

Python DataFrame API: *PySpark*

Borrar: Borrar unca columna en específico

```
dataframe_auxiliar = dataframe.drop("apellido")
dataframe_auxiliar.show()
```

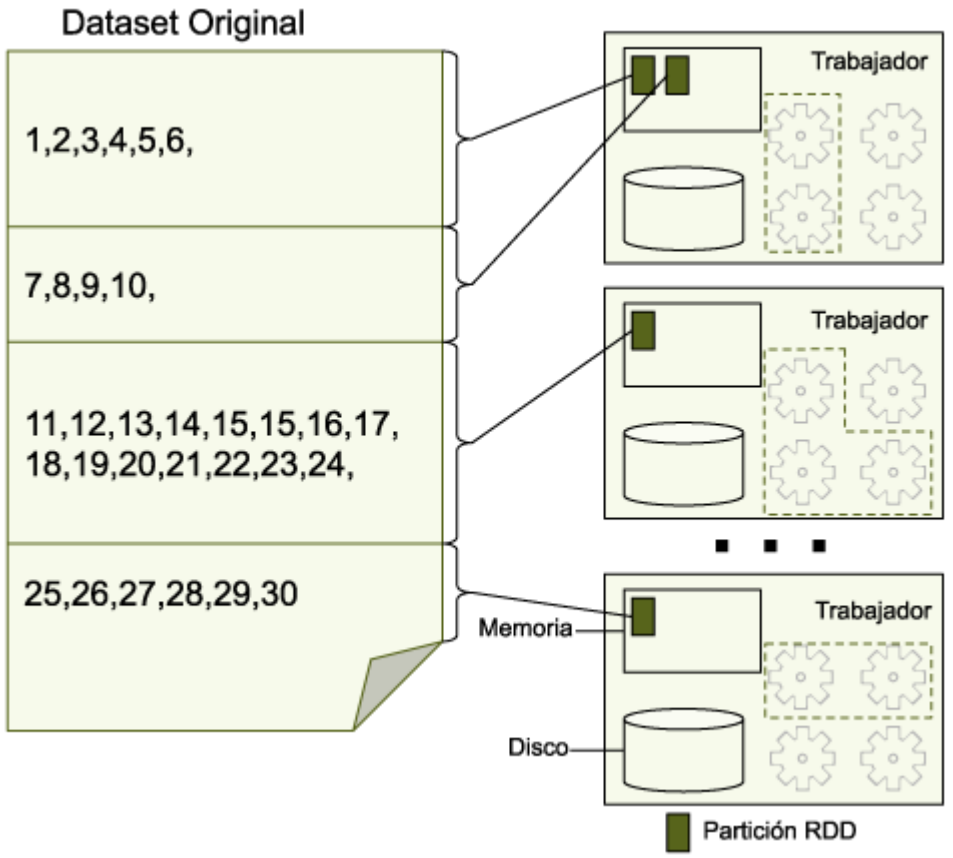
id	nombre	apellido
-1	Daniel	Perez
-1	Nicolas	Roldan
-1	Marcela	Vargas
20	Leonardo	Rodriguez
30	Luisa	Cardona
40	Tani	Forero



id	nombre
1	Daniel
1	Nicolas
1	Marcela
2	Leonardo
3	Luisa
4	Tani

5 Apache Spark: RDD

- Un **RDD** es una colección de elementos tolerante a fallos que puede ser operada en paralelo.
- Proporcionan la API de bajo nivel para la manipulación de datos.
- Un **RDD** es inmutable después de haber sido creado.
- Existen tres maneras de crear **RDDs**:
  - Paralelizando colecciones en el *driver program*.
  - Cargando datos desde HDFS, *Cassandra*, S3 o Sistemas de archivos locales.
  - Obtención de *RDDs* mediante transformaciones a los *RDDs* existentes.
- Un *RDD* crea particiones de los datos que se almacenan en la memoria de los *workers* (*The Apache Software Foundation.* s.f.).

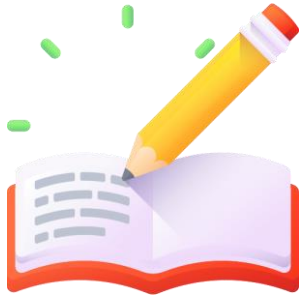


6 Spark MLib

Mlib es una librería de Machine Learning escalable de Apache Spark que es possible usarse en Java, Scala, Python y R. Posee algoritmos de alta calidad que son 100 veces más rápidos que *MapReduce*, como:

1. **Algoritmos de Clasificación**: Regrésión logística, Naive Bayes, ...
2. ***Algoritmos de regresión:*** regresión lineal generalizada, regresión de supervivencia, ...
3. Árboles de decisión, bosques aleatorios y árboles potenciados por gradiente.
4. **Algoritmos de recomendación:** alternancia de mínimos cuadrados (ALS : Alterning Least Squares)
5. **Algoritmos de clusterización:** K-means, mezcla de gausianas (GMMs),...
6. **Modelado de temas latentes:** asignación latente de Dirichlet (LDA: Latent Dirichlet allocation)

Actividades



Para esta unidad existen tres actividades:

- Taller guiado Apache Spark con Jupyter Notebook
- Taller guiado Apache Spark con Databricks
- Actividad calificable: Reporte sobre taller guiado de Apache Spark con Databricks

Despedida

¡Gracias por su atención!

Jorge Eliécer Camargo Mendoza, PhD.

UNIVERSIDAD NACIONAL DE COLOMBIA

https://dis.unal.edu.co/~jecamargom/

jecamargom@unal.edu.co

Departamento de Ingeniería de Sistemas e Industrial

Facultad de Ingeniería

Universidad Nacional de Colombia

Sede Bogotá

Referencias

The Apache Software Foundation. (s. f.). Apache Hive.
https://hive.apache.org/

The Apache Software Foundation. (s. f.). Apache Hive.
https://cwiki.apache.org/confluence/display/Hive#Home-HiveDocumentation

The Apache Software Foundation. (s. f.). Apache Spark.
https://spark.apache.org/docs/latest/quick-start.html

The Apache Software Foundation. (s. f.). Apache Spark.
https://spark.apache.org/

Derechos de imágenes

Romainrr (s.f) *HiveQL Editor in Hue* [imagen]
https://commons.wikimedia.org/wiki/File:HiveQL_Editor_in_Hue.png

The Apache Software Foundation. (s.f.). *Hive logo.* [Logo].
https://hive.apache.org/images/hive_logo_medium.jpg

The Apache Software Foundation (s.f.). *Apache Spark logo.* [Logo].
https://spark.apache.org/images/spark-logo-trademark.png

Flaticon Icons:

- https://bit.ly/3gNqKFh
- https://bit.ly/3XDaRC7
- https://bit.ly/3OLdPAq

Créditos

Facultad de
INGENIERÍA

Autores
Jorge Eliécer Camargo Mendoza, PhD

Asistente docente
Leonardo Avendaño Rocha
Alberto Nicolai Romero Martínez

Diseño instruccional
Claudia Patricia Rodríguez Sánchez

Diseño gráfico
Clara Valeria Suárez Caballero
Milton R. Pachón Pinzón
Brian Chaparro Cetina

Diagramadora PPT
Daniela Duque

Fecha
2022-II