Sesión: Procesamiento distribuido de datos

Apache Spark es un motor de procesamiento distribuido que permite manejar grandes volúmenes de datos de manera rápida y eficiente.

por Kibernum Capacitación S.A.

CONFIGURACIÓN, CONEXIÓN Y CONTEXTO DE SPARK

La configuración y creación del contexto de Spark es el primer paso para ejecutar cualquier tarea. El SparkContext es la entrada principal que permite interactuar con las funcionalidades principales de Spark, y es responsable de coordinar la ejecución de tareas en un clúster distribuido. Al usar SparkSession, que es una evolución de SparkContext introducida en Spark 2.0, se simplifica la inicialización del entorno y proporciona acceso unificado a diferentes APIs (RDD, DataFrame y SQL).

¿Por qué es necesario?

- Proporciona la interfaz para conectarse al clúster y controlar la distribución de tareas.
- Permite definir configuraciones clave como memoria, núcleos y parámetros de ejecución.
- Es esencial para ejecutar transformaciones y acciones en los RDDs y DataFrames.
- Sin una sesión o contexto de Spark, no es posible procesar datos distribuidos en un entorno Spark.

```python
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("MiPrimerContextoSpark") \
    .master("local[*]") \
    .config("spark.executor.memory", "2g") \
    .getOrCreate()

    # .appName  Nombre de la aplicación
    # .master Ejecutar en modo local usando todos los núcleos disponibles
    # .config Configuración de memoria del ejecutor

# Mostrar información sobre la sesión
print("Sesión de Spark creada exitosamente")
print(f"Nombre de la aplicación: {spark.sparkContext.appName}")
print(f"Master: {spark.sparkContext.master}")
print(f"Versión de Spark: {spark.version}")
```

Código completo: https://colab.research.google.com/drive/12a9sEzDgQuLiLT31LxONvnJe4l_uwgoh?usp=sharing

RDD (RESILIENT DISTRIBUTED DATASET)

Qué es un RDD

Un RDD (Resilient Distributed Dataset) es una estructura de datos fundamental en Apache Spark que representa una colección inmutable de elementos distribuidos a través de un clúster. Los RDDs permiten realizar operaciones de procesamiento de datos de forma paralela y eficiente mediante transformaciones y acciones.

Creación y carga de un RDD

- Desde una lista o colección: Crea un RDD a partir de datos locales en memoria.
- Desde un archivo externo: Lee datos desde archivos de texto, CSV, JSON, etc.
- Desde una base de datos o fuente externa: Usando conectores específicos.

Almacenamiento de un RDD

El almacenamiento de un RDD en Spark permite optimizar el rendimiento para futuras operaciones. Esto se hace mediante las funciones:

- persist() – Almacena el RDD en memoria o disco según el nivel de persistencia.
- cache() – Almacena el RDD en memoria para acceso rápido.

Almacenamiento de un RDD

El almacenamiento de un RDD en Spark permite optimizar el rendimiento para futuras operaciones. Esto se hace mediante las funciones:

- persist() – Almacena el RDD en memoria o disco según el nivel de persistencia.
- cache() – Almacena el RDD en memoria para acceso rápido.

```
# Almacenar un RDD en memoria (cache)
rdd_lista.cache()

# Persistir un RDD en disco
from pyspark import StorageLevel
rdd_api.persist(StorageLevel.DISK_ONLY)

# Mostrar el almacenamiento
print(f"RDD en cache: {rdd_lista.is_cached}")
print(f"RDD persistido: {rdd_api.getStorageLevel()}")
```

Código completo:

https://colab.research.google.com/drive/12a9sEzDgQuLiLT31LxONvnJe4l_uwgoh?usp=sharing

TRANSFORMACIONES EN SPARK

Una transformación en Apache Spark es una operación que toma un RDD o DataFrame como entrada y produce un nuevo RDD o DataFrame como resultado. Las transformaciones son perezosas (lazy), lo que significa que no se ejecutan inmediatamente cuando se definen, sino que se evalúan solo cuando una acción es invocada.

Tipos de Transformaciones

- Narrow Transformations (Transformaciones Estrechas): Los datos de una partición específica del RDD de entrada son transformados en una sola partición del RDD de salida. Ejemplos: map(), filter(), flatMap().
- Wide Transformations (Transformaciones Amplias): Los datos de múltiples particiones del RDD de entrada son transformados y repartidos en múltiples particiones del RDD de salida. Ejemplos: groupBy(), reduceByKey(), sortBy().

Principales métodos

- filter() – Filtra los elementos que cumplan con una condición
- map() – Aplica una función a cada elemento del RDD
- flatMap() – Descompone oraciones en palabras individuales.
- sample() – Toma una muestra aleatoria del RDD sin reemplazo.
- union() – Une dos RDDs (se permiten duplicados).
- distinct() – Elimina valores duplicados del RDD unido.
- sortBy() – Ordena el RDD de forma ascendente o descendente.

Transformaciones en spark

```python
from pyspark.sql import SparkSession

# Crear una sesión de Spark
spark = SparkSession.builder \
    .appName("TransformacionesSpark") \
    .master("local[*]") \
    .getOrCreate()

# Crear un RDD de ejemplo
datos = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
rdd = spark.sparkContext.parallelize(datos)

# 1. filter(): Filtrar valores mayores a 5
rdd_filter = rdd.filter(lambda x: x > 5)

# 2. map(): Multiplicar cada número por 2
rdd_map = rdd.map(lambda x: x * 2)

# 3. flatMap(): Descomponer frases en palabras
oraciones = ["Hola mundo", "Apache Spark es increíble"]
rdd_oraciones = spark.sparkContext.parallelize(oraciones)
rdd_flatmap = rdd_oraciones.flatMap(lambda x: x.split(" "))
```

Código completo:

https://colab.research.google.com/drive/1Nlr-L3O7VG5XelW6t05fIej-F9nm5WFl?usp=sharing

ACCIONES EN SPARK

Una acción en Apache Spark es una operación que desencadena la ejecución del DAG (Directed Acyclic Graph) y devuelve un resultado al controlador o escribe los datos en un sistema de almacenamiento. A diferencia de las transformaciones, que son perezosas (lazy), las acciones fuerzan la evaluación de los RDDs o DataFrames.

Principales acciones:

Acción	Descripción	Resultado
collect()	Devuelve todos los elementos del RDD como una lista.	Lista
take(n)	Devuelve los primeros n elementos del RDD.	Lista
top(n)	Devuelve los primeros n elementos ordenados.	Lista
count()	Devuelve el número total de elementos del RDD.	Entero
reduce()	Aplica una función de reducción a los elementos.	Valor único
foreach()	Aplica una función a cada elemento sin devolver nada.	Ninguno
saveAsTextFile()	Escribe los datos del RDD en un archivo de texto.	Archivo
takeSample()	Devuelve una muestra aleatoria de elementos.	Lista

Código completo: https://colab.research.google.com/drive/1JHgdvW0SUNTAAjsZkKay8nqOVEWOa8DV?usp=sharing

Acciones en Spark:

```python
from pyspark.sql import SparkSession

# Crear una sesión de Spark
spark = SparkSession.builder \
    .appName("AccionesSpark") \
    .master("local[*]") \
    .getOrCreate()

# Crear un RDD de ejemplo
datos = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
rdd = spark.sparkContext.parallelize(datos)

# 1. collect(): Obtener todos los elementos del RDD
resultado_collect = rdd.collect()
print("Resultado de collect():", resultado_collect)

# 2. take(n): Obtener los primeros 5 elementos
resultado_take = rdd.take(5)
print("Resultado de take(5):", resultado_take)
```

Codigo completo:

https://colab.research.google.com/drive/1JHgdvW0SUNTAAjsZkKay8nqOVEWOa8DV?usp=sharing

JOB SPARK Y ACTIVIDAD PRÁCTICA GUIADA

JOB SPARK

Es secuencia de transformaciones de datos.

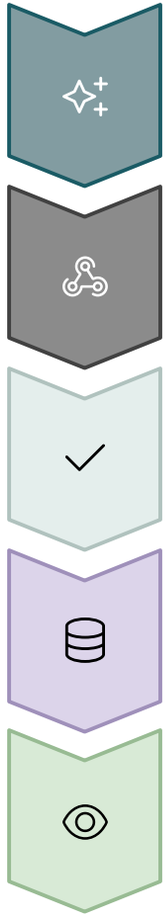
Un Job en Spark es una tarea que se ejecuta como resultado de una acción que ha sido invocada en un RDD o DataFrame. Cuando se llama a una acción, Spark crea un DAG (Directed Acyclic Graph) de transformaciones y envía ese DAG al clúster para su ejecución.

```python
from pyspark.sql import SparkSession

# Crear sesión de Spark
spark = SparkSession.builder \
    .appName("JobSparkEjemplo") \
    .master("local[*]") \
    .getOrCreate()

# Crear un RDD de ejemplo
datos = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
rdd = spark.sparkContext.parallelize(datos)

# Aplicar transformaciones (Lazy Evaluation - No se ejecutan aún)
rdd_transformado = rdd.map(lambda x: x * 2).filter(lambda x: x > 10)

# Acción que inicia un Job
resultado = rdd_transformado.collect()

# Mostrar resultado
print("Resultado del Job:", resultado)
```

Código completo:
https://colab.research.google.com/drive/1U6Qav9YUpJbwYDfdAqzZ2c9McIm2HllU?usp=sharing

ACTIVIDAD PRÁCTICA GUIADA

OBJETIVO: Aplicar los conceptos aprendidos sobre información obtenida de una API pública



Paso 1

Crear una sesión de Spark.

Paso 2

Obtener datos desde la API JsonPlaceholder.

Paso 3

Verificar si la respuesta es exitosa.

Paso 4

Crear un RDD desde los datos obtenidos desde la API.

Paso 5

Mostrar algunos registros del RDD para validar la carga.

El código con los pasos es el siguiente:

```python
import requests
from pyspark.sql import SparkSession

# Crear una sesión de Spark
spark = SparkSession.builder \
    .appName("ActividadAPITransformacionesSpark") \
    .master("local[*]") \
    .getOrCreate()

# URL de la API pública
url_api = "https://jsonplaceholder.typicode.com/posts"

# Obtener datos desde la API
respuesta = requests.get(url_api)

# Verificar si la respuesta es exitosa
if respuesta.status_code == 200:
    datos_json = respuesta.json()
else:
    raise Exception(f"Error al obtener datos de la API. Código: {respuesta.status_code}")

# Crear un RDD desde los datos obtenidos de la API
rdd_api = spark.sparkContext.parallelize(datos_json)

# Mostrar algunos registros del RDD para validar la carga
print("Primeros 5 registros del RDD desde API:")
for registro in rdd_api.take(5):
    print(registro)
```

Código completo:

https://colab.research.google.com/drive/1DJDg0o63H1j3l-8uGwbEfCrOwaUdJC-S?usp=sharing