Unidad 2 - Tarea 3 - Procesamiento de Datos con Apache Spark

**Nombre del estudiante**

Giovanny Alejandro Pardo

**Grupo:**

Big Data (202016911\_27)

**Tutora**

Sandra Milena Patino Avella

Universidad Nacional Abierta y a Distancia-UNAD

Escuela de ciencias básicas, tecnológicas e ingeniería

Ingeniería de sistemas

Palmira – octubre 04 del 2024

**Definir los conceptos de RDD**

Un RDD (Resilient Distributed Dataset) es la estructura fundamental de los datos en Apache Spark. Se puede afirmar que los RDD son colecciones de elementos que difieren a fallos y que se pueden distribuir entre diferentes nodos en un cluster y que trabajen en paralelo. Permiten almacenar y operar con grandes conjuntos de datos distribuidos, teniendo siempre una inmutabilidad y una alta disponibilidad.

**Principales características de los RDD**

* **Resiliente:** En caso de perderse una partición, esta se regenera automáticamente usando las transformaciones previas, lo que significa qe tiene la capacidad de recuperase de errores o fallos en el cluster. Lo que garantiza la integridad de los datos.
* **Distribuidos:** hace referencia a los RDD que estan divididos en particiones que estan repartidas a lo largo de los múltiples nodos del cluster, permitiendo el procesamiento paralelo de datos. Lo que significa que mantiene un procesamiento eficiente y rápido de grandes volúmenes de datos.
* **Dataset:** Es una colección de datos particionados a los que se puede acceder en paralelo.
* **Persistencia:** En Spark se puede optar por persistir (almacenar en caché) un RDD en memoria o en disco para un acceso más rápido en acciones posteriores, lo que es útil cuando se necesita acceder rápidamente a los mismos datos.

Imagen 01

Texto

Descripción generada automáticamente

Nota: En esta imagen se evidencia el uso de la persistencia.

**Propiedades de los RDDs**

* **Inmutabilidad**

Después de haberse creado un RDD este no se puede modificar, en su lugar, cualquier modificación sobre un RDD generar un nuevo RDD sin alterar el original.

Lo que aporta una tolerancia a fallos y consistencia, ya que se podrá reconstruir el estado anterior de los datos aplicando las transformaciones necesarias sobre el RDD original.

* **Particionamiento**

Los RDDs estan divididos en múltiples particiones que se distribuyen a lo largo de los nodos del clúster. Cada partición puede ser procesada de forma paralela, lo que permite un alto rendimiento en operaciones sobre grandes volúmenes de datos.

Se pueden definir los esquemas de particiones para optimizar la eficiencia del procesamiento, por ejemplo, utilizando **hash partitioning** o **range partitioning**.

**Tipos de Operaciones en RDDs:**

* **Transformaciones:**

Las transformaciones crean nuevos RDDs a partir de uno existente. Las transformaciones son perezosas (lazy), lo que significa que se ejecutan y registran solamente cuando se realiza una acción, que permitirá la optimización eficiente de las operaciones.

Diagrama

Descripción generada automáticamente

**map:** aplica una función a cada elemento del RDD y genera un nuevo RDD con los resultados.

Imagen 02

Texto

Descripción generada automáticamente

Nota: En esta imagen se evidencia el uso de **map**.

**filter:** filtra los elementos del RDD a partir de un criterio y crea un nuevo RDD con los elementos que cumplieron con el criterio del filtrado.

Imagen 03

Texto

Descripción generada automáticamente

Nota: En esta imagen se evidencia el uso de **filter**.

**flatMap:** Se puede afirmar que es similar a map, pero en flatMap cada elemento se puede mapear a múltiples valores que devuelve un nuevo RDD.

Imagen 04

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Nota: En esta imagen se evidencia del uso de **flatMap**.

**groupByKey:** Permite agrupar los datos a particiones de una clave, repartiendo los resultados (shuffle) entre los nodos:

Imagen 05

Una captura de pantalla de un celular

Descripción generada automáticamente

Nota: En esta imagen se evidencia el uso de **groupByKey**.

reduceByKey: Esta transformacion se usa en RDDs de pares clave-valor y permite agregar valores para cada clave.

Imagen 06

Texto

Descripción generada automáticamente

Nota: En esta imagen se evidencia el uso de **reduceByKey**.

* **Acciones:**

**collect:** Permite devolcer todos los elmentos del RDD como una matriz al programa principal lo que resulta útil para la recuperación de todos los resultados en el controlador.

Imagen 07

Un reloj digital en la pantalla

Descripción generada automáticamente con confianza media

Nota: En esta imagen se evidencia el uso de **collect**.

**count:** Nos devuelve el número de elementos en el RDD.

Imagen 10

Texto

Descripción generada automáticamente

Nota: En esta imagen se evidencia el uso de **count**.

**take:** Este devuele los primeros n elementos del RDD.

Imagen 11

Imagen que contiene Gráfico

Descripción generada automáticamente

Nota: En esta imagen se evidencia el uso de **take**.

**reduce:** Aplica una función que agrega elementos en el dataset.

Ejemplo.

>> rdd.reduce(lambda x, y: x + y) sumaria todos los elementos del RDD

**Referencias**

El mundo de los datos (11/ 04/ 2023) Dominando Apache Spark (III): Explorando RDD (Resilient Distributed Datasets) y su poder en el procesamiento de datos. https://elmundodelosdatos.com/dominando-apache-spark-iii-explorando-rdd-resilient-distributed-datasets-y-su-poder-en-el-procesamiento-de-datos/

Aito-medrano.github.io () Spark RDDs - Inteligencia Artificial y Big Data.

<https://aitor-medrano.github.io/iabd/spark/rdd.html>

Huawei (07/ 20/ 2022) Introducción a SPARK| Certificación HCIA-Big Data. <https://forum.huawei.com/enterprise/es/introducci%C3%B3n-a-spark-certificaci%C3%B3n-hcia-big-data/thread/667230497530593280-667212895836057600>