TRANSFORMACIONES:

1. map

Código:

Texto

Descripción generada automáticamente

La transformación map aplica una función a cada elemento del RDD y genera un nuevo RDD con los resultados. En este caso, multiplica cada número por 2.

Propuesta de Mejora: Como mejora, se podría añadir un paso para verificar el resultado con un pequeño subconjunto de datos (take) antes de aplicar la transformación completa en grandes datasets.

2. filter

Código:

Texto

Descripción generada automáticamente

La transformación filter selecciona elementos que cumplen con una condición, en este caso, números pares.

Propuesta de Mejora: Podría ser útil combinar filter con map si hay procesamiento adicional que deba aplicarse después de la filtración.

3. flatMap

Código:

Texto, Carta

Descripción generada automáticamente

flatMap aplica una función que puede devolver múltiples valores para cada entrada, expandiendo el RDD en más elementos. Aquí se divide cada frase en palabras.

Propuesta de Mejora: Se podría usar map con split si se desea obtener una lista de listas, pero flatMap es más adecuado cuando se necesita aplanar el resultado en un solo RDD.

4. union

Código:

Texto, Carta

Descripción generada automáticamente

union une dos RDDs en uno solo. Aquí combina RDD de números pares e impares.

Propuesta de Mejora: Distinct podría aplicarse al final si queremos eliminar duplicados después de la unión.

5. intersection

Texto, Carta

Descripción generada automáticamenteCódigo:

intersection devuelve los elementos comunes entre dos RDDs.

Propuesta de Mejora: Aplicar distinct antes puede reducir el tamaño de los RDDs y optimizar el proceso.

6. distinct

Texto, Carta

Descripción generada automáticamenteCódigo:

distinct elimina los elementos duplicados en el RDD.

Propuesta de Mejora: Distinct puede usarse en conjunto con particiones de datos específicas para distribuir la carga en un clúster grande.

7. groupByKey

Código:

Texto

Descripción generada automáticamente

groupByKey agrupa valores de un RDD por clave.

Propuesta de Mejora: reduceByKey es más eficiente que groupByKey para sumar valores o realizar operaciones de agregación, ya que reduce el tráfico de red al aplicar la función en cada partición primero.

8. reduceByKey

Código:

Texto

Descripción generada automáticamente

reduceByKey aplica una función de reducción a pares clave-valor.

Propuesta de Mejora: Este es un uso óptimo de reduceByKey para operaciones de suma o conteo.

9. sortByKey

Texto

Descripción generada automáticamenteCódigo:

sortByKey ordena el RDD por clave.

Propuesta de Mejora: Es adecuado para ordenar datos pequeños. En grandes RDDs, sortBy ofrece mayor flexibilidad para ordenar por valores en lugar de claves.

10. join

Tabla

Descripción generada automáticamenteCódigo:

join une dos RDDs mediante claves comunes.

Propuesta de Mejora: Para reducir el tamaño, se podría aplicar filter antes de join si se requiere unir solo subconjuntos de los datos.

11. cogroup

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteCódigo:

cogroup combina múltiples RDDs por clave, devolviendo valores de cada RDD en listas.

Propuesta de Mejora: Ideal para combinaciones complejas, aunque join puede ser preferible para datos más simples.

12. coalesce

Código:

Texto

Descripción generada automáticamente

coalesce reduce el número de particiones del RDD.

Propuesta de Mejora: Es útil para optimizar rendimiento en etapas finales del procesamiento especialmente al guardar en disco

**Acciones**

**1.** reduce

Texto

Descripción generada automáticamente

La acción reduce aplica una función a todos los elementos del RDD, combinándolos en un solo valor. Aquí se usa para sumar todos los números de 1 a 19.

Propuesta de Mejora**:** En caso de realizar operaciones complejas en un gran RDD, es recomendable realizar una reducción en etapas dentro de las particiones para mejorar la eficiencia.

Imagen que contiene Texto

Descripción generada automáticamente2. collect

collect trae todos los elementos del RDD al driver. Es útil para analizar datasets pequeños.

Propuesta de Mejora**:** collect puede consumir mucha memoria en el driver si el RDD es grande. Para grandes volúmenes de datos, se puede usar take o takeSample para obtener solo una muestra.

Texto

Descripción generada automáticamente3. Count

count devuelve el número de elementos en el RDD.

Propuesta de Mejora**:** No se necesita mejora, ya que count es eficiente en esta implementación. Es ideal para obtener un tamaño exacto.

4. First

Texto

Descripción generada automáticamente

first devuelve el primer elemento del RDD.

Propuesta de Mejora**:** Esta acción es eficiente, especialmente en conjuntos de datos ordenados. Para un subconjunto, take(n) podría ser más útil.

Texto, Carta

Descripción generada automáticamente5. Take

take devuelve los primeros n elementos del RDD.

Propuesta de Mejora**:** Take es eficiente y no necesita mejora, pero se puede especificar numSlices para controlar la paralelización en grandes datasets.

Texto, Carta

Descripción generada automáticamente6. TakeSample

takeSample devuelve una muestra aleatoria de elementos del RDD. Aquí se toma una muestra con reemplazo de tamaño 20.

Propuesta de Mejora**:** Para análisis estadísticos, es una buena práctica probar con muestras de diferentes tamaños o sin reemplazo (replace=False).

7. TakeOrdered

Texto

Descripción generada automáticamente

takeOrdered devuelve los primeros n elementos ordenados del RDD. Por defecto, se ordena en orden ascendente.

Propuesta de Mejora**:** takeOrdered es eficiente para obtener los elementos más pequeños o grandes. Para ordenar en otro criterio, se puede especificar una función de ordenación.

8. saveAsTextFile

Texto, Carta

Descripción generada automáticamente

saveAsTextFile guarda el contenido del RDD en archivos de texto. Se crea un archivo por partición.

Propuesta de Mejora: Es conveniente definir el número de particiones antes de guardar grandes datasets para evitar demasiados archivos pequeños. coalesce puede usarse para reducir el número de archivos.

Imagen que contiene Texto

Descripción generada automáticamente9. saveAsSequenceFile

saveAsSequenceFile guarda el RDD en formato de archivo de secuencia binario. Es ideal para almacenamiento eficiente y recuperación rápida.

Propuesta de Mejora: Asegurarse de que el formato sea compatible con las herramientas de análisis que se utilizarán posteriormente, ya que el formato binario puede ser más difícil de leer.

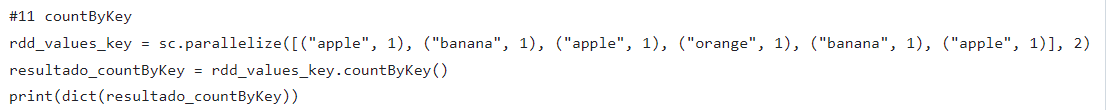
Texto

Descripción generada automáticamente con confianza baja10. saveAsObjectFile

saveAsObjectFile guarda el RDD en un archivo binario de objetos, que puede ser leído nuevamente en Spark.

Propuesta de Mejora: Este método es útil para almacenar RDDs de datos complejos. Es importante usar este formato cuando se necesiten leer los datos en Spark posteriormente sin convertirlos.

11. countByKey



countByKey cuenta el número de elementos por clave en un RDD de pares clave-valor.

Propuesta de Mejora: countByKey envía todos los datos al driver, lo que puede causar problemas de memoria en datasets grandes. Alternativamente, reduceByKey(lambda x, y: x + 1) es más escalable en estos casos.

Imagen que contiene Gráfico

Descripción generada automáticamente12. foreach

foreach aplica una función a cada elemento del RDD. En este caso, se usa map para transformar los elementos y luego collect para traer los resultados al driver.

Propuesta de Mejora: Este código usa map en lugar de foreach directamente en el RDD, lo cual es más útil si se necesitan los resultados en el driver. Si foreach se utiliza en acciones que producen efectos secundarios (como escribir en una base de datos), se debe tener cuidado con la idempotencia, ya que cada elemento se procesa en paralelo.