¿Cómo funciona la distribución de datos en un clúster de Apache Spark, y cuáles son los beneficios de este enfoque de procesamiento de datos distribuido?

R/

La distribución de datos en Spark se basa en particionar los RDDs en fragmentos. Cada partición almacenada en un nodo diferente del clúster. Esto permite que múltiples nodos pueden trabajar simultáneamente en las particiones de datos.

Entre los beneficios encontramos:

- rendimiento mejorado a través del paralelismo
- tolerancia a fallos
- capacidad de almacenamiento en memoria para acceso rápido a datos
- opción poderosa para el procesamiento de datos a gran escala

Explica el concepto de RDD (Resilient Distributed Dataset) en Apache Spark y su papel en la tolerancia a fallos.

R/

Básicamente es una colección inmutable y distribuida de datos particionados en un clúster.

Su papel central en la tolerancia a fallos consiste en su capacidad para recuperarse. Debido a su inmutabilidad, cuando un nodo falla, Spark puede reconstruir los RDDs perdidos a partir de los RDDs existentes y las transformaciones aplicadas, asegurando que los datos y las operaciones permanezcan disponibles y confiables.

Explique la diferencia entre acciones y transformaciones en Spark y dé ejemplos de cada una.

R/

Las transformaciones son operaciones perezosas que crean un nuevo RDD a partir de uno existente sin realizar cálculos inmediatos. Por ejemplo map() que aplica una función a cada elemento de un RDD.

Las acciones desencadenan la ejecución real y devuelven resultados o escriben datos. Ejemplo: count() para contar los elementos de un RDD.

Las transformaciones construyen un plan de ejecución, mientras que las acciones ejecutan ese plan para producir resultados.

Describe el concepto de evaluación perezosa (lazy evaluation) en Apache Spark. ¿Cómo mejora el rendimiento y la utilización de recursos?

R/

Las transformaciones en un RDD no se ejecutan de inmediato cuando se aplican, sino que se registran en un plan de ejecución. Spark solo ejecuta estas transformaciones cuando se llama a una acción que requiere resultados. Esto mejora el rendimiento y la utilización de recursos al evitar cálculos innecesarios. Spark puede optimizar el plan de ejecución y realizar operaciones en memoria, reduciendo la necesidad de lecturas y escrituras en disco.

¿Qué es el mecanismo de particionamiento en Apache Spark y cómo afecta el rendimiento de las operaciones?

R/

El mecanismo de particionamiento implica dividir un RDD en fragmentos más pequeños llamados particiones, y cada partición se procesa de forma independiente en nodos distribuidos.

El número y tamaño de las particiones pueden afectar el rendimiento. Un número adecuado de particiones mejora el paralelismo y evita la sobrecarga, mientras que un número excesivo puede aumentar la sobrecarga de gestión y reducir el rendimiento. Por lo tanto, un particionamiento bien ajustado es esencial para optimizar el rendimiento de las operaciones.

¿Qué son las transformaciones estrechas (narrow transformations) y las transformaciones anchas (wide transformations) en Spark?

R/

- Las transformaciones estrechas son aquellas en las que cada partición de entrada se mapea a una única partición de salida, sin necesidad de mezclar o redistribuir datos entre las particiones. Ejemplo: map o filter.
- Las transformaciones anchas requieren la mezcla o redistribución de datos entre múltiples particiones, lo que genera un shuffling de datos y suele ser una operación costosa en términos de rendimiento y recursos. Ejemplo: groupByKey o reduceByKey.