CASE TO BE SOLVED: PART TWO

Informe de Consultoría Técnica: Aplicación de Apache Spark en SoftaCorpSolutions

Preparado para: Dirección de SoftaCorpSolutions

Preparado por: Consultor Principal en Arquitectura de Big Data, RiseData

Fecha: 20/09/2025

Asunto: Recomendación Estratégica: Adopción de Apache Spark para el Liderazgo en Analítica Deportiva

1. ¿Cómo funciona la distribución de datos en un clúster de Apache Spark y cuáles son sus beneficios para SoftaCorpSolutions?

⇒ **Concepto y técnica:**

- Apache Spark no es una sola supercomputadora, sino un clúster o grupo de ordenadores (nodos) que trabajan en colaboración. Cuando carga una gran cantidad de datos como por ejemplo, las estadísticas de temporada en una liga deportiva, Spark no almacena en una única máquina; lo divide en fragmentos más pequeños (particiones), las cuales se distribuyen mediante los diferentes nodos trabajadores (worker nodes) del clúster.

- Cuando SoftaCorpSolutions realice un análisis (transformación o acción en Spark), el programa principal (driver) envía el código de la operación a cada nodo y cada uno de estos ejecuta la tarea en paralelo sobre la porción de datos que tiene almacenada de modo local. Luego, estos resultados individuales se recopilan y se consolidan para entregar una respuesta final.

⇒ **Aplicación y ejemplo en el ámbito deportivo:**

- Imaginemos que SoftaCorpSolutions quiere calcular la distancia media recorrida por cada jugador de fútbol en los últimos 100 partidos. Este conjunto de datos es masivo.

⇒ **Ruta de Ejemplo:**

- Spark toma los datos de seguimiento o tracking data de 100 partidos y los divide así por ejemplo:
  - Datos de los partidos 1-10 irían al Nodo A
  - Del 11-20 al Nodo B
  - Y así sucesivamente.

- El programa principal ordena: Calcular la distancia media por jugador:
  - El Nodo A calcula la media solo para sus 10 partidos
  - El Nodo B para los suyos
  - Y todos los demás nodos harían lo mismo simultáneamente.

Al final, el programa principal recoge los resultados parciales de cada nodo y los combina para obtener la media total por jugador.

⇒ **Por qué es bueno para SoftaCorpSolutions:**

- Velocidad y alto rendimiento:
  - Procesamiento en paralelo reduce mucho el tiempo de análisis.
  - Cálculos que tardarían horas se harían en minutos como un análisis casi en tiempo real durante un partido, para predecir la probabilidad de éxito de una jugada ofensiva basándose en datos del posicionamiento de jugadores.
  - Escalabilidad horizontal: SoftaCorpSolutions al adquirir datos de más ligas o sensores biométricos, no necesita comprar un servidor más potente, sólo añade más nodos al clústes y Spark distribuye el trabajo de forma automática (solución flexible y rentable para crecer).

- Eficiencia de datos (data locality):
  - Spark intenta ejecutar las tareas en los mismos nodos donde residen los datos.
  - Esto minimiza el movimiento de información por la red, que es uno de los mayores cuellos de botella en sistemas distribuidos (Shuffling).

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

⇒ **Concepto y Técnica:**

- El RDD (Resilient Distributed Dataset):
  - Estructura de datos fundamental de Spark.
  - Colección de datos inmutable (no se puede cambiar una vez creada), distribuida en particiones mediante el clúster.
  - Su característica robusta está en la resiliencia (capacidad para recuperarse de fallos).

- La clave de esta resiliencia es el "linaje" (lineage).
  - En vez de guardar los datos después de cada paso, Spark registra como una "receta" de transformaciones (filter, map, join) aplicadas a los datos originales para llegar al estado actual de un RDD.

⇒ **Aplicación y ejemplo en el ámbito deportivo:**

- SoftaCorpSolutions ejecuta un cálculo complejo por varias horas, como un modelo de ML para predecir el riesgo de lesiones de atletas basado en terabytes de datos biométricos de toda una temporada.

- Ruta de Ejemplo (fallo y recuperación):
  - El RDD con datos biométricos se distribuye en 100 nodos.
  - A la mitad del cálculo el Nodo 57 falla en el hardware y su partición de datos se pierde.
  - En un sistema tradicional esto detendría todo el trabajo pero con Spark el sistema no entra en pánico.
  - Utiliza el linaje del RDD para ver exactamente qué operaciones se aplicaron para crear esta partición perdida.
  - De forma automática Spark re-ejecuta esa secuencia en otro nodo disponible para reconstruir sólo ese fragmento perdido y así se continúa sin interrupciones.

⇒ **Por qué es bueno para SoftaCorpSolutions:**

- Garantía de fiabilidad:
  - Se aseguran los análisis largos y complejos, sin perderse por fallos de hardware (inevitable en clústeres grandes).
  - Proporciona resultados consistentes y fiables, crucial para el BI.

- Eficiencia en la Recuperación:
  - Más eficiente que la réplica tradicional de datos (HDFS), la cual requiere mucho más espacio en disco para guardar copias.
  - Spark solo necesita la "receta" para reconstruir datos.

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

⇒ **Concepto y Técnica:**

- El particionamiento es el mecanismo de Spark para dividir los datos de un RDD o DataFrame en fragmentos lógicos (particiones) y así distribuirlos en el clúster.
- La forma en que se agrupan los datos es crítica para el rendimiento, porque define el grado de paralelismo y la cantidad de datos por moverse mediante la red durante operaciones complejas.
- Mayor impacto en las operaciones que requieren reorganización de datos o "shuffle" (groupByKey, join), un proceso costoso en donde se mueven datos entre nodos para agrupar todas las claves relacionadas en la misma partición.

⇒ **Aplicación y ejemplo en el ámbito deportivo:**

- SoftaCorpSolutions busca analizar el rendimiento de cada jugador (ID_jugador) en diferentes partidos.
  - Ruta de Ejemplo (buen vs. mal particionamiento):

- Particionamiento ineficiente (malo): Datos de un mismo jugador dispersos por todo el clúster y para calcular sus estadísticas totales debe realizar un shuffle

masivo, moviendo datos de ID_jugador = 'Navas' desde todos los nodos a un único nodo para agruparlos (extremadamente lento).

- Particionamiento estratégico (bueno): SoftaCorpSolutions particiona los datos explícitamente por ID_jugador, entonces todos los registros de 'Navas' están en la misma partición (mismo nodo). Al ejecutarse el cálculo no hay necesidad de shuffle, porque la operación es local y mejora mucho el rendimiento.

⇒ **Por qué es bueno para SoftaCorpSolutions:**

- Maximización del paralelismo: El número de particiones define cuántas tareas se hacen a la vez. Una buena estrategia asegura que se aprovechen todos los recursos del clúster.
- Minimización del tráfico de red: Un buen particionamiento reduce o elimina el costos del shuffle (principal cuello de botella en el rendimiento de Spark), lo cual propicia un análisis más rápido y eficiente en su infraestructura.

Extra: Desafíos y estrategias para la optimización del rendimiento en Spark con grandes volúmenes de datos.

⇒ **Concepto, uso y técnica:**

- Optimiza trabajos de Spark a gran escala como en el análisis deportivo (desafíos estratégicos más allá del particionamiento).

⇒ **Aplicación y ejemplo en el ámbito deportivo:**

- SoftaCorpSolutions enfrenta varios desafíos al analizar petabytes de datos deportivos:

  - Desafío 1: Sesgo de datos (Data Skew)

    - Problema: Un equipo muy popular (Real Madrid) genera más datos que otros y crea una partición masiva (cuello de botella). Las tareas en este proceso tardan mucho y ralentizan el trabajo.

    - Estrategia - Salting: Añadir un sufijo aleatorio a la clave sesgada (ej. RealMadrid_1, RealMadrid_2, etc.), para distribuir los datos de la clave popular en múltiples particiones y equilibrar la carga de trabajo que permita un proceso paralelo.

  - Desafío 2: Recálculo Innecesario

    - Problema: El algoritmo de ML requiere acceder al mismo conjunto de datos de entrenamiento en cada iteración. Si no se optimiza Spark lo recalcula desde el origen cada vez (muy ineficiente).

    - Estrategia - Caching: Cache() o persist() para almacenar el DataFrame de entrenamiento en la memoria del clúster después del primer cálculo. De esta manera, en las siguientes iteraciones Spark accede directamente desde la memoria (acelera mucho el proceso).

- Desafío 3: Optimización Manual de Consultas

  - Problema: Escribir código RDD de bajo nivel para optimizar cada consulta (complejo y propenso a errores).

  - Estrategia - usar DataFrames/Datasets y el optimizador Catalyst: En vex de RDD´s se prefieren APIs de DataFrame o Dataset, las cuales utilizan un optimizador de consultas llamado Catalyst que analiza el plan de operaciones y lo reordena automáticamente, encontrando la ruta de ejecución más eficiente (por ejemplo: aplicar un filtro antes de una unión costosa), lo cual mejora el rendimiento sin intervenir manualmente por parte del desarrollador.

⇒ **Por qué es bueno para SoftaCorpSolutions:**

- Estas estrategias avanzadas garantizan que los análisis deportivos de SoftaCorpSolutions sean eficientes, escalables y robustos.
- Permite obtener respuestas y en un tiempo razonable, potenciando Spark como una solución rentable y con una ventaja competitiva decisiva en el mercado.