MODULO 07 - EJERCICIO 2-A

ALEXIS YURI M.

Lee un caso de uso (por ejemplo: detección de fraude en tiempo real, predicción de demanda, análisis de logs) y esquematiza cómo intervendrán los componentes principales de Spark: Driver, Executors, Cluster Manager, RDDs, DataFrames, etc.

1. Selección del caso de análisis.

El caso de estudio elegido es la predicción de demanda, un problema común en la logística y el e-commerce. Por ejemplo, una empresa de venta minorista quiere predecir cuántos productos necesitará en sus almacenes para la próxima semana.

2. Identificación de los componentes del ecosistema Spark.

Los siguientes componentes de Spark serían utilizados:

- Spark SQL: Se usaría para cargar y limpiar los datos de ventas históricos, que a menudo se encuentran en formatos estructurados como CSV. Estos datos se transforman en DataFrames, una estructura de datos más optimizada que permite realizar consultas SQL.

- MLlib: Es el módulo clave para la predicción. Se usarían algoritmos de regresión (como la regresión lineal o los árboles de decisión) o de series de tiempo para entrenar un modelo que aprenda a predecir la demanda futura basándose en datos pasados.

- Spark Core: Este es el motor subyacente que maneja las operaciones de procesamiento. El Driver de Spark coordinaría las tareas, y los Executors las ejecutarían en los nodos del clúster.

- DataFrames: Los datos de ventas se cargarían y manipularían como DataFrames para aprovechar la optimización del planificador de consultas de Spark.

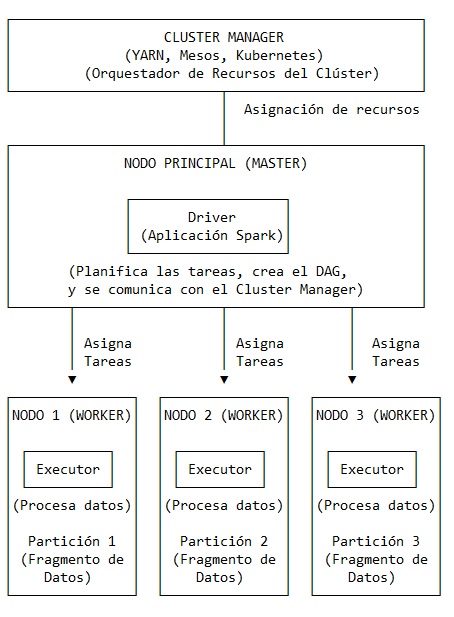
- API de Resilient Distributed Datasets (RDD): La API original de Spark. Los RDD son colecciones inmutables de objetos que se pueden procesar en paralelo en un clúster.

- API de DataFrame: Es una colección distribuida de datos organizada en columnas con nombre, similar a una tabla en una base de datos relacional. Los DataFrames se basan en la API de RDD, pero permiten optimizaciones de alto nivel, como el planificador de consultas Catalyst Optimizer de Spark, lo que mejora significativamente el rendimiento.

- API de Dataset: Esta API es una extensión de la de DataFrame que añade las ventajas de la seguridad de tipos, lo que la hace ideal para el desarrollo de aplicaciones que utilizan lenguajes de programación fuertemente tipados como Scala y Java.

3. Representación gráfica de la distribución en clúster.

El procesamiento se distribuiría de la siguiente manera:



- Cluster Manager: Este componente (como YARN o Kubernetes) se encargaría de la asignación de recursos y la gestión de los nodos en el clúster.

- Driver: Reside en el nodo principal y es el responsable de la planificación de la aplicación. Se divide la tarea de entrenamiento del modelo en etapas y se las asigna a los ejecutores.

- Executors: Estos procesos se ejecutan en los nodos de trabajo del clúster. Cada Executor procesa una parte de los datos (una o varias particiones) de forma paralela.

4. Justificación de por qué usar Spark y no otra herramienta.

Spark es significativamente más eficiente que Hadoop MapReduce por su capacidad de procesar datos en memoria. MapReduce está diseñado para escribir los resultados de cada etapa en disco (HDFS), lo que genera una latencia considerable. Spark, en cambio, mantiene los datos en la RAM, lo que lo hace hasta 100 veces más rápido para tareas iterativas. Además, Spark utiliza un Grafo Acíclico Dirigido (DAG) para optimizar el plan de ejecución, minimizando las operaciones de lectura y escritura. MapReduce tiene un modelo rígido de dos fases que no ofrece esta flexibilidad.

Mientras que Flink está optimizado para el procesamiento de streaming evento a evento, Spark es más versátil. La principal ventaja de Spark es que tiene una API unificada para el procesamiento por lotes, streaming, consultas SQL y machine learning. Esto permite a los desarrolladores usar una sola plataforma para múltiples casos de uso, lo que simplifica la arquitectura y el mantenimiento del sistema. Flink, por su parte, es muy potente en streaming, pero su ecosistema no es tan completo o maduro como el de Spark en otras áreas como el procesamiento por lotes o las bibliotecas de Machine Learning (MLlib).

En resumen, usar Spark se justifica por las siguientes 3 razones clave:

- Velocidad: Al procesar los datos en memoria, Spark reduce la latencia entre las iteraciones, lo que acelera el entrenamiento del modelo de predicción, en comparación con herramientas como Hadoop MapReduce, que escriben los datos intermedios en el disco duro.

- API Unificada: Spark ofrece una API consistente para el procesamiento por lotes y streaming, lo que simplifica el desarrollo y el mantenimiento del código. Si la empresa quisiera predecir la demanda en tiempo real, podría usar Spark Streaming sobre el mismo código base, lo que no sería posible con otras herramientas monolíticas.

- Biblioteca Integrada: Spark tiene su propia biblioteca de machine learning (MLlib) integrada, lo que evita la necesidad de exportar los datos a otras plataformas externas para el análisis.