# Resumen #3

Edgar André Araya Vargas 2020142856

## Apache Spark: A Unified Engine for Big Data Processing

Existen grandes desafios en el ambito informatico actual cuando los sistemas tradicionales de procesamiento de datos necesitan procesar grandes cantidades de estos datos. Es decir, problemas de rendimiento, escalabilidad y consistencia usualmente ocasionado a la desorbitante cantidad de datos que deben procesar. Apache Spark nace como una solución unificada para computación distribuida y procesamiento de grandes datos. Apache Spark nace como una solucion para estos desafios.

Hablamos de un sistema informático distribuido que se ejecuta en un grupo de equipos. Uno de los secretos clave mencionados de Apache Spark es su poder de procesamiento en memoria, ya que los datos se almacenan en la memoria y no en el disco duro, lo que puede acelerar significativamente el procesamiento. Apache Spark puede continuar funcionando incluso si una de las computadoras en el clúster falla, y soporta para múltiples lenguajes de programación, lo que permite a los desarrolladores usar el lenguaje de programación con el que están más familiarizados.

Es importante recalcar los principales beneficios de Spark.

- Es mas sencillo el desarollo de applicaciones al utilizar una API unificada.
- Es mas eficiente al combinar tareas de procesamiento al poder correr funciones sobre los mismos datos en memoria.
- Spark permite nuevas aplicaciones que no se creian posibles con los sistemas anticuados (queries interactivas en grafos y streaming machine learning).

Hablemos un poco de la arquitectura de Spark:

`lines = spark.textFile(“hdfs://...”)` `errors = lines.filter(` `s => s.startsWith(“ERROR”))` `println(“Totalerrors:“+errors.count())`

El ejemplo de código dado demuestra cómo crear un RDD a partir de archivos de registro, filtrarlo en función de las líneas que comienzan con "ERROR" y luego imprimir el número total de errores. Los RDDs se evalúan de manera perezosa, lo que permite a Spark encontrar un plan eficiente para la computación del usuario. Las transformaciones devuelven un nuevo objeto RDD, mientras que las acciones devuelven un resultado al programa. Los RDDs proporcionan soporte explícito para el intercambio de datos entre las computaciones, y los usuarios pueden persistir los RDDs seleccionados en memoria o en disco para su reutilización rápida. Este intercambio de datos es la principal diferencia entre Spark y modelos de computación anteriores como MapReduce. El intercambio de datos proporciona grandes mejoras de velocidad, a menudo hasta 100 veces, para consultas interactivas y algoritmos iterativos. También es la clave de la generalidad de Spark.

Se muestra un gran ejemplo con el "Batch gradient descent" un algoritmo iterativo simple que calcula una función de gradiente sobre los datos repetidamente como una suma paralela donde Spark facilita la carga de los datos en la memoria RAM una vez y la ejecución de múltiples sumas. Como resultado, se ejecuta más rápido que el tradicional MapReduce. En un trabajo de 100 GB, por ejemplo (ver Figura 4), MapReduce tarda 110 segundos por iteración porque carga los datos desde el disco en cada iteración, mientras que Spark solo tarda un segundo por iteración después de la primera carga.

Spark se integra con Hadoop Distributed File System (HDFS), Amazon S3, Apache Cassandra, entre otros sistemas de almacenamiento. Los usuarios pueden cargar datos desde estos sistemas de almacenamiento en RDDs y aplicar transformaciones y acciones a los datos. Spark también proporciona una API de alto nivel para trabajar con bases de datos relacionales utilizando JDBC. En resumen, Spark se integra con diferentes sistemas de almacenamiento para permitir a los usuarios trabajar con diferentes fuentes de datos de manera eficiente y fácil.

Spark Streaming proporciona procesamiento de datos en tiempo real y una API de alto nivel para el procesamiento de flujo de datos. El artículo también menciona la capacidad de Spark para representar gráficos a través de GraphX. GraphX es una biblioteca de Spark que proporciona una API para la manipulación de gráficos y el análisis de redes.

Otras bibliotecas de Spark importantes son la biblioteca de aprendizaje automático (MLlib) y la biblioteca de análisis de SQL (Spark SQL). La biblioteca MLlib proporciona algoritmos de aprendizaje automático para tareas como la clasificación, la regresión y la agrupación en clústeres. La biblioteca Spark SQL proporciona una interfaz de consulta para procesar datos estructurados, lo que permite a los desarrolladores interactuar con los datos mediante el lenguaje SQL.

Esta claro que Spark brilla en el campo de análisis de datos, se ha utilizado para realizar análisis de texto, procesamiento de gráficos, extracción de datos y análisis de datos científicos. Tambien para analizar redes sociales, analizar registros de llamadas, analizar datos meteorológicos, modelar recomendaciones de comercio electrónico, en el aprendizaje automático (donde es efectivo para entrenar modelos complejos). Modelos de aprendizaje profundo y sistemas de recomendación basados en el aprendizaje automático son utilizaciones de Spark.

Claro, al final del día Spark se utiliza para procesar y analizar grandes cantidades de datos en tiempo real y se utiliza en aplicaciones como análisis de transacciones financieras en tiempo real, análisis de datos de sensores y detección de anomalías en tiempo real. La cantidad de usos para Spark puede ser casi infinita cuando no existe un limite de archivos que queramos procesar.

En general, Spark se ha convertido en una herramienta popular para el procesamiento de big data debido a su capacidad para procesar datos a escala, velocidad y facilidad de uso.