Apuntes del 28/02/2023 Bases de Datos II

Moisés Solano Espinoza

Carnet: 2021144322

Spark

RDD (Resilent Distibuted Datasets)

Dataset. Raw data, no nos preocupamos de cómo venga la información, solo se lee, esta raw data queda como bytes.

- **No schema:** No tiene esquema de base de datos.
- **Read only:** los datos no se modifican en memoria, cuando se define un RDD se le pueden aplicar transformaciones y estas crean un nuevo RDD.
- **Acciones:** al RDD se le pueden aplicar acciones o transformaciones, de modo que al principio se va a tener el RDD1, luego el RDD2 ... ; no se modifican los datos de las anteriores
- No hay limite en la cantidad de transformaciones que se pueden hacer
- **Lazy evaluation:** los cambios no se van a ejecutar automáticamente. Si no se ejecuta acción, no se aplica transformación a todos los datos. Spark imagina cómo se verán los datos. Identifica los tipos de datos y hace mapping.
- Si se tiene un data set de 1tb por ejemplo, primero se leen los primeros 100 records como una partición y a estos se les aplican las transformaciones (RRD1, ...)
- **Speculative execution / task:** las particiones de información P1, P2 ... tienen información y cada servidor va a tener asignado 1 o más particiones. Cada transformación va a ser aplicada a cada partición y dependiendo del tiempo que tome ejecutar una tarea en el servidor esta tarea especulativa va a ser matada o seguir ejecutándose. Ayuda a la tolerancia de fallas.
- La recalendarización en spark sale muy cara y difícil.
- **Structured Data:** tienen nombre y tipo, un orden que tienen que seguir.
- **Semiestructured Data:** no se le fuerza el tipo.
- **Unestructured Data:** no tienen formato
- **Spark streaming:** manipulación de big data en tiempo real.
- **Provisioning:** tiempo que yo tardo en realizar toda la estructura necesaria para manejar datos.
- **Schema:** tipo de dato + nombre ***(en la clase solo vamos a trabajar con datos primitivod)***
- Spark maneja nativamente Scala.

Dataset

En palabras del profe: "es un RDD con el doble de hormonas".

- Strongly typed
- Object-Oriented
- Tiene un esquema relacional, con un orden
- **Serializable:** cuando tenemos datos se tienen clases. se debe buscar un lenguaje en común, una forma de hacerle una transformación de los datos en memoria en memoria a un archivo, normalmente XML o JSON para que pueda ser leída en otra computadora con otro lenguaje.

- Serializable
- Query-able
- In-memory Computation
- Fault tolerance

Para utilizar Elasticsearch con Spark

Instalación de Elasticsearch Hadoop

Para utilizar Elasticsearch con Spark se debe destruir primero el contexto creado por defecto para aplicar la configuración adaptado a Elasticsearch y arrancar de nuevo el contexto de Spark. Los dataset que se cargan están en crudo listos para ser utilizados.

- Todos los comandos en Spark se tienen que ejecutar en consola.
- Se puede configurar completamente, la configuración se carga y esta va a ser distribuida a todos los servidores que van a trabajar.
- Cuando se cargan los datos se genera un Dataframe en Spark, es una abstracción que contiene un RDD.
    - Cuando es big data primero se van a leer las primeras columnas.
    - Se puede configurar que la primera linea van a ser los encabezados de las columnas.
- Con el ***mapreduce*** se pueden empezar a hacer las transformaciones.
- Utilizaremos sql para manejar los datos.
    - Se le pueden mandar querys de sql para hacer consultas.
- Se puede empezar a transformar los datos crudos no estructurados a semiestructurados como JSON o PARQUET [es más poderoso, hace una compresión de más del 70% de la información textual, es un almacenamiento columnar en vez de rows]
    - Cuando se hace por rows las búsquedas van a ser muy lentas y se debe recorrer por completo hasta encontrar el registro buscado.
    - Por columnas se ahorra tener que llevar toda la memoria a disco.
    - En los archivos parquet se guarda el tipo.
    - Un ejemplo que se hizo en clase fue comprimir un archivo que pesaba 11megas a uno que pesa 3.7 con parquet.
- Evitar hacer collects con mucha información porque se puede llenar la memoria y matar la pc.

Para cargar a Elasticsearch

```
spark.sql("pets").savetoes([name])
```

Some Kibana code

```
GET _cat/indicesDELETE ...
GET pets/_search
```