Preguntas Teóricas

1- Suponga que usted esta a cargo de una ingesta de una tabla que contiene billones de registros utilizando Apache Spark. Que parámetros de spark tendría en cuenta a la hora de realizar dicha ingesta? Explique brevemente en que consta cada uno de ellos. En que formato de archivo escribiría los resultados? Por que?

Los parámetros que tendría en cuenta son:

- Número de particiones: Para controlar la cantidad de particiones en las que se divide el conjunto de datos, así como también seleccionar el campo de partición correcto.
- Tamaño del bloque: Determina el tamaño máximo de los bloques de datos que se leerán en paralelo.
- Estrategia de Escritura: Uso de escritura por lotes para mitigar la sobrecarga de escritura.
- Compresión: Elegir el algoritmo de compresión adecuado para encontrar el equilibrio entre rendimiento y espacio.

En cuanto al formato de archivo para escribir los resultados, para un conjunto de datos tan grande, un formato eficiente en términos de tamaño de archivo y velocidad de lectura/escritura seria Parquet, ya que es un formato de archivo columnar que es eficiente para consultas analíticas. Permite una compresión efectiva y una lectura selectiva de columnas, lo que puede reducir significativamente el tamaño de los archivos y mejorar el rendimiento de las consultas.

2- Teniendo en cuenta la tabla de la pregunta anterior, describa como realizaría el proceso de ingesta diaria de dicha tabla hacia un datalake. Que herramientas/lenguajes utilizaría? por qué?

Primero debemos separar el proceso en etapas, la etapa de extracción de los datos, la de transformación de los datos en caso de que sea necesario, y luego la ingesta/load de los datos en el datalake. El datalake puede ser una DB o un File System, para el caso de los datalakes AWS recomienda utilizar un S3.

Extraccion:

Para la etapa de extracción de los datos desde el origen dependiendo de la fuente de datos puede variar la tecnología que utilizaremos.

En el caso que sea una tabla de una base de datos podemos utilizar AWS DMS para la replicación de los datos a una tabla en un esquema landing, esto seria raw data. Es un método rápido y menos costoso para replicación entre dbs, además DMS nos permite manejar distintos tipos de DB origen y destino. También podemos utilizar Apache Spark con pySpark y sus librerías.

En el caso de un archivo podemos utilizar SQL y Python en conjunto con Apache Airflow, o también Apache Spark con pyspark para extraer los datos a un dataframe y luego ingestarlos en una DB o en un File System, como S3, en formato de archivo Parquet.

Cuando tenemos un servicio en la nube utilizaría Apache Spark con pyspark utilizando las librerías de Flask y Request. Ya que es de fácil y rápida programación, o también podríamos utilizar Python y SQL con Apache Airflow.

Transformación:

En esta etapa puede pasar que los datos se ingesten directamente en el datalake sin ninguna transformación, por ejemplo que se quiera la raw data, o puede pasar que los mismos necesiten algún tratamiento.

Esto se puede realizar utilizando las mismas herramientas mencionadas anteriormente, junto con librerías de procesamiento de datos como Pandas (Python) y Apache Spark SQL por ejemplo.

También se puede hacer la transformación de los datos utilizando SQL con Airflow, donde definiría un modelo de 3 capas:

- una landing donde está la raw data,
- una capa de staging donde está la data pre procesada
- y una capa final de data provision donde quedarían las tablas transformadas como por ejemplo facts, Dims, obts, Snaps, trabajando bajo un modelo estrella.

Ingesta:

Una vez que los datos están listos, se cargan en el Data Lake. El formato de archivo elegido para el almacenamiento puede ser Parquet u otros formatos compatibles con el Data Lake dependiendo de la cantidad de registros que tenga el modelo, en este caso sería parket porque son millones de registros, o también se puede ingestar en una DB.

Para la ingesta, se pueden usar herramientas como Apache Hadoop, Amazon S3 y también un RDS o Redshift, este último esta más preparado para datawarehouse. En esta etapa utilizaria también Herramientas como Apache Spark y Python, o SQL con Airflow.

Bonus Track!

1- Describa brevemente implementaciones que haya hecho con Apache Spark o problemas con los que se haya topado y como logro solucionarlos.

En cuanto a implementaciones realizadas con apache Spark, solo implemente la lectura de datos desde una API, la transformación de estos datos en un DataFrame y luego la creación de un archivo CSV en AWS S3, luego ese archivo era tomado por un pipeline croneado en Airflow y con el comando de COPY en SQL se ingestaba en un Redshift.

2- Describa brevemente su nivel de experiencia con los distintos lenguajes de programación que conoce.

- C++, tengo 15 años de experiencia, desarrollando aplicaciones para reportes de bancos y manejo de datos.
- JAVA, tengo 5 años de experiencia, desarrollando aplicaciones web, tanto frontend como backend.
- PYTHON, tengo 4 años de experiencia, desarrollando aplicaciones para pipelines de datos y microservicios.
- SQL, tengo 20 años de experiencia, trabajando en diferentes bases de datos como Oracle, DB2, Redshift, PostgresSQL, SQLServer.
- PYSPARK, tengo 1 año de experiencia, lo utilice con AWS Glue

3- Si cuenta con experiencia desarrollando sobre alguna nuble publica, detallar brevemente que proveedores cloud ha utilizado y que servicios de ellos conoce.

En cuanto a la nube de AWS trabaje con las siguientes herramientas:

- CostExplorer
- CloudWatch
- S3
- Redshift
- RDS (PostgresSQL)
- Dynamo DB
- DMS
- Airflow
- Api Gateway
- Kinesis Data Stream
- Kinesis Data Firehose
- Cloud Formation
- IAM
- EC2
- Lambda
- Glue

En cuanto a la nube de Azure trabaje con las siguientes herramientas:

- Active Directory
- Key Vault
- Batch
- API Apps
- API Manager
- Data Factory
- BLOB Storage
- SQL Data Warehouse
- Azure Databricks

4- Comente brevemente su conocimiento o experiencia utilizando distintos tipos de bases NO SQL.

La única base de datos NO SQL que utilice es Dynamo DB de AWS. La ventaja que tiene de trabajar en la nube es que esta base de datos es auto escalable, por lo tanto podemos manejar cargas variables, también note que tiene tiempos de respuesta muy bajos, en el orden de los milisegundos.

Algo que me llamo la atención es la flexibilidad de los modelos, por ejemplo tengo una tabla con 4 columnas generadas y on de fly cuando ingesto una nueva tupla con una columna nueva, la misma se genera de manera automática. También puedo definirles una key y sortkey.

Por otro lado es una base muy segura ya que ofrece seguridad como cifrado de los datos y acceso a los datos con AWS Identity and Access Management.