Designing a Data Ingestion Pipeline

Design approach

Key components:

Data Extraction: We need to extract all the new logs from the data source where the source is coming from the website and the mobile platform.

Identify the new records: Based on the requirement, speed and efficiency, the data collected/extracted need to be label and identify in order to ingest data that are newest, therefore old can existing data record won’t be ingested.

Loading into redshift: With the label data it will need to ingest into the data warehouse in this scenario is Amazon Redshift.

For the step of how it work:

1. **Data extraction:**
   1. The logs need to be extracted from the source in this scenario is from the website and mobile platform from carsome.
   2. Tools to use can be asw glue or aws lamda to create a job and extract into a S3
2. **Identify the new records**
   1. Implementing Change Data capture to track changes
   2. This can be achieve by adding time stamp based, log based, trigger based or aggregate the data into a unique identifier
3. **Data Transformation and cleaning :**
   1. Before loading the data into redshift, we need to ensure the data that we are going to ingest are in a structure format
   2. Data need to clean and normalize, the tool as suggested by AWS are AWS glue ETL
4. **Loading data into Redshift:**
   1. With the structure data, data can be loaded to the redshift database using the COPY command
   2. As for real-time scenario, Amazon suggest using kinesis firehose to stream data into Redshift
   3. Based on the scenario given, batch loading into redshift will be more suitable as the CDC method is time-based therefore, data is aggregated. [[1]](#footnote-1)

Change Data Capture (CDC)

Change data capture (CDC) is a well-known technique to capture all mutating operations in a source database system and relay those operations to another system. CDC keeps all the intermediate changes, including the deletes. [[2]](#footnote-2) To my understanding the CDC is to log every changes made to the database therefor made tracking easier.

Use case for this Scenario:

* CDC is use to track and capture new user event logs
* Tool like AWS DMS are shown to be most popular option within the AWS ecosystem

**CDC Methods and Tools:**

* **Timestamp-based CDC**

A timestamp column is added to each record in the table. When a record is updated, the timestamp is modified. During CDC processing, records with timestamps greater than the last processing timestamp are considered as changes.

* **Log-Based CDC**

Log based is a way to keep track of changes in a database by looking at a special log that records every change, like new entries or updates. This helps quickly find and copy only the changed data to another place, making sure everything stays up-to-date without slowing down the main database.

Data Pipeline Architecture [[3]](#footnote-3)

Staging

Merge / Load

Source data

Redshift

S3

ASW Glue

Trigger extraction job

ASW Glue

Scalability and Performance

Scalability and performance are critical considerations in managing data and system operations effectively. Scalability involves expanding system capacity seamlessly to accommodate growing data volumes. Performance focuses on optimizing data handling processes, like using parallel processing for efficient data loading in Redshift. These strategies ensure systems can handle increasing demands while maintaining smooth and efficient operations.

**Scalability:**

* **Automatic Storage Expansion:**

We use a tool called Redshift Spectrum that allows us to access and analyze large amounts of data directly from our storage system, S3, without needing to move the data. This way, we can easily manage big datasets.

* **Sharing the Workload:**

Redshift uses a system called Massively Parallel Processing (MPP), which means it can break down tasks and share them across many computers to handle larger amounts of work quickly and efficiently. [[4]](#footnote-4)

* **Organizing Data Efficiently:**

We organize our data into different sections (called partitions) and create shortcuts (called indexes) to quickly find and process the data we need, making it easier to manage and access large volumes of information.

**Performance:**

* **Dividing Data Smartly:**

We split our data stored in S3 into smaller chunks based on dates or other important categories. This helps in quickly locating and processing the specific data we need without going through everything.

* **Loading Data Quickly:**

We use multiple commands at the same time to load data, which speeds up the process. Performing Parellel Prcessing use multiple concurrent COPY commands for efficient data loading.

Error Handling and Monitoring

Effective error handling involves rigorous data validation before loading into Redshift to maintain schema consistency. Using AWS CloudWatch, we monitor Glue jobs, Lambdas, and Redshift clusters in real-time, with alarms set to alert on failures or performance issues. Detailed logging across systems aids in quick troubleshooting and resolution of any issues that arise.

**Error Handling:**

* **Checking Data Before Loading:**

Before we put data into Redshift, we make sure it’s correct and matches the expected format. This helps prevent problems like mismatched data types or missing information.

**Monitoring:**

* **Keeping an Eye on Everything with AWS CloudWatch:**

We use AWS CloudWatch to keep track of our jobs and systems, like Glue jobs (which process data), Lambda functions (which run code), and Redshift (our database). It helps us spot any issues with errors or performance. [[5]](#footnote-5)

1. Best Practices for Building a Data Lake on AWS for Games https://docs.aws.amazon.com/whitepapers/latest/best-practices-building-data-lake-for-games/data-ingestion.html [↑](#footnote-ref-1)
2. Load data incrementally from transactional data lakes to data warehouses

   https://aws.amazon.com/blogs/big-data/load-data-incrementally-from-transactional-data-lakes-to-data-warehouses/ [↑](#footnote-ref-2)
3. Load data incrementally from transactional data lakes to data warehouses

   https://aws.amazon.com/blogs/big-data/load-data-incrementally-from-transactional-data-lakes-to-data-warehouses/ [↑](#footnote-ref-3)
4. Performance

   https://docs.aws.amazon.com/redshift/latest/dg/c\_challenges\_achieving\_high\_performance\_queries.html [↑](#footnote-ref-4)
5. Automating AWS Glue with CloudWatch Events

   https://docs.aws.amazon.com/glue/latest/dg/automating-awsglue-with-cloudwatch-events.html [↑](#footnote-ref-5)