Federated Data Ingestion & Processing on GCP – Solution Design

Summary

At Latin America's largest airline there exists the challenge of enabling analytics cross regions while simultaneously combating issues such as data silos, inconsistent schemas and latency issues. To solve this problem I propose an architecture based on native GCP services mostly that lets a logic data federation, keeping the regional control without losing the centered analysis capacity.

General Architecture

We focus on storing the data in **BigQuery**, a native GCP service that offers a transversal connection between different GCP services, enabling unified analysis across multiple regions and data sources without the need to move or replicate data. Its federated query capabilities allow seamless access to structured and semi-structured data in Cloud Storage, CloudSQL

Data Ingestion

Here we take the datastream from different sources and upload them to **BigQuery**. We have 2 options here, batch ingestion and real-time ingestion depending on the data type.

1. **Batch ingestion**: For data that is not required to save instantly or KPIs not monitored at real time. For example finance accounting invoices, they can be ingested periodically during the day.

   In this case we upload the data in different sources than BigQuery as Cloud Storage and Cloud SQL and ingest them through Dataflow templates to BigQuery.

2. **Real-Time ingestion**: For data that is important to monitor at real time KPIs related to ticket events, flights and purchases for example.

   In this case we capture the data through **Pub/Sub** (for example API REST data published with http) and then process them with **Cloud Functions** or **Dataflow** to upload them into bigquery in their raw format.

Schema Normalization

It is important to have defined and standardized schemas for the tables and validate them before uploading. For this purpose we use

1. **Cloud Functions/Composer** to validate the schema before uploading.
2. Schema Registry in **Git**, to have versioned schema
3. **Dataform** to define dimensions, metrics and SQL transformations to raw data

Transformation and Processing

Then for processing and transforming the data into KPIs, views or more valuable tables for the business we can use **Dataflow** for batch or streaming depending on the importance of the KPI (NPS, flights, etc.). We can do it also with **BigQuery SQL** and **Dataform**, and if there are other clouds used like AWS or a stronger ecosystem for DevOps with Git I recommend using **dbt.** If there are complex pipelines required, maybe we would use **Composer.**

Federated Query & Storage

In order to create a federated query we use regionalized datasets in BigQuery with async replication to centralize the data (for that we can use Data Transfer Service, Scheduled Queries or Dataform/dbt for selective replication).

On the other hand, if we decide not to ingest data from Cloud Storage or Cloud SQL we can use BigQueryExternal tables (this reduces costs and also offers a fast implementation but doesn't improve latency). And **BigQuery Omni only** if there are **other clouds** used as AWS services.

Catalog & Governance

In order to keep order between the different tables and views we proceed building the governance of the data with help of **Dataplex** that offers an automatic data inventory in GCS and BQ, quality rules and access policies and sensible data labels (considering PII). To manage access to data, labels, categorize data and many others actions related to governance we use **Data Catalog** that offers centralized metadata and semantic search**.**

On the other hand we also do Schema Harmonization to standardize fields, indicators and have the same format among different regions, then it is simpler to make queries not worrying about the differences between regions.

Cost & Performance Optimization

With the goal of reducing costs we use **Caching** for dashboards in Looker Studio, and in BigQuery we use Partitioning, Clustering for large tables. Also in BQ we use Materialized Views for frequently accesed metrics/tables. Finally we also use BQ Reservations in case of high volume in a region.

Fallback & Trade-offs

When there is a failover of real-time, we can save the data temporarily in a batch in Cloud Storage.

Also supporting caching in BigQuery through temporary tables or materialized views. Adding partial consolidation if cross-region performance is low.