Brief description GCP services used

1. **Analytics Hub** is a service for publishing and subscribing to data products from various data domains. This data exchange capability enables access to information from different projects and business areas, facilitating the creation of new data products that enrich the data team's information and lead to improved insights.

   Trade-offs:

   a. The absence of a centralized operational model could lead to duplicated efforts if data stewards do not synchronize their backlog of solutions across different teams.
   b. Can you present high-latency when execute query trough cross-region
2. **Dataflow** is a good tool to create ELT or ETL pipelines. The extract step can utilize JDBC protocols or Python libraries to retrieve information from traditional data sources such as RDBMS or NoSQL databases.

   Trade-offs:

   a. Dataflow uses the Apache Beam framework, which relies on community support for continuous issue fixes. Additionally, a specific version will reach its end of support within one year. This creates several problems for the data team because it is mandatory to define a strategy to upgrade the version within the year.
   b. GCP charges a fee for support related to Apache Beam solutions.
   c. Cold starts take approximately 4 minutes, while compute SKUs are billed continuously.
   d. To maximize the benefits of this tool, solutions should be designed for parallel processing to leverage horizontal scaling. Therefore, if the extraction process cannot be parallelized, this tool is not suitable for retrieving the information efficiently.
3. **Datastream** service is an excellent option for retrieving information in near real-time (NRT) from traditional databases such as Oracle, MySQL, PostgreSQL, or SQL Server. For Oracle sources, this tool utilizes archive logs to capture different transaction states, which eliminates the impact on database queries. Furthermore, the service offers various features such as duplication control, automatic schema conversions, and other functionalities.

   Trade-offs:

   a. The backfill method retrieves all data from the source to compare it with the destination. While this is beneficial initially, if operational issues occur and the service goes down, a subsequent backfill will necessitate the service retrieving all information again for comparison to identify the data that needs to be loaded into the table.
   b. Additionally, you can only partition data up to 5 years prior for retrieval, and the clustered field uses the primary key, which might not be efficient when dealing with a high volume of daily events.

    c. Another point to consider is that if your table experiences a large number of events like row creations, updates, and deletions, you can incur significant costs. Therefore, it is crucial to identify a clear business case to warrant a positive return on investment (ROI).

4. **Cloud Run** is an excellent tool for provisioning a backend with various endpoints that can execute tasks in other services by utilizing their APIs. This service can connect to your Virtual Private Cloud (VPC) and leverage a load balancer to distribute traffic across a serverless solution transparently to the user.

   Trade-offs:

    a. Securing deployment demands the management of numerous supporting resources, such as load balancers, SSL certificates, domains, routes, firewall rules, IP addresses (ephemeral or static), container images, and more.
    b. As with Apache Beam, maintaining your Cloud Run application requires continuous attention, including regular checks for code vulnerabilities by monitoring library versions.

5. **Cloud Batch** enables the execution of tasks by leveraging the advantages of cloud computing. Its workload management capabilities allow for the creation of virtual machines to perform specific tasks embedded within a reusable framework, offering significant flexibility for various operations. For instance, it can utilize other APIs to execute asynchronous tasks.

   Trade-offs:

    a. As with Apache Beam, maintaining your Cloud Run application requires continuous attention, including regular checks for code vulnerabilities by monitoring library versions.

6. **Cloud Run Functions** is an excellent tool for event-driven architectures. It allows you to execute atomic tasks in serverless instances, such as moving a file to reorganize a bucket or loading a file into BigQuery without utilizing instance memory (e.g., by employing the BigQuery Write Storage API). Similar to Cloud Run, it also enables the execution of asynchronous tasks to send information to other services.

   Trade-offs:

    a. As with Apache Beam, maintaining your Cloud Run application requires continuous attention, including regular checks for code vulnerabilities by monitoring library versions.
    b. Billing is based on the number of requests. Therefore, it's crucial to consider the trade-off between continuously running an instance 24/7 versus the cost of invoking Cloud Run Functions instances based on demand.

1. How data will be ingested, cataloged, and processed across regions?

The approach to data ingestion, cataloging, and processing across regions involves several potential services, and the optimal choice depends on the specific business case.

Data Ingestion: The diagram illustrates various services that can be used to integrate different data sources. The selection of the appropriate tool will be determined by the specific requirements of the business case.

Data Cataloging: Cataloging data involves a two-step process facilitated by Dataplex. First, Dataplex automatically discovers and centralizes BigQuery metadata across the organization into a central hub, enabling data discovery and identification of data products. Second, to fully understand these data products, manual cataloging of business metadata is necessary. Subsequently, your ETL frameworks should be configured to interact with the Dataplex API. This integration will automatically populate technical metadata, providing a comprehensive view of the data.

Processing Across Regions: Processing data across regions can be costly. Effective solutions include leveraging Analytics Hub to create linked datasets across projects and regions, allowing you to enrich your data products without physically moving the data. Alternatively, consider utilizing multi-region locations in data services like BigQuery to optimize data locality and reduce cross-region processing costs.

2. How you'd manage schema consistency and versioning?

About schema consistency, one approach is to replicate the source schema directly in BigQuery. This helps ensure consistent data types and structures for ingested information. Another strategy is to initially load all columns as string data types in BigQuery. Subsequent transformation steps can then be used to curate the data and enforce a consistent final schema across regions and projects.

For schema versioning, infrastructure-as-code tools like Terraform, combined with CI/CD pipelines such as GitLab CI, can be utilized for continuous deployment of schema changes. This approach also allows for capturing the history of schema modifications for specific data products.

3. How you would approach cross-region query performance and cost optimization? Any trade-offs and fallback strategies (e.g., caching, materialized views)?

To cross-region query performance and cost optimization will primarily focus on minimizing data movement and optimizing query execution.

1. **Minimizing Data Movement:**

   Analytics Hub provide a logical, read-only view of data residing in other regions. This allows users in one region to query data in another through linked datasets without the need for physical data duplication. Significant geographical separation can still lead to latency.

   Fallback for Low Latency -> Data replication

   a. Trade-offs:
      i. Increased Costs: Duplicated storage and data transfer costs.
      ii. Operational Complexity: Requires building and maintaining sophisticated data pipelines.
      iii. Data Consistency Challenges: Ensuring data quality and consistency across replicas

   b. Consideration: Data replication warrants careful consideration only for vital datasets with need performance or availability requirements where the benefits clearly outweigh the increased costs and complexity.

2. **Optimizing Query Execution:**

   BigQuery's automatic caching is a great built-in performance booster. However, it's essential to be aware of the conditions that prevent cache usage. If your workload frequently involves data updates, non-deterministic functions, and others limitations you might not consistently benefit from caching and may need to consider other optimization strategies.

   Fallback for optimizing query execution

   Partitioning and Clustering: Employ partitioning and clustering to minimize full table scans. Additionally, BigQuery offers a flag to enforce partition usage in query predicates.

   a. Limitations: Data updates or non-deterministic functions loss the caching benefit for the project and user (depends of your Bq compute pricing model)

   Materialized Views are excellent for accelerating analytical workloads on large datasets by pre-aggregating and storing results. They can significantly reduce query costs and improve performance for common reporting or dashboarding scenarios. The automatic refresh capability helps maintain data freshness.

   b. Limitations: Doesn’t work with BQ Editions in standard version