Nigel Mun xData Data Engineer Technical Test

Data Architecture Design
Referencing the same case study as the 1st question:

Dataset A is formed by storing streaming data from all the video camera sensors from the edge. Each row in Dataset A is an event. There are around 10000 events per seconds.

You can assume Dataset B, is a static reference table that is unchanged.

Suppose your PM wanted you productionize this architecture, by considering the following requirements:

1. The PM wants to build a dashboard to visualize the results by joining Dataset A and B without any duplicates.
2. The join results of dataset A and B shall be available as soon as the events in Dataset A has been published.
3. Duplicate events may occur upstream due to retries, etc.

You are required to design an architecture to describe how you will be proposing the store, process and finally the considerations when accessing data. To score full points, you need to detail the questions you will ask the end user and considerations or assumption you will make in your design.

You can also assume the environment that you are using, i.e. Hadoop, Azure, GCP and etc.

You will also explain the tech stacks and storage tech stack that you are using why this tech stack shall be used. Your answer can be saved as `design.pdf` under the main repository. (30 marks).

Answer:

For this design, I will use AWS as the reference environment as I have experience with the AWS platform. The choice of AWS over other cloud services will be explained below.

High-Level Architecture Overview

Data Ingestion: Use *Amazon Kinesis Data Streams* to ingest streaming data from video camera sensors. Kinesis can handle the high throughput of up to 10,000 events per second from Dataset A, ensuring very scalable and reliable data ingestion especially for high volume and velocity of data.

Duplicate Event Handling: Implement *Amazon Kinesis Data Firehose* with a *Amazon Lambda* function with duplicate removal function and events based on unique identifiers within the payload. This step ensures that only unique events are stored, addressing the requirement to eliminate duplicates upstream.

Data Storage: Store the processed stream of data in Dataset A in *Amazon S3* in a partitioned format (e.g., by date/hour) to facilitate efficient access. Dataset B which is static data can also be stored in *Amazon S3*, in a different bucket. S3 serves as a durable, scalable, and cost-effective storage solution

ETL Process: Use *AWS Glue* for extract, transform, load operations. This involves potential schema management and metadata storage for Dataset A and Dataset B, apply transformations if necessary (e.g., schema normalisation), and prepare the data for efficient querying. Overall, this makes it easier for subsequent steps to understand and perform operations on the data structure.

Data Processing and Joining: Use *Amazon Athena* to perform SQL-based joins between Dataset A and Dataset B directly on data stored in S3. This service allows querying data in place without the need to load it into a separate analytics database, meeting the requirement for immediate availability of join results, suitable for new and updated data published in Dataset A.

Real-Time Dashboarding: Integrate *Amazon QuickSight* with the previously stated *Amazon Athena* for real-time dashboarding capabilities. QuickSight provides native integration with the service, enabling visualization of the join results with very low latency.

Overall Pipeline:

AWS Kinesis Data Stream → AWS Kinesis Data Firehose → AWS Lambda functions → S3 Storage → AWS Glue → AWS Athena → AWS QuickSight

Tech Stack Rationale:

- AWS offers a broader set of cloud services for compute, storage, database, analytics, machine learning, and application integration. This allows for a better integrated architecture that can address complex data processing requirements.

- All of AWS services are designed for high availability and scalability, which is perfect for this scenario. Amazon Kinesis for data streaming, Amazon S3 for data storage, and Amazon Athena for serverless querying scale automatically to handle massive volumes of data and high concurrency, ensuring consistent performance.

- AWS provides comprehensive security features that align with global compliance protocols. These include data encryption, identity and access management (IAM), network security, and logging and monitoring services, helping organizations meet strict data security and privacy standards.

Why choose AWS over GCP or Azure?

- **Ecosystem and Integration**: AWS's larger service ecosystem mean there seems to be an AWS service or feature for nearly every use case we require in this situation, hence no need to integrate third-party solutions.
- **Community and Support**: AWS benefits from a large user base, documentation, and QnA forums. This can make finding support and resources easier for AWS than GCP or Azure.
- **Track Record and Experience**: AWS has been the leader in the cloud space for much longer than Azure and GCP, hence is more reliable.

Assumptions in Design:

When designing the proposed architecture, several assumptions are made to guide the choice of technologies and configurations.

Scalability: The architecture assumes that handling scalability dynamically is crucial due to the continuous high velocity and volume of data from video camera sensors. It prioritizes services that can automatically scale to meet demand. Hence, S3 is chosen compared to other services like DynamoDB. S3 is more suitable for the data lake due to its ability to store vast amounts of data in various formats and its integration with analytical tools like Amazon Athena, while DynamoDB is a NoSQL database for OLTP.

Real-Time Analytics Requirement: The design assumes a strong need for real-time or near-real-time data processing to support operational decision-making and dashboard visualization, guiding the choice of streaming and processing services. Batch processing of data is not favourable based on the requirements of visualising new data inputs. Technically, *AWS IoT Core* can also be used as a solution, but since speed and scalability is the priority, *AWS Kinesis Data Streams* has a larger capability for real-time data streaming, higher throughput and lower latency.

Security and Compliance: The design incorporates robust security measures (e.g., encryption at rest and in transit, IAM policies) assuming that maintaining data security and compliance is very important.

Considerations and Questions for the End User

1. **Data Freshness and Latency Requirements**: Understanding the acceptable latency from event generation to visibility in the dashboard is crucial. This influences the choice of streaming and processing technologies.
   - *What is the maximum acceptable latency from the time an event occurs to its visibility on the dashboard?*
   - *How quick must the data need to be shown in the dashboard for optimal operational/business decision-making?*
2. **Data Volume and Growth**: Estimating the current volume and expected growth rate of Dataset A will help in sizing the infrastructure correctly and choosing the right storage formats (e.g., S3, DocumentDB, DynamoDB).
   - *What is the current volume of events generated per second, and what growth do you anticipate over the next 12-24 months? Linear or exponential growth?*
3. **Access Patterns and Query Performance**: Knowing the common queries and access patterns will guide the optimization of data partitioning and indexing strategies in S3 and Athena.
   - *What are the most common types of queries you anticipate running against the data?*
   - *Do you expect to perform more ad-hoc analyses, or will there be a set of standardized reports and dashboards?*
4. **Security and Compliance Requirements**: Clarify any data security, privacy, and compliance requirements to ensure proper data encryption, access control, and auditing mechanisms are in place.
   - *Are there specific data security standards that the architecture needs to comply with, such as making use of Identity Access Management?*
   - *What are the data encryption (at rest, in-transit, or both), retention, and access control requirements?*
5. **Budget and Cost Optimization**: Discuss budget constraints to balance cost and performance effectively. For instance, choosing between Athena for pay-per-query access patterns versus Redshift Spectrum for more frequent querying can impact costs.
   - *What is the budget allocated for cloud services related to this project?*
   - *How would you prioritize cost vs. performance if trade-offs are necessary?*