REFACTORING THE DATA PROJECTION STORE

by **Balamurali Srinivasan**

Streaming Architecture Patterns and the Journey to Kappa

Have you ever come across CQRS (Command-Query Responsibility Segregation)?

CQRS is a fascinating pattern built around the idea of having distinct models for reading (Read model) and writing (Write model) information. Behind this lies what is known as the Projection Store.

Today, let's dive into the world of streaming architecture patterns, particularly those used to populate the Projection Store in the realm of cash management.

Consumerisation Of Commercial Banking

Gone are the days at iGTB when we solely focused on displaying accounts and balances or completing payments. The current era is about the Consumerisation Of Commercial Banking. Our iGTB cash management products excel due to the rapid data access and smooth transaction and account navigation they offer. The initial version of our Projection Store drew inspiration from the renowned Lambda architecture, a concept introduced by James Warren and Nathan Marzi. But, as with any innovation, issues arose.

This journey from Lambda to Kappa was partly fuelled by an enlightening article by Jay Kreps titled "Questioning the Lambda Architecture".

Kappa vs. Lambda:

The Culinary Analogy - Lambda and Kappa represent distinctive data processing architectural styles. Let's digest this difference using a relatable foodie analogy:

Lambda Architecture: Think of Lambda as a restaurant where food is cooked in

large batches, with one team preparing and another serving the dishes. Though this ensures everyone is fed, it's inflexible. Making any changes can cause delays, waiting for the next batch.

Kappa Architecture: Here, there's a constant flow of food prep. Chefs cook in real-time, directly serving fresh dishes to diners. This allows instant adaptation to any menu changes or last-minute requests.

Both styles make sure everyone gets their meal, but their operational implications are starkly different.

The separate teams of chefs and servers follow a predetermined plan for cooking and serving food.

Both approaches ensure nobody goes away hungry, the way it is realised in Lambda and Kappa have a lot of operational implications.

"Moving to Kappa Architecture eliminates the need for separate batch and real-time processing layers"

Kappa

Agility and Customisation: With the ability to cook and serve in real-time, the chefs can adapt to changes and additions on the spot. They can accommodate different dietary preferences, adjust flavours, or even create personalised dishes based on specific guest requests. This flexibility enhances the overall dining experience and guest satisfaction.

Reduced Waste: Since the food is prepared in smaller portions as needed, there is less chance of excess food going to waste. The chefs can focus on optimising ingredient usage and avoiding unnecessary leftovers.

Fresher and Timely Service: By eliminating the need for pre-cooked batches, the Kappa approach ensures that the food is served immediately after it is cooked. This results in fresher and hotter meals being served to the guests, enhancing the overall quality of the dining experience.

Real-time Adjustments: The chefs can incorporate feedback from guests and make immediate adjustments to enhance the taste, and presentation, or even introduce new dishes. This enables continuous improvement and keeps the dining experience dynamic and engaging.

Lambda

Delayed Responsiveness: With the pre-cooked batches, any changes or additions to the menu require the chefs to wait for the next batch to be ready before accommodating them. This introduces delays in providing fresh and customised meals to the guests.

Resource Inefficiency: The pre-cooked food might not be consumed entirely, leading to potential wastage. Additionally, the chefs have to allocate resources and space for storing and managing multiple batches of food.

iGTB's Move to Kappa Architecture

Transitioning to the Kappa approach, we visualised data as an uninterrupted stream, doing away with separate processing layers. This data stream's introduction brought agility to our system, simplified our stack, and allowed for:

"This transition offers a tremendous advantage as data is ingested from various sources"

- Unified Data Architecture: We started treating all data, whether batch or real-time, as a continuous stream
- Streamlined Processing Logic: Adapting to Kappa led us to rewrite processing logic to suit streaming
- Direct Data Integration: Most batch layers were phased out, with data streamed directly into a unified pipeline
- Observability: With the advent of the cloud paradigm, having real-time event monitoring and observability became critical, leading to continuous performance metrics, ensuring peak efficiency and scalability

Where Are We Today?

The Projection Store concept involves storing normalised and de-normalised data in a single database, allowing users to access their transactions and navigate accounts effortlessly.

"Shift from the Lambda to Kappa architecture significantly improved resources efficiency"

By storing both forms of data, we achieve a balance between data integrity and query performance. Normalised data ensures consistency and reduces redundancy, while de-normalised data speeds up data retrieval and enables faster queries. With Projection Store, we strike a harmonious chord between data normalisation and de-normalisation, providing the best of both worlds. In the world of event processing, intellect has moved most of its significant architecture products to eMACH.ai