PROWESS SOFT

Event Streaming Platforms

Event Streaming Platforms & Case Study of Airport Baggage Operations

www.prowesssoft.com

+91-98498 77544 info@prowesssoft.com

PROWESS SOFT

Contents

Introduction 3

What is Event Streaming 4

Architecture 6

Use Case: Airport Baggage Handling Operations 8

Solution 9

Technology Architecture 9

Solution Architecture 11

Performance 13

Conclusion 14

The Whitepaper delves into the modern-day systems adoption of Event Streaming Platforms. It covers the evolution of Systems from the time of the monolithic application to the present-day Streaming Platform systems. Also included is a case-study of an Event Streaming System using a market-leading Event Streaming tool - TIBCO Streaming.

Introduction

As systems started to process a massive number of events, on the magnitude of millions per second, they had to innovate to the next paradigm of processing. Vertical and Horizontal scaling would only go so far and would still be limited to processing the data loads for which the systems were configured and optimized. In the current day, some systems cannot predict the events in-flow rates. For instance, with a viral video on social media, one cannot predict the in-flow of events into the processing system. There are emerging and prominent systems that are dealing with traffic that was not as predictable as they were hitherto. In other words, a use-case emerged for systems to process unbounded data.

Event Streaming systems work on processing the event data as it is received, while still in motion. The data is being analyzed and aggregated while it is still in motion. This strategy removed a huge bottleneck for systems - they started processing with zero I/O operations. Furthermore, the aggregated data is retained in memory and used for analytics. The system still inherently being event-based, the processes could still leverage all the capabilities of an event-based system, for instance, event pattern detection. Event Streaming platforms also provide a mechanism to store aggregated data in-memory, which enables analytics and the follow-up actionable insights derived on the fly.

The following sections explain the emergence of Event Streaming, the architectural components that make up the Streaming Platform and then goes on to describe a use-case implemented through TIBCO Streaming Platform.

What is event streaming?

Since the dismantling of the early monolithic application, computer systems have been evolving non-stop. Earlier systems relied on accumulating data on the disk, and on a configured schedule, the data would be pumped into the next system. As the next system processed this data, it stored the needed information in an operational data store (ODS) and created another set of data for the next system to consume. The lifecycle of the active data to-be-processed ended when it reached the last hop and the last system completed processing it. In a typical flow, comprising a decent number of hops, one could easily visualize that this was a staggered approach. The processing of data was interspersed with waiting periods until the data is sent to the next hop, and then another.

When messaging systems were introduced into the mix, the systems started to work closer in real-time. The staggered approach based on a scheduler was replaced with real-time systems that leveraged messaging platforms. As messages were received, in a typical process model, the data was parsed, enriched, transformed, and/or aggregated as needed. The result was stored in its ODS, and then messages were published out to the next system. The change in approach was huge considering the messages were independently processed, regardless of any schedules.

As messages (events) started flowing into systems in real-time, doors opened-up for newer possibilities. System owners understood that potential indicators of market insights were missed if systems are not able to correlate real-time events. The old way of processing individual messages lacked this correlation. It was not equipped to utilize the full potential of the knowledge that could be inferred through correlating a series of events. A new paradigm emerged, event-based architecture, that provided a way of correlating events that were seemingly unconnected, but when combined, gave valuable actionable insights to the systems. Event-based systems started solving interesting use-cases as they started aggregating data and looking for event-patterns - they could perform predictive analysis, they could build self-adjusting system based on events coming in. Event-based systems started pushing the throughput barriers. An event-based system along with an in-memory data grid formed a powerful combination as there were almost no I/O operations during processing.

Over time, the number of events shot up exponentially, and systems started searching for solutions for further optimization. A closer look into the systems revealed that there were a few things that could be optimized further - (1) the dependency on I/O operations when combining data from various sources, (2) the multiple hops the same message makes from one component to another was increasing the latency on processing.

There were specific use-cases that showed massive performance improvement when individual events were used and stored only as part of the aggregation. Also, more importantly, the data was aggregated in-flight and the aggregated data stayed in-memory for quick access. The individual events were treated as transient data and discarded after the aggregation. For instance, an event that carried data about an

flight schedule is only valid until the next update is received. An aggregated data with the latest update and the total number of updates can be more useful than all the individual events leading up to the last update. However, not every use-case fits the bill. For example, you could not discard all the debit and credit transactions and only keep the final update, in some scenarios each piece of data is important. The systems started adapting to the Streaming nature of events.

Architecture

An Event Streaming Architecture has three main components:

- An aggregator that gathers events from a variety of data sources.
- An analytics engine that allows for data correlation and brings the streams together.
- An interface that makes data available for consumption.

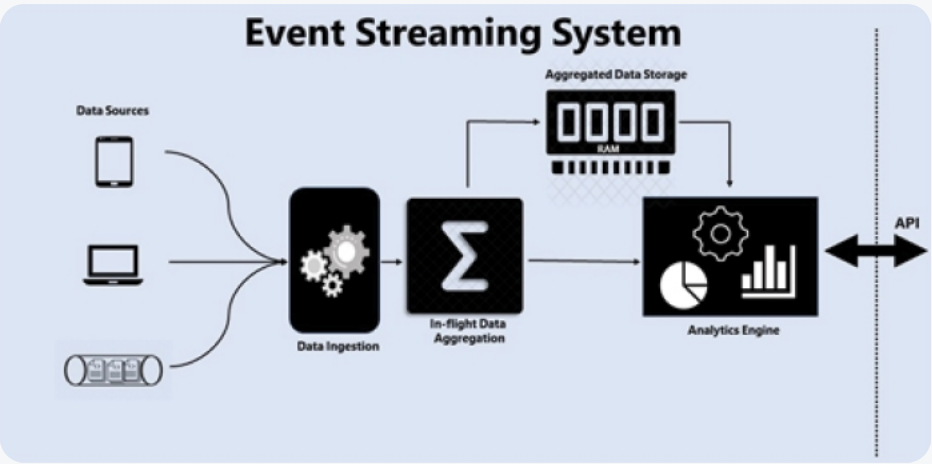

Figure 1: Event Streaming System

As shown in the picture above, an essential part of an Event Streaming platform is to provide a robust set of adapters that can connect with a variety of data sources. In today's omnichannel touchpoints for an application, any lack of connectivity is equivalent to the loss of data.

A central theme of the event streaming system is the processing of data while it is in motion. A streaming platform would have a rich set of features to create in-memory aggregation windows through with disparate events that can be correlated. For example, consider a GPS that is processing requests to route between a set of coordinates. If there is a sudden surge of requests from multiple sources requesting to find alternate routes from a certain point on the path, then an intelligent Streaming system can be programmed to capture such seemingly unconnected events within a certain time-window to raise appropriate alerts, and automatically process the next requests through alternate routes. If a traditional procedural process model were used, each of these requests to re-route would be processed for alternate routes, but without event correlation and aggregation, the system would not have tools to produce actionable insights from these events.

Another component you would see from the picture above is the analytics engine. The analytics engine is where the aggregated data can be used to create rules for insights. A feature-rich platform would expose these insights through an API for external consumers. The following sections discuss the application of a streaming use-case implemented through TIBCO Streaming.

Use Case: Airport Baggage Handling Operations

The customer is a large International airport serving over 200K passengers on a busy day. With over 400 flights operating on a regular day, the client has a lot of events to assimilate, aggregate, and action based on the information. Baggage Handling staff of the Airport Authority gather the information they need from various systems. They need information regarding the upcoming departures, arrivals, and the corresponding flights' baggage.

The baggage information for departures include the number of checked-in bags, the bags expected due to transfers, bags loaded, the bags that would be unloaded before the flight takes off on its next hop, etc. With regards to Arrivals, the baggage crew needs information about the expected flight arrivals ahead of time, along with the expected baggage numbers. The crew would use these numbers to allocate the Baggage collection belts to flights, based on individual belt capacities. Appropriate alerts were needed for flights that are about the depart, for gates that will soon be closed, for delayed transfer bags, for arrival bags that may overload belts due to individual belt capacities, etc.

Each event in the system can be potentially significant in terms of the unified view that the Baggage Handling crew needs. For example, a canceled flight event, can result in the corresponding Gate becoming un-assigned, and consequently, available for the next flight. On the other side, a missed bag message can result in an alert showing the flight is still awaiting an expected bag to be loaded.

Solutions

Aside from viewing it as an Airport Operations System, from a technical standpoint, this is a use-case for Event Streaming. It is an event-based system, but what makes it an event-streaming use-case is the unbounded nature of the events. There is a continuous stream of events that enter the system, and each one could make a potentially significant change. Also, for the unified view that the crew seeks, the view stays relevant only until the next event comes in.

Technology Architecture

There is an inherent need to correlate data from various systems in this use-case. The baggage data needs to be correlated with the corresponding flight. This use-case would need an integration tool, that can provide connectivity to the different systems exposed through different technology implementations. The flight information system needs to be connected through establishing a connection based on AMQP protocol implementation, while the Baggage System provides connectivity through a TCP connection for its data. Also, the use-case needs a tool that is feature-rich for working with streaming data. There is a need to host the processed data, from where it is easily accessible, with low latency, to the clients. A messaging system is needed for data transmission and communication between internal systems. A user interface is needed for display of the data and finally, a persistent data storage for backup.

TIBCO Streaming

TIBCO StreamBase: *TIBCO StreamBase provides various out-of-box operators to work with in-stream data. It also provides out-of-box adapters to connect with various types of endpoints that would be*

TIBCO LiveView: *TIBCO LiveView is a live business intelligence component of TIBCO Streaming. LiveView allows real-time streaming data to be hosted on the LiveView Server. Data that is hosted on the LiveView server is made accessible through StreamBase Connectors, APIs including REST API calls.*

TIBCO Enterprise Message Service (EMS)

TIBCO EMS is an implementation of industry-standard JMS Specification. TIBCO EMS is used for internal message transfer and communication within various components of the System. By

Angular and Apache HTTPD Server

Angular is used for building the web application that hosts the Departure and Arrival Views

MySQL Database

MySQL database is used as an Operational Data Store for persistent data

Solution Architecture

The component diagram below shows the various parts of the Baggage Handling System.

- *An aggregator that gathers events from a variety of data sources.*
- *An analytics engine that allows for data correlation and brings the streams together.*
- *An interface that makes data available for consumption.*

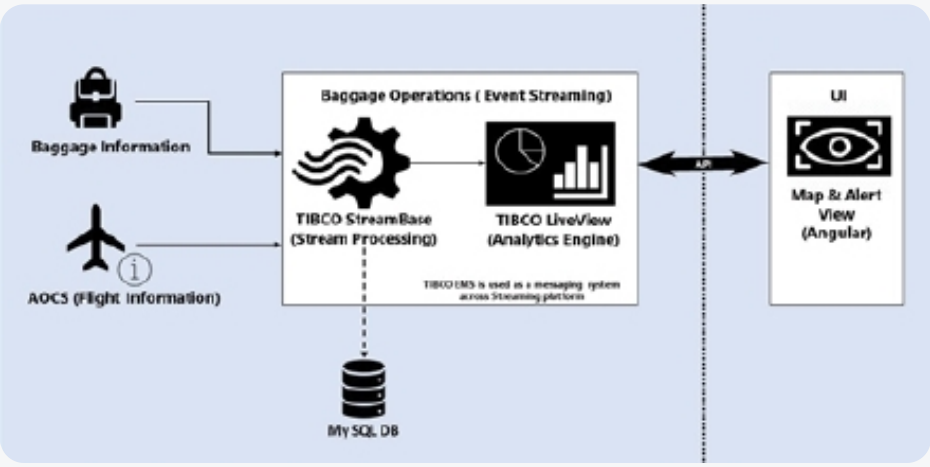

Figure 2: Baggage Handling System component diagram

AOCS sends data over AMQP connections implemented by the Apache Qpid server. TIBCO StreamBase to connects with AOCS using an Apache ActiveMQ JMS Connection Factory. Real-time Flight information is received into the system. Flight information such as new schedules, canceled flights, updated ETAs, etc., are received through these messages. An Industry standard, Aviation Info Data Exchange (AIDX) canonical message structure is adopted to comply with standards. These messages are parsed, and the in-stream messages are processed to check if this information updates any

The baggage system sends Bag-Messages through a TCP/IP connection after successful authentication and subscription. A custom adapter was built to make a connection to the Baggage System through a Bi-directional socket utility that TIBCO StreamBase provides. This component connects to the Baggage System and streams the Bag-Messages into the Baggage Handling System. The Bag-Messages contain data that can make a change to the baggage counts for a flight on the dashboard. It can indicate if the bag is just scanned at a certain terminal and is expected to be loaded, or it has reached the final milestone and if it can be considered as a loaded bag. More frequently, bag messages provide intermediate milestones. Each bag passes through a finite number of milestones before it can be considered as a loaded bag (for departures). Our system was built to capture various milestones within a bag's journey and using that provided certain critical statistics regarding the average time it is taking for a bag to move from one milestone to the next. This data is used by Baggage Operations staff to allocate internal staffing based on the alerts.

The usage of TIBCO LiveView to store live data helped in utilizing the REST API that it provides out-of-box.

TIBCO EMS was implemented for messaging between multiple applications. The flow was broken down based on the processing area. Separate applications were built for

- Receiving data for flight information and baggage information
- Processing departure flight stats, arrival flights stats
- Persisting data to the operational data store
- Hosting data for LiveView
- Health-check and cleanup processes

TIBCO EMS was implemented to serve as a backbone for carrying data across these applications. It helped in separating the time-critical real-time processing and the I/O intensive processing by bridging messages from one process to the other and letting them process in parallel.

Using Angular, an open-source web application framework, that has rich features with powerful template syntax, also helped with easy integration of REST APIs. The web application was deployed on Apache HTTP Server, that was just enough for the needed speed and look-and-feel for the UI.

Performance

The Airport serves more than 100 Airlines with over 65 million passengers passing through it every year. This translates to approximately 100 million bags going by an average of 1.5 bags per person. Considering each bag generates an average of 4 milestones on its way before it is loaded, there would be close to 400 million messages being processed for baggage messages.

Apart from the Baggage information, there are other messages in the system that include, current and upcoming flight schedules, change in the flight status, flight cancellations, updates to Estimated Departure and Arrival time. These events cause near real-time re-alignment of airport resources based on the changes.

With a water-tight design that avoids any memory leaks, and with the timely cleanup of messages from the system, thereby avoiding heap space overload, the system happily cruised without much of a blip.

Tests were conducted for peak volumes of 10 times greater than the expected load and the system performed without any loss of messages. More importantly, the back-end services were running on a 4-core processor, with an allocated heap of 8 GB for load-intensive engines and 4 GB for other engines.

Conclusion

The implementation of the Streaming platform using TIBCO StreamBase helped in multiple ways

- The high-frequency message processing was done while the messages were still in-flight, thereby avoiding an I/O and keeping the data on the dashboard current. This ensured a near real-time view for the operations crew
- The usage of TIBCO LiveView as a central component for hosting current data, not only helped with easy access to flight data for the UI, it also opened up a world of possibilities for any future custom views as it supports other programming language APIs
- The system was built with provision to change business rules with minimal impact. The segregation of the various components within the system was based on the end-systems, departures, arrivals, logging components, etc. Any changes in rules would be limited to the respective components and would not need re-testing or deployments of other components. The Baggage Handling crew were not scrambling for data across various systems and the manual counts for each flight's baggage aggregates. They started having access to the data as soon as it was received by external systems. This improved their overall operational efficiency.

www.prowesssoft.com

PROWESS SOFT

Event Streaming Platforms & Case Study of Airport Baggage Operations

Suite No.306 & No.1102 A & B,
Manjeera Trinity Corporate,
Kukatpally, Hyderabad
Telangana 500072 , India

+91-98498 77544 info@prowesssoft.com