Designing a Cloud-Native, Real-Time Data Hub and Reporting Dashboard

Supporting Cox Multi-Channel Contact Center Operations

A technical paper prepared for presentation at SCTE TechExpo24

Thomas Youngblood
Systems Administration Manager
Cox Communications, Inc.
thomas.youngblood@cox.com

Table of Contents

1. Introduction

In today's hyper-connected world, consumers expect instant communication through social media, fast shopping and shipping services, real-time news alerts, and prioritizing their time with family, friends, and leisure activities. The efficiency and performance of contact centers are critical to maintaining customer satisfaction in this environment. Large corporations often utilize internal staffing and outsource partners to manage the capacity of contact center interactions across multiple contact types (voice, chat, email, and social media). Organizations that have multi-vendor contact centers have the unique challenge of managing and analyzing metrics across multiple ACD (Automatic Call Distribution) platforms. When hold times matter and voice interactions build empathy and trust in business relationships, it is crucial for contact center administrators to have a holistic view of real-time metrics and make swift decisions to preserve those precious seconds.

A cloud-native application not only provides scalability and redundancy to ensure performance and uptime, but also supports customer-focused development by leveraging integrated platforms for interoperability and robust centralized logging and health metrics. Transitioning the platform from on premise Kubernetes to cloud-native enabled the platform to utilize a comprehensive suite of technologies with specialization in capabilities specific to real-time data metrics (including databases, data transformations, data streaming, and analytics and machine learning).

This white paper outlines how Cox harnessed readily available data, aggregated it, enhanced it, performed calculations, and displayed it to our contact center administrators in near real-time. The solution is enterprise-scalable, redundant, and diverse, with future development in mind, all while maintaining a cost-effective approach.

2. Real-Time Data Aggregation Challenges

Aggregating Contact Center voice data in real-time from multiple ACD platforms poses several technical and operational challenges, which require careful consideration and strategic solutions. In today's world, where communication is easily accessible to everyone, the contact center environment has evolved with the use of social media, artificial intelligence, and web/SMS chat solutions. However, voice communication remains a crucial element of customer service, where empathy and trust are established. To evolve, voice contact centers have expanded by leveraging multiple outsourcing partnerships. This approach helps reduce the average speed of answer, provide specialization and expertise, mitigate risks, offer geographical, linguistic, and cultural diversity, and maintain cost-effectiveness. Customer service is the top priority. The business leadership team manages contracts with outsourcing partners, diversifies call volume, and oversees customer service expectations. The technology team is then tasked with designing and implementing a robust communication and interoperability solution within the technical and infrastructure constraints set by the outsourcing partners. Key challenges to overcome include data consistency and integration, latency, scalability, and error handling. Addressing these challenges is crucial for creating a solution that provides real-time metrics aggregated from multiple platforms, enabling the business team to scale, enhance, and adjust customer experiences in real-world scenarios.

3. Mercury Platform

As Cox expanded its use of outsourced voice communications, the Mercury solution was developed to aggregate real-time and historical queuing and agent statistics from multiple ACD platforms. Cox developed this solution due to no existing holistic platform was available. Each organization and ACD provides updated metrics for call queues and agent statistics every few seconds. These metrics have transformations run in real-time to enable data enrichment and contact volume calculations. The Mercury solution streamlines operations, ensures data consistency, and provides customized reports and

dashboards. These features facilitate easier views for administrators to adjust, enhance, and manage customer service queuing effectively.

3.1. Overview

The Mercury solution is a comprehensive platform designed to meet goals through a robust architecture, enterprise-scalable infrastructure, and modern development languages. It's flexible, modular design promotes future expansion and accommodates custom feature requests. The key components of the Mercury solution include:

1. Back-End API Service

   The back-end API service is the cornerstone of the Mercury solution, responsible for receiving payloads containing real-time and interval historical data from the ACD platforms of Cox and Cox's Business Partners. It also retrieves data from a back-end database for consumption by the UI. The key features include:

   i. Standardized Payloads: A standardized data payload format ensures that data received from any ACD platform contains only specified elements relevant to the management of queuing and agent statistics. This standardization facilitates seamless integration and uniform data processing.
   ii. Data Security: Payloads are received by the API service exclusively from authorized entities, using an encryption key to ensure secure data transmission. This robust security protocol safeguards sensitive information and maintains data integrity. TLS v1.3 protocols are used as industry standard. Certificates and keys are stored in a secure certificate manager.
   iii. Real-Time Data Reception: The API service is capable of handling high volumes of data in real-time, ensuring a timely and accurate information flow into the system. This capability is crucial for maintaining up-to-the-minute insights and responsive decision-making.

2. Back-End Database

   The back-end database is designed to store, consolidate, and enhance the payloads received from the Back-End API service. Its primary functions are:

   i. Data Consolidation: Aggregates payloads based on call intents, providing a unified structure of call and agent data from multiple sources.
   ii. Metadata Augmentation: Enriches the raw payload with metadata, making it more accessible and understandable for human consumption.
   iii. Historical Data Management: Maintains both real-time and historical data, enabling comprehensive analysis over different time periods.

3. Front-End UI/UX

   The front-end UI/UX component offers and powerful, user-friendly interface for users to consume customized reports and administrators generate customized reports, add, remove, or edit metadata, or expand new call paths or destinations. Key features include:

   i. Customized Reporting Views: Administrators can create tailored report views to allow users to monitor specific metrics and performance indicators relevant to their needs.

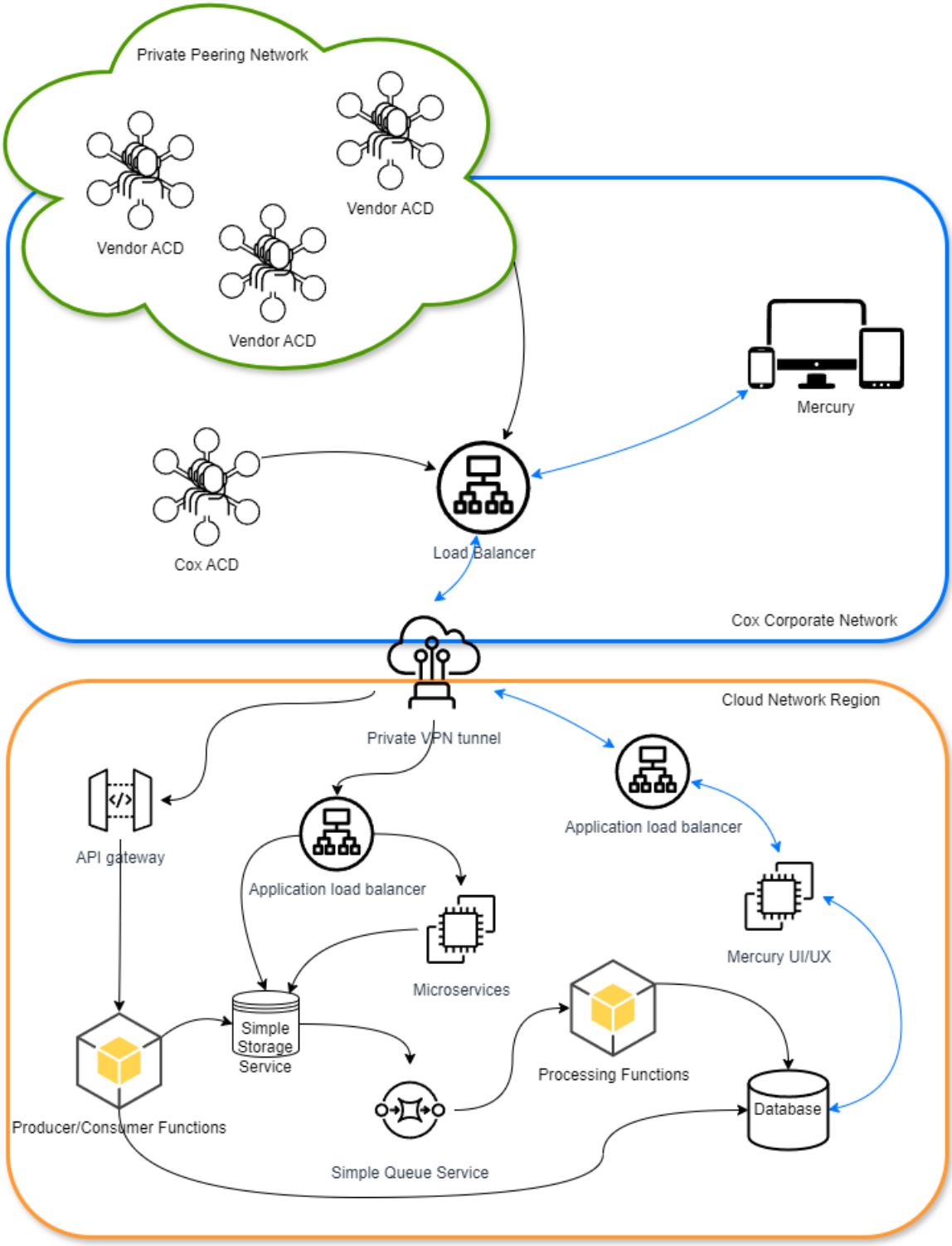
ii. Intuitive Design: The interface is designed for ease of use, allowing administrators to quickly generate templates and layouts based on business needs.
iii. Interactive Analysis Tools: Provides tools for in-depth analysis, including filtering, sorting, and visualization options to facilitate better optimization of call queue distribution, and early warning signs of major call volume drivers before disposition reports are received.

3.2. Business Requirements

The following business requirements were developed as a collaborative effort from Business and Technology teams to allow for future development opportunities.

1. User Interface Functions and Features
    a. Administrators create a Customized Layout view based on requirements. This ensures that specific reporting elements are visible to end users in a standardized format.
    b. Administrators create a template which executes standardized data requests assigned to custom layouts. This allows for data caching and prevents excessive number of queries against the back-end database.
    c. Real-time display of key metrics such as call volumes, average handle times, and agent states.
    d. Real-time display of incoming payloads to ensure accurate representation of consolidated data across multiple ACD platforms.
2. User Management:
    a. Integration with Enterprise security tools, to ensure access control, requiring approvals process.
    b. Role-based access control to manage permissions for different user levels.
    c. Group-based access controls to manage access to specific datasets within layouts.
    d. Administrative interface for user to group assignments.
3. Mobile Accessibility:
    a. User interface optimized for mobile devices, including smartphones and tablets.
    b. Consistent data representation between mobile views and desktop application.
4. Alerting and Alarming:
    a. Administrators create alerts based on specific metrics and thresholds.
    b. Alerts delivered via e-mail, SMS, and UX visualization.
5. Metadata Management Interface:
    a. User-friendly interface that allows Administrators to add, update, and remove metadata.
    b. Track changes to metadata by user and date.
6. Streamlined Onboarding process:
    a. Support for a wide range of ACD platforms.
    b. API documentation for custom integrations.
7. Scalability and Performance:
    a. Cloud-based architecture to support scalable data processing and storage.
    b. Load balancing to handle high volumes of real-time payloads.
    c. Continuous monitoring of system performance and resource utilization.
    d. Redundant infrastructure solutions to ensure high availability.
    e. Disaster recovery plan with failover mechanisms.

3.3. Architecture and Design



3.4. Cloud-Native

The decision to migrate the Mercury solution to a cloud-native application, featuring a modern web framework front-end, an enterprise class back-end database on a cloud-based computing service, and an API gateway for the API service, is driven by several key factors aligned with the outlined business requirements. These choices ensure scalability, performance, security, and ease of integration while leveraging modern technology stacks.

The rationale for adopting a cloud-native architecture includes scalability, high availability, and cost-effectiveness. The cloud-native approach provides auto-scaling capabilities to handle varying loads efficiently, allowing the platform to grow with the business and manage high volumes of real-time data. A cloud-based computing service offers a robust infrastructure with built-in redundancy and disaster recovery options, ensuring the Mercury solution remains available and resilient to failures. The pay-as-you-go pricing models enable cost optimization, allowing the company to pay only for the resources used, aligning with the goal of maintaining cost-effectiveness. Additionally, this technology stack provides specific capabilities for developing real-time reporting applications, enabling Mercury to utilize best-in-class solutions for real-time metrics use cases. The benefits include rapid deployment and updates, enhanced performance and reliability, reduced operational overhead, and a comprehensive technology stack specific to real-time data reporting.

For the front-end, a modern web framework is chosen due to its ability to provide a powerful framework for building dynamic and responsive user interfaces. This is crucial for delivering customizable dashboards and real-time data visualization features. The component-based architecture promotes reusability and maintainability of code, speeding up development and reducing bugs. The capabilities for building responsive and progressive web applications ensure a seamless experience across different devices, including mobile. This choice results in rich, interactive user interfaces, faster development cycles, and consistent performance across platforms.

The use of an API gateway is motivated by the need for security, scalability, and ease of management. The API gateway provides robust security features, including API keys, usage plans, and throttling, ensuring that only authorized entities can access the API service. It can handle thousands of concurrent API calls, scaling automatically to match demand. The seamless integration with other cloud-based compute services simplifies the management of APIs, monitoring performance, and deploying updates. This leads to enhanced security and control, automatic scaling and high availability, and simplified API management and monitoring.

The rationale for selecting an enterprise class back-end database centers on data integrity, security, and scalability. Enterprise class databases are known for their robust security features and reliability, essential for handling sensitive customer service data and ensuring data consistency. They can efficiently manage large volumes of data and complex transactions, supporting the scalability requirements. A comprehensive suite of tools and services facilitates seamless integration with other systems and applications. The benefits include high data integrity and security, efficient handling of complex queries and large datasets, and strong support and community.

By migrating to a cloud-native architecture with these components, the Mercury solution will achieve scalability, performance, security, and ease of integration, all while leveraging modern technology stacks to meet business requirements.

4. Conclusion

The decision to make the Mercury solution cloud-native architecture ensures the platform is scalable, secure, and capable of delivering high performance and a superior user experience. Leveraging these technologies, Mercury will be well-equipped to handle the complexities of modern contact center operations and support future growth and feature expansions.

Abbreviations

ACD	automatic call distribution
API	application programing interface
UI	user interface
UX	user experience
SCTE	Society of Cable Telecommunications Engineers