Video streaming & OTT testing Client Case Studies

Backend Server Expertise

- **Server side Applications: Segmenter, nDVR, DRM key management, Ad Server**
- **Backend application scaling**

- Live Segmenter
- Network DVR
- Dynamic Ad Insertion into the Live Segmenter workflow
- Implement CPIX Specification compliant DRM key delivery mechanism
- Implement Key delivery of encryption keys for Widevine, FairPlay and PlayReady DRM
- Enhancing Backend codebase for efficiency and scaling
- Containerization of backend services

Scope:

Implement support for CPIX Specification compliant DRM encryption key delivery mechanism

Challenge:

1. Implement an industry standard best practice mechanism for Key exchange between Packager, Key Management Service, Player and DRM License servers.
2. Backward compatibility with existing use cases.

Solution:

1. Use CPIX (Content Protection Information Exchange Format) standard to parse incoming key requests consisting of information like stream type, DRM standard, Key rotation etc.
2. Maintain the parsed information in an efficient custom designed class and data structures with efficient accessibility.
3. Generate encryption keys using the received information and subsequently recreate the CPIX document to be sent as a response from the Key management system.

Outcome:

A generic Key Management System (KMS) that can respond with keys to any packager using the CPIX Open standard.

Client:
Mediakind

Industry:
OTT Media Delivery

Technology Service:
OTT backend DRM Support

Scope:

Implement Key Rotation mechanism for delivery of encryption keys for Widevine, FairPlay and PlayReady DRM

Challenge:

1. Static DRM keys for media are vulnerable to being hacked in case of any security breach.
2. This compromises the entitlements based on individual subscriptions.
3. Need to find a way to safeguard media playback of live channels under all situations.

Solution:

1. Implementation of key rotation based on certain intervals.
2. This applies to multiple DRM types such as Widevine, FairPlay and PlayReady.
3. This significantly reduces the chances of compromising the media playback; irrespective of the client device.

Outcome:

1. Secure OTT playback with keys refreshing at certain intervals.
2. In case of security breach, the situation get rectified automatically without any requirement to manually change the keys in the database.

Client:
Mediakind

Industry:
OTT Media Delivery

Technology Service:
OTT Backend DRM Support

Scope:

Incorporate third party Redis database client library (Redis++) in OTT Backend codebase to leverage connection pooling mechanism for efficient scaling

Challenge:

1. An increasing number of Redis connections and corresponding latency were responsible for adding delays into OTT Backend APIs frequently used during playback.

Solution:

1. Utilize connection pool for Redis connections.
2. Utilize Redis Plus Plus; a popular open-source client with an inbuilt pool mechanism.
3. Implement a Singleton design approach for maintaining the Redis connections from OTT Backend.

Outcome:

1. Reduced connection attempts to Redis along with minimal latency for creating the connection is achieved.
2. Scale test results showed significant improvement with increased efficiency and reduced latency.

Client:
Mediakind

Industry:
OTT Media Delivery

Technology Service:
OTT Backend Infrastructure

Client:
Mediakind

Industry:
OTT Media Delivery

Technology Service:
OTT Backend Infrastructure

Scope:

Containerization of backend services to attain efficient scaling of services

Challenge:

1. Some backend OTT services (such as heartbeat/beacon service) need to support a higher volume of transactions as compared to others.
2. This service earlier was a part of the Content Controller appliance and hence it was not practical to increase the appliance instances where other services offered by the appliance would stay unutilized.
3. Need to maintain the same business logic and utilize the same codebase when converting the service to a container.

Solution:

1. Creation of separate heartbeat(beacon) microservice in the form of a container image.
2. Generation of beacon pod with master and side-car containers for automated service discovery, log management and KPI export.

Outcome:

1. Beacon pods successfully deployed in production-site Kubernetes clusters.
2. Auto-scaling and fault tolerance of beacon service achieved using tools provided by K8S.
3. Beacon Service latency reduced to less than 20ms after containerized deployment.