BUILDING SCALABLE AND RESILIENT OTT SERVICES AT SKY

T. C. Maule

Sky Plc, London, UK

ABSTRACT

Sky have tackled some of the most challenging scalability problems in the OTT space head-on. But its no secret that we've got it wrong in the past, disappointing our customers at the most critical of times, most recently in 2014. This paper will take you through some of the innovative scalability and resiliency patterns implemented across Sky's OTT estate since that time which contribute to delivering the reliable streaming services that our customers have come to expect today. From ensuring reliable delivery of VOD content, to handling spiky load profiles for Linear events, Sky ensures that the customer can always view their content, always failing in their favour if a dependent system becomes unavailable. Discover how we learnt from our previous mistakes, to build our most resilient and scalable OTT platform and clients to date.

INTRODUCTION

Sky is Europe's largest entertainment and communication business serving 22 million customers across 5 countries. Operating on over 80 different consumer devices, Sky's numerous OTT propositions offer tens of thousands of episodes of entertainment content and thousands of movies on demand, and 200 linear channels to millions of customers.

With some of the most popular Linear and VOD content including the Premier League and Game of Thrones available to an increasingly larger audience, Sky served 2700 petabytes of content via our CDN (Content Delivery Network) last year to peaks of over 750k concurrent users in the UK alone. Such enormous scale has led Sky to have to tackle some of the most challenging scalability problems in the OTT space head-on.

But in 2014 we got it wrong and disappointed a large number of customers just as the latest season of Game of Thrones was premiering (1). This paper will introduce some of the innovative scalability and resiliency architectural changes implemented across Sky's OTT estate since that time which contribute to delivering the reliable streaming services that our customers have come to expect today. From ensuring reliable delivery of VOD content, to handling spiky load profiles for Linear events, Sky ensures that the customer can always view their content, always failing in their favour if a dependent system becomes unavailable.

VOD VS LINEAR STREAMING

VOD (Video On Demand) streaming is a well-known concept, and there are many large players who are well established in this space (YouTube, Netflix, Amazon, etc.). Typically, VOD consumption is very predictable, with load rising and falling in a regular pattern based on the time of day.

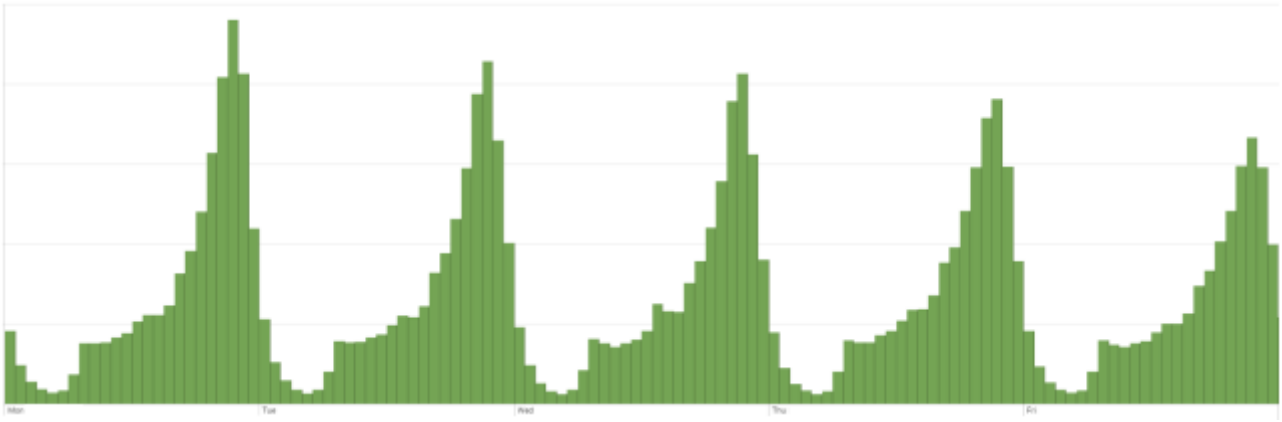

Figure 1 - Predictable VOD load pattern over a 5 day period

We can use this predictability to our advantage to bring about a number of benefits including:

I. **Auto-scaling algorithms** - If your peak hours see many multiple times the load compared to your quietest hours, operating with peak capacity 100% of the time would be very wasteful (and expensive). If your system can be scaled up and down automatically based on a reliably predicted load, then this extra unused capacity would not be needlessly wasted (or paid for).

II. **Identifying anomalies for reporting/alerting** - If you see the same load pattern over the course of each day and that pattern suddenly changes then this could indicate a fault, or worse - a malicious attack. Alerting and reporting on anomalies to your regular load pattern will allow your team to react far quicker to the pending threat.

But where our challenges *really* lie are in Linear (or Live) streaming. Linear streaming, or the streaming of TV channels in real-time, comes with the additional challenge of needing to handle your audience all initiating playout at roughly the same time. The more popular the event, the larger the challenge. With large-scale events such as Premier League football games and season premieres of some of the most popular Entertainment series including Game of Thrones and Westworld, it's not been uncommon for us to exceed 125 times our regular platform load in the few minutes either side of the start time of these large-scale events (see Figure 2).

Figure 2 - Linear Load Spike - Game of Thrones premiere 2016

HANDLING PEAKS IN LOAD

While our video content itself (the video manifest and video chunks) is highly cached and the accompanying surge in load on our CDN is appropriately handled, it is the surge in traffic to our OTT Platform APIs which is the more challenging problem. Traditionally, when a client application is started by a user there are a number of operations that must succeed in order for that user to be able to start watching content, including:

I. Content Discovery & Personalisation
II. User Authentication
III. Entitlement Validation
IV. Stream Initiation

Being able to guarantee all of these operations for every user, all within a few minutes of each other has been a unique challenge that has driven a number of design decisions and patterns at Sky over the past few years. It is these designs patterns that this paper will focus on.

I. CONTENT DISCOVERY & PERSONALISATION

Sky's OTT clients discover the content catalogue via an HTTP RESTful Content Catalogue API which delivers content metadata and imagery. Without this metadata and imagery, the client application would have nothing to display and therefore nothing for the user to browse and select to watch.

We have intentionally kept this fundamental service entirely non-personalised, allowing every piece of content metadata and every image to be heavily cached by CDN. This has two key benefits:

I. The CDN offload for content metadata and imagery is over 99%. This means less than 1 out of every 100 requests made by our client applications breaks through the cache to our origin servers, resulting in the scale of our origin servers remaining relatively low despite rapidly increasing customer numbers.

II. Because the CDN is handling 99+% of metadata and imagery requests, should our origin become unavailable, less than 1% of requests from the clients will fail since our CDN configuration ensures delivery of cached content, even if the cache is stale. That is, if our catalogue API origin is unavailable, our CDN will continue to serve a last-known-good until the origin has recovered and the cache can be gradually refreshed.

Our personalisation services, i.e. those services which in some way vary (or prioritise) the content presented to the user, are requested by our client applications separately from the Content Catalogue. Since these APIs are personalised, they do not lend themselves to being cached by the CDN in the same way as the Content Catalogue and hence the load to our origin is more significant.

To manage the load, our personalisation services inform the client of the client-side caching policy on every response which is strictly adhered to by the client. This ensures two key behaviours:

I. The client has a local cache to fall-back to should the personalisation service become unavailable.

II. The personalisation service is in control of managing its own expected future load, increasing the cache time to encourage clients to back-off in peak times, or reducing the cache time to encourage clients to refresh more frequently in quieter hours.

Additionally, in every case of personalisation, our client applications have circuit-breaker logic built in which ensures that the experience is not degraded should the request to the origin take too long or fail entirely. The client can fall-back to its local cache (if available), or at worst case simply provide an un-personalised view until regular service resumes.

By keeping our personalisation of the user's content discovery experience separate from the underlying content metadata and imagery, we are able to fall back to at *least* a non-personalised browsing experience if our origins were to fail entirely.

II. USER AUTHENTICATION

Traditional authentication involves a user entering credentials in a client application and submitting to some form of Identity System which would validate the credentials and issue a session token to the client. Typically, this session token would then be stored by the client application and submitted on every authenticated request to identify the calling user.

However, in a microservice architecture, this approach leads to a number of your backend services all having a dependency on your Identity System to validate the session token provided on every request and provide some form of user or account identifier that can then

be used to fulfil the request. This puts your Identity System in the heart of your architecture as a single point of failure (see Figure 3); lose your ability to validate a session token and you lose your ability to serve every single authenticated request in your system.

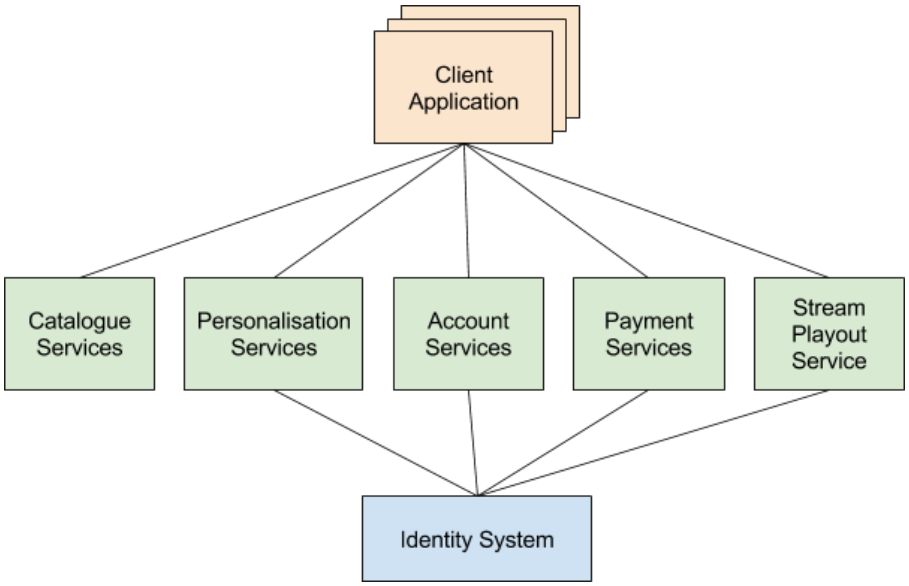

Figure 3 - Identity System As A Single Point of Failure

To solve this problem, we've developed our own authentication token 'wrapper'. This wraps our regular session tokens in our own custom wrapper, allowing us to securely carry an account identifier within the token itself (see Figure 4). Our token wrapper is secured through server-side encryption with rotating keys, and each wrapped token has a limited time to live.

Figure 4 - Authentication Token Wrapper

When a user now authenticates in our client application, a Token Wrapper Service handles the authentication request, obtaining the session token from our Identity System along with the associated account identifier, and then wrapping both together (and encrypting). This single wrapped token is then returned to the client application and subsequent requests to our various OTT microservices are carried out as normal using this wrapped token.

Each OTT microservice includes a common Token Wrapper Client library which obtains and stores the current (and past) encryption keys from the Token Wrapper Service on start-up, and periodically contacts the Service to retrieve updated keys when necessary. The Client library allows each microservice to obtain the account identifier from a wrapped token without any real-time dependency on either the Identity System or the Token Wrapper Service itself (see Figure 5).

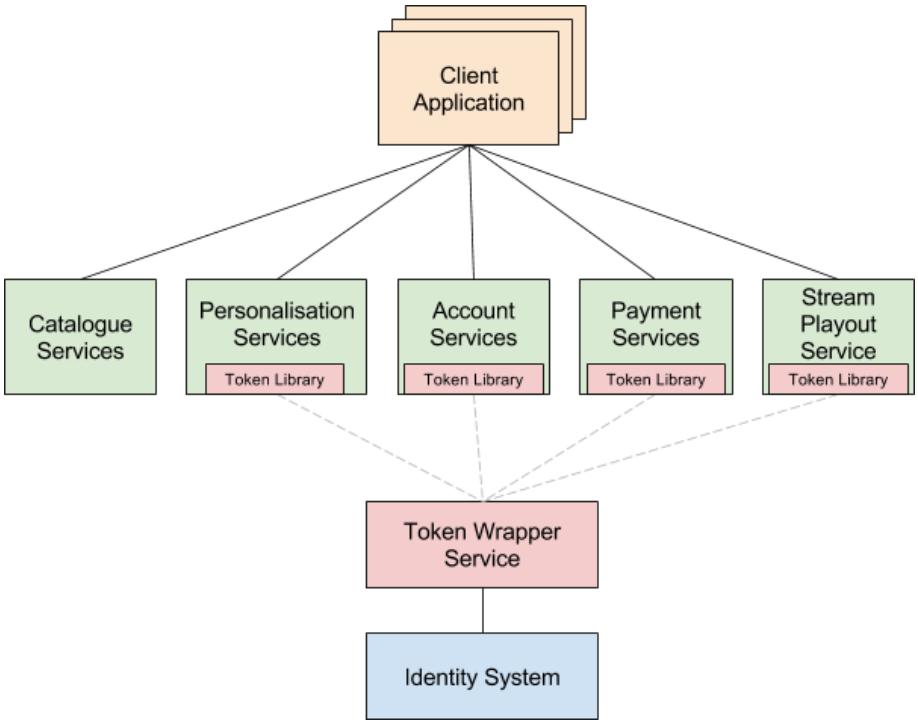

Figure 5: Token Wrapper Client Libraries removing real-time dependency on the Identity System

Should the Identity System and/or the Token Wrapper Service become unavailable, all OTT microservices can continue to serve requests for authenticated users, thus removing the single point of failure in our architecture.

III. ENTITLEMENT VALIDATION

The permissions granted to our users to consume different groups of content are termed Entitlements. Entitlements are created upon a successful purchase of a product and revoked at the end of their term (normally determined by a duration set at point of purchase).

Entitlements are critical to the playout initiation journey in order to verify that the user is permitted to consume the content that they are attempting to play. If the user doesn't hold the appropriate entitlement, then the playout request is refused and the user is encouraged to purchase a product to gain the entitlement.

This previously meant that at the point of playout we had a dependency on our entitlements store being available. If we were unable to verify that the user was entitled to play the content, then we prevented them from doing so - including blocking entitled users incorrectly.

Additionally, in order to drive display logic in our user interface prior to any CTA (Call to Action), we verified users' entitlements on page load. This introduced a large scalability concern and a dependency on our entitlements store for just browsing the catalogue.

In order to resolve these scalability and resiliency concerns, we further enhanced our custom authentication token wrapper to additionally allow us to carry user state within the token,

including entitlement information (see Figure 6). Entitlements are now looked up just once at point of authentication, then embedded into the token wrapper and passed from the client on every request, eliminating the real-time dependency on the entitlement store.

Figure 6: Entitlements within Token Wrapper

We call it the 'Bring Your Own State' model. In degraded operation mode, the client can now securely tell the server what the user can access, rather than the other way around. We can even fall-back to previous user state from an existing token stored on the device if the system has already degraded prior to authentication.

The token wrapper enhanced with user state provides both resiliency and performance gain, as fewer database queries and internal API requests are necessary between our microservices when the required state is passed in from the client. Our Token Wrapper Service even stores a last-known-good of all previously authenticated users' state of entitlements should the entitlement store be unavailable at point of authentication.

IV. STREAM INITIATION

The final piece of the discovery and playout journey is stream initiation. At Sky, this is the process by which the client application obtains the location of the video assets on the CDN and acquires a DRM (Digital Rights Management) licence in order to decrypt and playback the content on the user's screen.

Working to the assumption that our CDN guarantees high availability of the encrypted video assets themselves, the remaining critical components in this critical last step to playout are our Stream Playout Service and DRM licence servers. Both must be available in order for a successful stream initiation on a user's device.

We have re-architected the Stream Playout Service to ensure that every dependency on other systems, components or data stores has been appropriately circuit-broken guaranteeing fall-back behaviour in case of subsystem failure. This means that wherever we are unable to acquire the necessary data or decision, we ensure that we have a fall-back to allow us to proceed without impacting the user's experience. In all instances we aim to fail in the user's favour, protecting their Stream in every eventuality by leveraging our investment in our Bring Your Own State resiliency model and ensuring last-known-good caches exist.

Our next step to even greater resiliency in this area is a dissimilar redundancy solution. Such a solution would see a completely alternative implementation of the Stream Playout Service's capabilities, completely rewritten, utilising different hardware and deployment locations, different technologies, different approaches and different dependencies / data sources. The theory behind this solution is that if any aspect of the Stream Playout Service becomes unavailable, its dissimilar equivalent in the redundant system is highly unlikely to be affected, allowing us to maintain service by switching to the redundant system.

A similar approach has also been taken for our DRM licence servers. Without a valid DRM licence, client applications would be unable to decrypt our encrypted video streams, leaving our users unable to watch any content despite being able to discover the content and being entitled to watch.

Currently, our resiliency approach for our DRM licence servers follows industry standards for resiliency, including active-active cross-data centre deployment and redundancy. Future development however, will bring a dissimilar redundancy solution similar to our approach for the Stream Playout Service, maximising our availability for delivering a DRM licence to the user device.

SUMMARY

From login, through content discovery, entitlement validation and playout, our services can be heavily degraded whilst keeping our users able to consume content. With just a few innovative resiliency design patterns, we have maximised the availability of our OTT propositions, remaining highly available even in the case of failure in dependent systems.

By separating cacheable and uncacheable aspects of content discovery, we can guarantee at least a non-personalised discovery experience. By wrapping an account identifier and user state within the authorization token we can eliminate the real-time dependency on our identity and entitlement systems. And by developing dissimilar redundancy solutions we can maximise our playout initiation availability.

Learning from our previous mistakes (1), we continue to perfect our resiliency approaches, and our focus on the customer experience will ensure that our resiliency decisions and design trade-offs will always be in the user's' favour.

REFERENCES

1. Maule, T. C. 2016. NOW TV and Linear Streaming: The unpredictable scalability challenge. https://youtu.be/E7PGQJCGOPg, 14 Jul 2016.