Just-In-Time Packaging vs. CDN Storage

Yuval Fisher
RGB Networks

Abstract

Operators delivering video-on-demand (VoD) to multiple devices using HTTP streaming must select between two options: store assets in multiple formats to be delivered via a content delivery network (CDN), or utilize an on-the-fly, or just-in-time (JIT), packaging to convert VoD assets into the required client format when it's requested. This paper discusses the benefits of JIT packaging and then proposes a model to evaluate the costs associated with each approach, discussing the parameters associated with various use cases. We also discuss the implications of the cost model for more general edge processing, such as just in time transcoding.

INTRODUCTION

HTTP streaming of video based on protocols defined by Apple, Microsoft and Adobe (see [HLS], [MSS], and [HDS]) has led to the development of a new component in the video delivery chain – the packager (sometimes also called a segmenter or fragmentor). This component creates the segmented video files that are delivered over HTTP to clients that then stitch the segments together to form a contiguous video stream. The packager may be integrated into the encoder/transcoder that creates the digital encoding of the video, but often it is a separate component. Separating the components has various advantages, including the ability to capture the output of the encoder/transcoder as a mezzanine format that can be reused for packaging in both live and off-line scenarios.

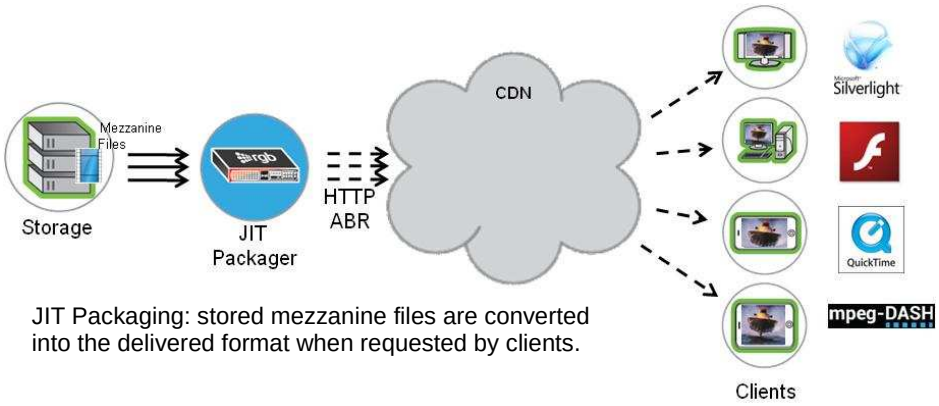
The emerging MPEG DASH (see [DASH]) standard attempts to standardize and unify these protocols under one open specification umbrella; but in the near term, DASH adds more formats that service providers may need to address, since HLS, MSS, and HDS will not disappear immediately, if ever. In fact, DASH has several profiles that have very different underlying delivery formats, so that it may be necessary for packagers to serve not just HLS, MSS and HDS, but an MPEG-2 TS DASH profile and a base media file format DASH profile as well.

In this paper, we focus on one specific use case: just-in-time packaging (JITP), which is applicable for VoD and network digital video recorder (nDVR) applications, including catch-up and restart TV. In all of these applications, each client makes a separate request to view video content (typically from its beginning), so that unlike broadcast video, viewing sessions are independent.

When delivering HTTP streams, two options are possible: either the assets are stored in an HTTP-ready format, so that clients can make HTTP requests for video segments directly from a plain HTTP server. Or, assets can be stored in a canonical (or mezzanine) format which is then converted to HTTP segments as the client makes requests for them – just-in-time. The first option is more disk storage intensive, while the second is more computationally intensive.

Just-in-Time Packaging

In a typical JITP use case, VoD assets are created from live content that is first transcoded into MBR outputs and captured by a "catcher" component that converts the live streams into files stored in a chosen mezzanine format. Alternatively, file assets, rather than live streams, are transcoded into a mezzanine format which uses H.264/AAC for the video/audio codecs and a pre-selected container format. MPEG-2 TS container



JIT Packaging: stored mezzanine files are converted into the delivered format when requested by clients.

format is a natural choice for the mezzanine files, since it can contain much of the signaling present in the original signals in an industry-standard way.

Clients that request a stream from the JIT packager first receive a client-manifest describing the available profiles (bitrates, resolutions, etc). The JIT packager will create the manifest when it is requested the first time; subsequent requests are served from a cached copy. Clients subsequently request specific chunks from the packager which extracts the requested chunks from the mezzanine files and delivers them to the clients. Thus, each client request is served from the JIT packager – the more subscribers that exist, the more JITP capacity is needed.

Selecting a Mezzanine Format

What characteristics should the mezzanine format have? It should:

- be computationally simple to package just-in-time;
- retain metadata in the input streams;
- be a commonly used format with an ecosystem of creation and diagnostic tools.

There are two commonly used mezzanine formats: ISO MPEG file format and MPEG-2 TS files. The first has the advantage that multi-bitrate output can be stored in just one file, as opposed to as many files as profiles, as happens in the MPEG-2 TS case. This makes management of files easier. However, MPEG-2 TS files can provide standardized ways to store many types of commonly used metadata, e.g. SCTE-35 cues for ad insertion points or various forms of closed captions and subtitling, and these are not standardized in the MPEG file case. Moreover, MPEG-2 TS would normally be the format captured in the NDVR use case, and the ecosystem of support tools (e.g. catchers, stream validation tools, stream indexing) is larger in the MPEG-2 TS case. Thus, MPEG-2 TS files make a better mezzanine format than ISO MPEG files in most cases.

WHY USE JITP?

There are a number of reasons why JITP may be a better alternative to pre-positioning assets in all final delivery formats.

Storage Cost Favings

When multiple HTTP streaming formats are used, every asset must be stored in multiple formats, with associated storage costs. This is especially true for network DVR where legal requirements in some regions mandate that separate copies are stored for each customer.

Format future-proofing

The HTTP streaming protocols in use today are still evolving; using JITP of mezzanine-format assets eliminates the need to re-package VoD libraries when these formats change. Changes in formats can be addressed

via software updates of the JIT packager, which can then also manage a heterogeneous ecosystem of different format versions (e.g. various flavors of HLS). This is a huge boon to operators who must otherwise decide on a specific version of a format and thus potentially miss features in new format versions or not serve subscribers who haven't updated their video players.

Single Workflow

Using JITP for VoD with a caching CDN can automatically lead to an efficient distribution of contents in the CDN – that is, the caching of short tail (or commonly viewed) assets in the CDN and the use of JITP for un-cached long tail (rarely viewed) assets. This ensures that new assets automatically migrate into the CDN without requiring a separate offline packaging step in the workflow, as well as a separate, offline determination of which assets are short tail and which are long tail.

Graduated Investment

New VoD service offering using storage rather than JITP would require all assets to be stored in all formats up front, leading to large initial capital expenditure. With JITP, operators can add VoD capacity as the number of subscribers grows with capital expenses that match subscriber growth and revenue.

Unicast Relationship

Because the JIT Packager has a unicast session with the client, it can be used to encrypt VoD sessions uniquely for each client. Moreover, other unicast services, such as targeted ad insertion, can be integrated into the packager. Note that when chunks are encrypted per user, they cannot be cached in the CDN.

COST MODEL

In this section we describe a cost model for comparing storage with JITP. The cost model depends on whether the VoD streams are CDN-cachable or not, as could be the case, for example, if they are encrypted per user. If they are cacheable, the storage in the core used to store the assets, as well as the storage in the tiers of the CDN, can be compared to an equivalent JITP capacity. When the assets are not cacheable, the JITP cost is higher, since both the short and long tail content must be packaged just-in-time.

Cacheable Assets: Storage vs. JITP

A simple cost model (see also [Fisher]) can be created based on a few assumptions. First, we assume that short tail content will be served from the CDN and will not require JITP.

The cost of storing the complete library in multiple formats depends on multiple factors listed in the table below:

	Description	Values
L	Library size (hours)	10K-150K
B	MBR bitrate (Mbps)	10
S	Number of subscribers	100K-10M
P_c	Peak concurrency	5%
P_L	Percentage of long tail requests	10%
C_s	Cost of storage ($/TB)	US $2,000
T	Number of CDN storage tiers	2
F	Number of ABR formats	3

The total cost of storage C_{ts} is then:

$$C_{ts} = C_s \times T \times F \times 3600 \times L \times B \times 10^{-6} \times 1/8$$

For example, a library of 20,000 hours stored in three formats at the core with two CDN points of presence (or CDN roots or different CDN tiers) would cost $1.08M.

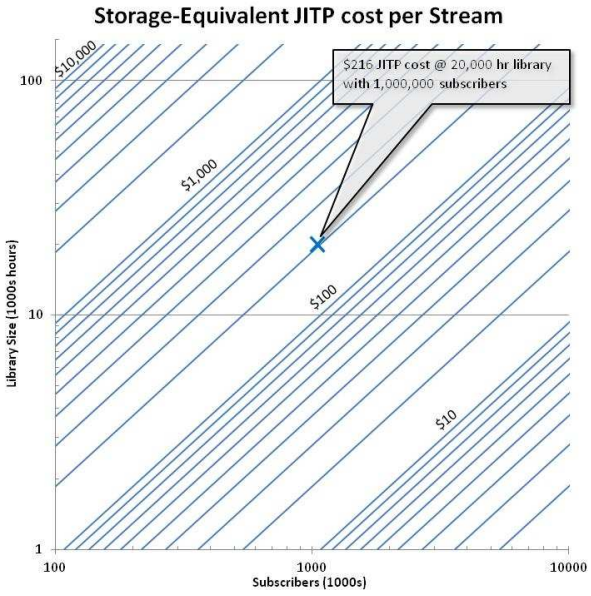
The equivalent cost C_{jitp} of serving a JITP stream rather than using storage is the total

storage cost divided by the number of long tail stream requests:

$$C_{jitp} = C_{ts} / (S \times P_c \times P_s)$$

So, in the example above, a million users would have an equivalent JITP cost per stream of $216.

We can look at the parameter space of library sizes and subscriber count to see where JITP provides value. Given that a high-end server can deliver hundreds of simultaneous JITP streams, the graph shows that the range of storage-equivalent JITP cost ranges from low (not even sustaining hardware costs) to very high (where significant savings can be achieved by delivering JITP streams rather than storage). Roughly speaking, the region where JITP leads to cost savings over storage is the upper left triangular half of the graph.



It's worth noting that JITP may incur an additional cost in inbound network traffic, at least when it is centralized. Of course, if JITP is not centralized, then the library must be stored multiple times at the edge, mitigating JITP's value. A complete analysis of every variation is beyond the scope of this paper, but the model described above can be easily modified and used in each situation.

Non-cacheable Assets

When VoD assets are not cacheable, the cost model can still be used by considering 100% of the assets to be long tail. This eliminates the benefit (and cost savings) of caching the short tail in the CDN.

Just-In-Time Transcoding

The cost model does not discuss what type of processing is done in the network – only its cost compared to storage. Since the computational density of transcoding is about two orders of magnitude less than for packaging, the cost graph shows which regions in the subscriber library parameter space are suitable for transcoding as well; this is (roughly) the upper-left triangular portion of the graph that supports processing costs above $1000 per stream.

CONCLUSION

JITP may offer significant cost savings over storage, but its real value may be in other benefits: a simplified workflow, per-subscriber encryption based on unicast delivery, future-proofing against the evolution of formats, and investment and growth in capacity that is commensurate with subscriber growth.

REFERENCES

[Cablevision] 2nd Circuit Court ruling on network DVR
http://www.ca2.uscourts.gov/decisions/isysquery/339edb6b-4e83-47b5-8caa-4864e5504e8f/1/doc/07-1480-cv_opn.pdf

[Fisher] Comparing Just-in-Time Packaging with CDN Storage for VoD and nDVR Applications, Proceedings of the Canadien SCTE, March 2012.

[HLS] HTTP Live Streaming, R. Pantos, http://tools.ietf.org/html/draft-pantos-http-live-streaming-06

[MSS] IIS Smooth Streaming Transport Protocol, http://www.iis.net/community/files/media/smoothspecs/%5BMS-SMTH%5D.pdf

[HDS] HTTP Dynamic Streaming on the Adobe Flash Platform, http://www.adobe.com/products/httpdynamicstreaming/pdfs/httpdynamicstreaming_wp_ue.pdf

[DASH] ISO MPEG 23009-1 Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats