ABR Delivery Architectures and Virtualization

Santhana Chari, Ph.D.
Senior Director, Engineering
ARRIS
santhana.chari@Arrisi.com

Niranjan Samant
Principal Systems Engineer
ARRIS
niranjan.samant@arrisi.com

Introduction

This paper discusses two emerging trends in video processing delivery, namely, migration of various video processing functions to the network cloud to leverage advances in virtualization and dynamic packaging techniques for adaptive bitrate (ABR) delivery of video.

ABR delivery requires various functional processing blocks such as transcoding, packaging, encryption, caching and eventual delivery to clients. Details of individual functional blocks are presented. Today, linear and on-demand video services are being offered using IP based ABR delivery by means of protocols such as HTTP Live Streaming, Smooth Streaming, or Dynamic Adaptive Streaming over HTTP (DASH.) These services are growing at a very healthy rate and are expected to continue to grow in the coming years as the delivery to the main screen starts using IP based video delivery. In addition to classic services such as linear TV and Video on Demand, IP based services such as network DVR, StartOver, and Lookback are becoming increasingly popular. As the scale of delivery and the type of new services grow, the need for performing certain functions such as transcoding and packaging in an on-demand or just-in-time (JIT) fashion will arise. We present different architectures using static and just-in-time packaging and describe their applicability to different services.

This paper further discusses the advantages, key requirements and challenges in implementing various processing functions in software and hosting them in a virtualized environment. Certain functions like caching, CDN based delivery and packaging are naturally amenable to virtualization. On the other hand, more computationally- intensive operations such as video and audio transcoding have historically been performed on custom hardware. However, there are several recent developments in general purpose processors that are making software based- and hence virtualized processing- of video compression possible, the details of which are presented herein.

ABR Processing & Delivery Functional Blocks

The primary functional blocks that make up an Adaptive Bitrate Delivery pipeline are multibitrate transcoding, packaging, content protection (encryption and DRM), storage, streaming and client playback. These are quite well known but described very briefly below for completeness. ABR formats considered in this paper are Apple's HTTP Live Streaming (HLS) [R1], Microsoft's IIS Smooth Streaming (HSS) [R2], and the MPEG group's Dynamic Adaptive Streaming over HTTP (DASH) [R3]. Within DASH we may further differentiate as DASH-TS, which covers MPEG-2 Transport Stream based DASH profiles, and DASH-ISOBMFF, which covers ISO Base Media File Format [R4] based profiles. Figure 1 shows the functional blocks that make up the ABR processing, delivery and playback pipeline.

Multibitrate (MBR) Transcoding is most commonly the process of creating multiple versions of a single media stream or asset. Media typically consists of video and audio components; it may also contain closed captions or subtitles. An MBR transcoder will ingest a single such MPEG-SPTS (Single Program Transport Stream) stream [R5] and create multiple SPTS outputs, each containing the same audio, but the video component in each output stream is of a different resolution and or bitrate. For a given service or asset, MBR transcoded outputs are IDR (Instantaneous Decoding Refresh) frame and PTS (Presentation Timestamp) aligned.. For ABR streaming the video is encoded in H.264 [R6] format and the audio is encoded in one of the version of AAC (Advanced Audio Coding) [R7] standard.

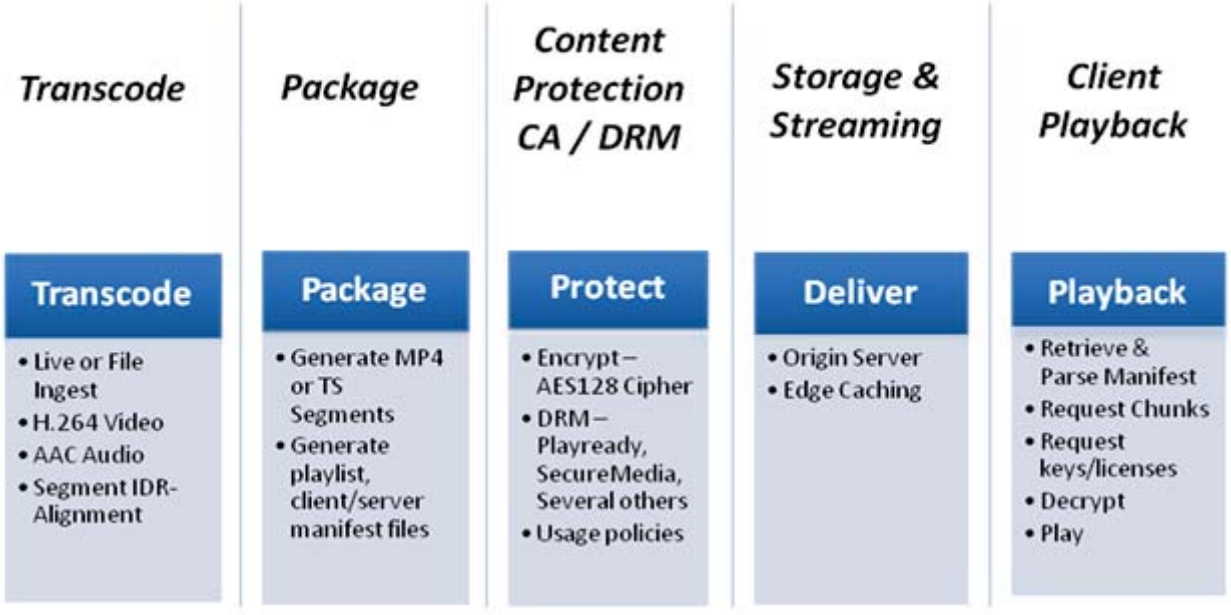

Figure 1. ABR Delivery Processing and Playback Pipeline

Packagers ingest MBR transcoded content and convert it to one or more ABR streaming formats. Each ABR format output essentially consists of temporally aligned chunks/segments/fragments of each constituent MBR transcoded stream and one or more accompanying text files, called manifests or playlists. In DASH the playlists are called Media Presentation Descriptions (MPD). Playlists contain details about different types of streams available in the ABR output and information in the form of URLs for how to request them in chunks. Packagers usually publish their ABR outputs to Origin Servers (OS) in a Content Delivery Network (CDN). Depending upon the ABR format and its specifications packager output chunks are either Transport Stream (TS) segments or MP4 fragments. MP4 fragments are based on the ISO Base Media File format (ISOBMFF) [R4].

In order to protect ABR content before it is made available to subscribers and their client devices, it may be encrypted and DRM rules may be applied. Packagers most often perform encryption of ABR content. Most ABR formats today require certain modes of AES-128 encryption. HLS specifies the AES-128 Cipher Block Chaining (CBC) mode for encryption. HSS specifies the AES-128 Counter (CTR) mode for encryption. MPEG-DASH output is encrypted based on Common Encryption (CENC). There are two CENC specifications either released or in progress also based on AES-128 modes of encryption, one for ISOBMFF-CENC [R8] and the other for MPEG2-TS-CENC [R9]. Many DRM vendors offer key and license servers to enable encryption at packagers, support delivery of keys securely to clients as well as application of DRM policies.

Packagers publish ABR content to origin servers, which are central entry points into Content Distribution Networks (CDN). CDNs provide storage and distribution of ABR content in an efficient manner. At the edge of CDNs are edge servers which may deliver content directly to clients, or the edge servers may terminate into MSO/Telco access networks that deliver the content to subscriber premises/devices.

In ABR streaming, clients request content for playback via HTTP [R10]. When clients first tune to any ABR channel or asset they receive a compatible playlist. Clients parse the playlist and then use URLs within the playlist to request content chunks. They also use information in the playlist to retrieve keys/licenses necessary to decrypt ABR content. An ABR playlist provides the client multiple choices for playout of the same ABR stream, each at a different resolution/bitrate. ABR clients apply heuristics to determine the most optimal stream to request, given current network conditions.

Besides the above mentioned functional blocks, there are supporting functions that enable ad-insertion, black-out or other content replacement, analytics and reporting, watermarking, and more.

Types of ABR Services/Applications

Today ABR streaming services complement high quality TV delivery to the STB (Set Top Box) by making the same services available on second and third screens. In the future they also hold the promise of supplementing high quality TV delivery to STBs, resulting in unified IP delivery to all devices inside and outside the home/business. In this section we discuss the various applications in which ABR streaming is/can be deployed or the types of services that ABR streaming enables.

Live Linear Applications (Live to Live): Linear broadcast (BC) services are delivered to ABR clients via this application. This includes off-air broadcast services, linear Cable/Telco TV services including sports and premium content channels.

In this application, live linear services are ingested and passed through the ABR delivery pipeline where, after packaging, they are published to an origin server/CDN. Authorized clients wanting to playback any of the available channels are directed to appropriate edge caches on the CDN to retrieve the service in ABR format. Edge cache "misses" for content are fulfilled by the origin server. This application can support content protection, ad-insertion, black-out and emergency alert.

Live on-Demand Applications (Live to File): These applications of ABR streaming delivery cover those services that straddle the boundary of on-Demand and live TV. These applications are created around live BC services where a subscriber may watch live linear BC TV but the next moment he may scroll back in time to watch the same program from its starting point. This allows the subscriber to watch a live program and also its partially recorded version on-Demand.

Based on the Live on-Demand application a service provider may offer its subscribers services such as or similar to StartOver TV or dynamic cDVR (cloud Digital Video Recorder). In an ABR service similar to StartOver TV a service provider may offer selected channels to subscribers where they can watch a live program already in progress from its starting point. In the cDVR application a subscriber selected program is stored in the cloud by the service provider and made available on-demand at any time for delivery over ABR streaming. If a subscriber starts watching a cDVR recording while the program and recording is in progress, this is called dynamic cDVR, and if a subscriber starts watching a cDVR recording after the live BC of the program is over it becomes a static cDVR recording.

Offline on-Demand Applications (File to File): These applications of ABR streaming delivery cover those services that are static at the time of streaming. Programs or channel content may be stored in the network or cloud by the service provider and a subscriber may request streaming of those stored services in an on-demand fashion at any time.

ABR streaming services that fall under this category are Video on-Demand (VOD), static cDVR, and LookBack TV/CatchUp TV like services. Classic VOD service does not need any further explanation. Static cDVR is streaming of a cDVR recording after the recording is complete and the asset has become static. LookBack or CatchUp TV is a type of service where a service provider makes content from selected broadcast channels available for on-demand viewing for a fixed period into the past, say one week into the past. Subscribers may scroll back up to one week or a few days in the past and view any program.

In the next section we shall explore architectures that make ABR delivery of the services/applications discussed above possible.

ABR Delivery Architectures

In this section we start with a simple ABR delivery architecture with a linear packager and then successively build on it to come up with an architecture that supports all services/applications mentioned in the previous section in the most efficient manner, using linear and just-in-time packagers (JITP). Figure 2 shows a linear packager publishing ABR streams to an origin server that feeds a CDN. ABR client devices are first directed to edge servers and cache misses at the edge are fulfilled by the origin. In this architecture live multibitrate (MBR) transcoded streams may be converted to live or VOD assets by the packager and posted to the origin. Pre-transcoded MBR files also may be ingested by the packager, packaged and published to the origin or storage that sits off the origin, for retrieval by clients. The linear packager can support live to file and also file to file type of applications.

This basic architecture can be configured to support live to file services as follows. The linear packager can convert incoming live services to VOD assets in real time and publish them to the origin. This architecture requires a large amount of storage at the origin and at the edges. Some of the control plane functional blocks are assumed and not shown in the figures. The linear packaging architecture may not be the most efficient for all types of applications. Live to live services should be served by the linear packaging architecture but for the other applications the linear packaging architecture is a good fit when the size of the deployment in terms of the amount of content and the total number of client devices or sessions is relatively small. Larger deployments providing all types of ABR applications should consider adding Just-in-Time Packaging (JITP).

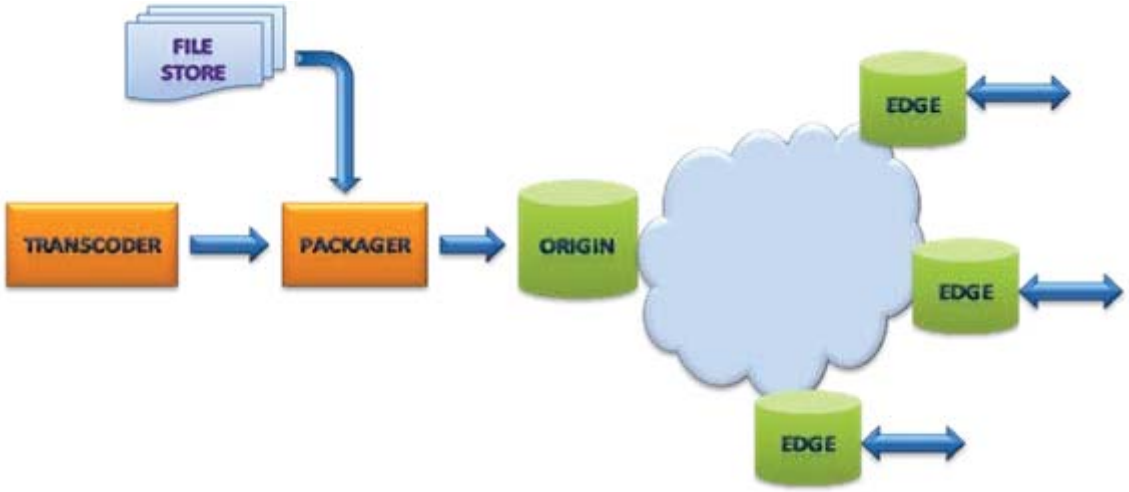

Figure 2. Basic ABR Delivery of Services with Linear Packager

The addition of Just-in-Time Packaging or Repackaging (JITP) enables many of the ABR delivery applications in a more efficient manner. Efficiency gains are realized in the form of less storage and less bandwidth requirements, thereby enabling the service provider to offer a larger selection of content in a more flexible manner. In addition, since packaging for a specific client is done at the time of playback, personalization of content and advertisements is efficacious.

Figure 3 shows the addition of Just-in-Time Packagers to the ABR delivery architecture. As can be seen, the JITPs can be located both centrally in the architecture and at the CDN edges. For the applications that are supported by JIT Packaging, the linear packager processes either live or VOD content and publishes it in a mezzanine ABR format to the origin. A typical mezzanine ABR format could be a MPEG-DASH TS profile or interoperability point. When a client wants to play an asset or service that is supported by JITP, the manifest URL it is given, triggers a JITP instance. The JITP instance locates the mezzanine ABR files and packages them for the specific device profile based on individual requests from the client. The JITP processes and packages only the segments of the bitstream variant that are being requested by the client. It does not publish all segments of all bitstream variants.

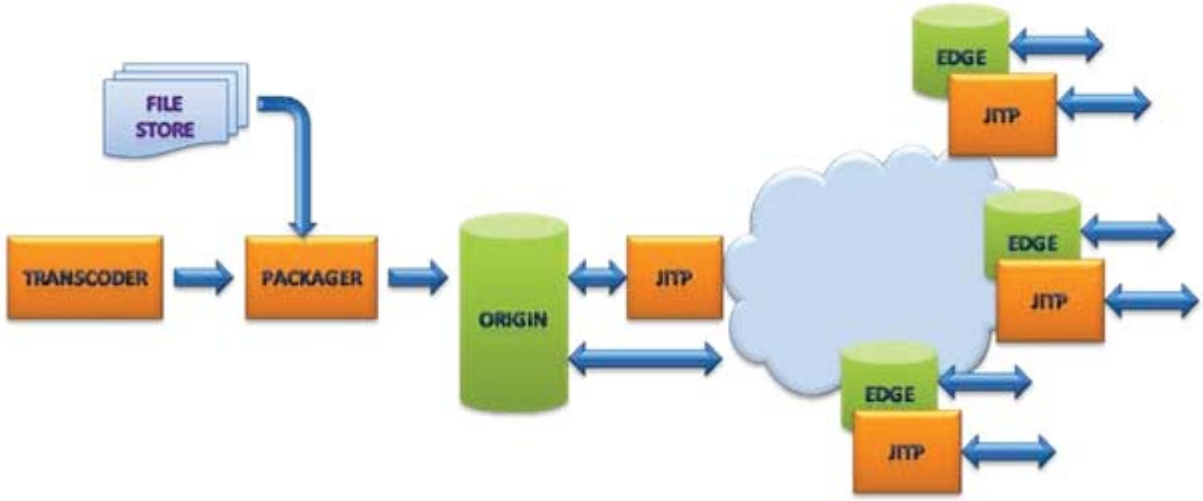

Figure 3. ABR Delivery Architecture with Linear and Just-in-Time Packagers

JITPs in the architecture may be utilized to support ABR streaming applications such as live to file and file to file. The linear packager may continue to support live to live applications. In addition, since the JITPs package content for the requesting device on the fly in a just-in-time fashion, their output can be highly customized or personalized; more on this later. The centrally located JITP can package content for a specific client device but due to its central location it may be leveraged to package popular content in such a fashion that it may be cached in the CDN or edge servers downstream from it for playout by multiple requesting clients. JIT packaging starts only when at least one client requests a given asset, as the JITP packages for the client its output is cached in the CDN. Any new clients requesting the same content may access the cached content. Cache misses in the edge or CDN are directed to the JITP instance and it fulfills the requests. Because the origin only stores a single mezzanine format and the JITP only packages and delivers specific segments requested by clients, considerable storage savings are realized at the origin. Further since the JIT packaged content is cached at the edges CDN bandwidth is used more efficiently. This centralized JIT packaging with caching at the edge is best suited for popular content. This type of ABR delivery is not suited for long-tail content since it does not require caching in the CDN; however the use case is not precluded.

The centrally located JITP is suited for packaging of popular content for the following ABR streaming applications such as StartOver TV, LookBack TV, and the popular VOD. In countries where network DVR (nDVR) or cloud DVR (cDVR) content storage is common, copies of the central JITP can be utilized for packaging both dynamic cDVR and static cDVR services.

The JITP at the CDN edge is suitable for packaging either long-tail content or for packaging content that is meant only for a specific client/subscriber or is highly personalized for a specific client/subscriber. Live or offline content is pre-packaged by the linear packager and published to the origin server in a mezzanine ABR format, such as a given MPEG-DASH TS profile or interoperability point format. When a client that is authorized to access this content makes a request for playout, the packaging request is directed to a JITP instance at the edge. The JITP instance locates the mezzanine content from the origin and starts JIT repackaging for the client on a request by request basis. The JITP only packages specific content requested by the client and the JIT packaged output need not be cached in the CDN for consumption by other clients.

It is quite clear that the edge JITP is predisposed to serve applications such as static and dynamic cDVR where unique copy requirements have to be met. Further the JITP at the edge may be deployed for services such as StartOver TV, LookBack TV, and VOD for long-tail content (i.e., content that does not need to be cached.) The edge JITP architecture also results in considerable storage savings at the origin or the edge.

Even though we have presented specific roles for the JIT packagers at the center and at the edge, they do not have to be exactly so and may be interchanged. Further, for very popular content the mezzanine ABR format may be pre-positioned at the edges at times when the CDN bandwidth needs are low. Then, during peak periods JITPs at the edge can serve clients on request. This spreads out bandwidth usage in the CDN over peak and non-peak periods.

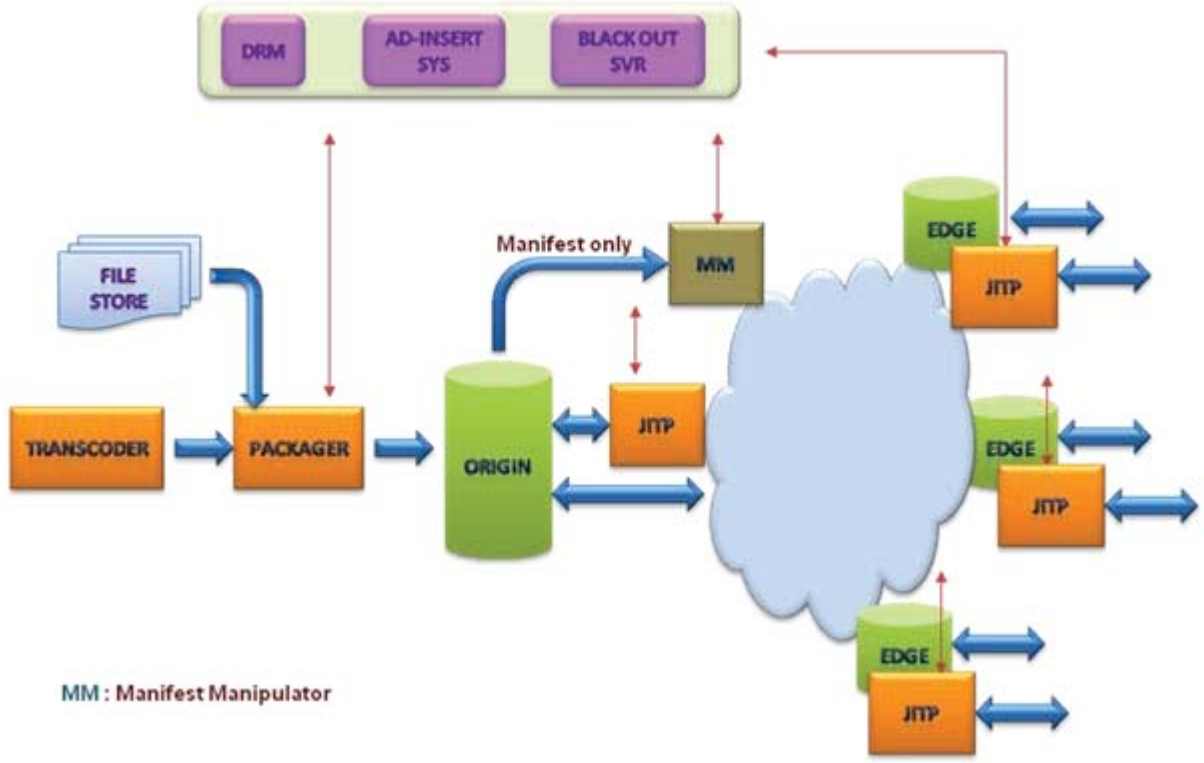

Figure 4. ABR Delivery Eco-System

A more complete ABR delivery eco-system is shown in Figure 4. DRM and license servers are used to provide encrypt/decrypt support. ABR streams are encrypted at the linear packager or the JITPs. They are decrypted at the end client just before playout. Ad-insertion in the ABR delivery architecture is performed at the regional level or at the individual subscriber/client device level. The manifest manipulator (MM) performs ad-insertion for multiple ABR formats by manifest manipulation. Ad-insertion is mainly achieved using SCTE-130 methods or using the VAST (Video Ad-Serving Template) specification. In linear packaging the manifest manipulator provides targeted manifests or playlists to clients. The JITPs can also provide ad-insertion support. Advertisement or other targeting/personalization may be at the individual subscriber/device level or at the regional level. Content blackout and replacement is implemented in a manner similar to ad-insertion by manifest manipulation. Additional safeguards to prevent spoofing of blacked-out content can be implemented via manipulation of encryption keys.

Advantages of Virtualization

Virtualization has become all too prevalent in conventional IT environments that utilize data centers. With virtualization, the notion of tight association between server hardware and the application running on it has been replaced by abstracting out the hardware and making the operating systems independent of the underlying hardware. This is

accomplished by running a virtualization layer of software called the *hypervisor* directly on the hardware, which in turn manages one or more guest operating systems that can be run on top of the hypervisor. Both commercial (VMware ESX, Citrix Xenserver, etc.) and open source hypervisors (Xen, KVM, etc.) are available. Most hypervisors support x86 processors which come with built-in support for virtualization. Support for other processors such as the ARM and Atom are available in some of the hypervisors.

Virtualization offers several operational advantages in IT data centers that extend to video processing delivery. Given the rapid increase in the processing power available on general purpose processor, video processing has been increasingly performed on commercial off-the shelf servers. Details of virtualization in the context of video transcoding and packaging will be presented in greater detail in the following sections. Main advantages of virtualization are:

- Efficient utilization of hardware resources: The clock frequency of CPUs has been increasing for decades due to improvement in processor technologies. More recently the number of cores available on a single CPU die has gone up significantly. Today server-grade CPUs can have up to 10 or 12 x86 cores, with the number of cores increasing with each successive generation of processors. This means that applications running on these CPUs do not use up all the existing server resources such as the processor, memory and storage. Some applications may manage to use all the resources for a small amount of time during high peak loads, but the average usage still remains to be relatively small. Virtualization allows us to consolidate and utilize the unused server resources by running several operating systems and applications on the same server.

- Application Isolation: Improved utilization of server resources can be accomplished by running multiple applications on the same operating system. This approach, however, opens up the possibility of one application corrupting or affecting the operation of other applications running under the same OS. Running individual applications on their own dedicated Virtual Machines guarantees application isolation.

- High Availability: Most virtualization platforms offer high availability solutions that continuously monitor the health of the hardware. When a server node fails, the VMs running on that node are restarted on another healthy node with minimal downtime.

- Operational Simplicity: Virtualization vastly improves and simplifies the day-today operations. Having different applications run on custom hardware devices with

unique management functionalities require operators to be trained on various platforms. Supporting a single or class of servers simplifies training, maintenance, and support for high availability and redundancy

Virtualization Requirements

To enable cloud based delivery of ABR services, various processing functions shown in Figure 1 need to be virtualized. An application or a processing function can be successfully virtualized if they meet certain design and performance requirements listed in Table 1 below:

Feature	Requirements	Advantages
Modularity	Processing function should be partitioned into well-defined modular sub-functions. Individual modules should be composable to achieve required high-level functionality.	Modularity allows matching the processing functions to available resources.
Scalability	Processing functions should be capable of operating under varying amount of resources such as CPU cores, memory, storage, network bandwidth, etc. Any performance penalty resulting from reduced resource availability should be well understood and documented.	Scalability allows managing peak vs. average loading scenarios.
Data handling	Processing functions should be designed to be stateless as much as possible. Any common data should be stored outside the scope of the individual processing modules without unnecessary data replication. Processing functions should be driven by well-defined API interfaces.	Improves data integrity and helps data persistence. In case of failures, allows for quick recovery.
High Throughput	Processing functions should be capable of providing high throughputs (data processing capability) in software on commercial off-the shelf hardware without requiring specialized hardware or interfaces.	High throughput allows managing large number of services in a centralized operations center allowing for incremental addition of hardware resources as the services grow.

Table 1. High Level Virtualization Requirements

Virtualization of ABR Processing

In this Section, we will start with the functional blocks shown in Figure 1 and assess the amenability of the individual functions to be virtualized.

Caching and Streaming:

With ABR streaming, the Origin and Edge caching servers- for all practical purposes- function as web servers. They respond to content request from end clients or downstream caching servers and deliver manifest file and media chunks over HTTP. With the current generation of COTS servers, the caching and streaming functionality can be implemented in software to provide very high throughput. Hence the Origin and Edge cache servers are naturally amenable to virtualization.

Packaging*:*

As described earlier, packagers ingest MBR video streams to generate IDR aligned ABR media chunks and manifest files that are used by the client to request media chunks. In addition they also perform AES-128 encryption to provide authorized access to the content. These operations can be performed efficiently in software without the need for any custom hardware. Most current generation x86 processors have built-in support for encryption that obviates the need for any custom silicon-based processing. With state of the art COTS hardware it is possible to build packagers with high data throughput, given the computation requirements needed are relatively lightweight. As an example, we used a HP Proliant server with dual 6-core Intel Xeon processor to perform packaging. Figure 5 shows the resource utilization on this server for running different number of packaging jobs. Here the packager runs HTTP Smooth Streaming jobs by ingesting live MBR video streams where each MBR video consists of eight media variants. As it can be seen, hundreds of jobs can be run on a single server and the utilization of CPU and memory resources increase linearly with increasing number of jobs, making packaging operation very amenable to virtualization.

Based on the throughput and scalability performance presented above, packager software can be developed as a virtual appliance that may be scaled on-demand on the hosting servers. It should be noted that a Just-in-Time Packager (JITP) instance not only packages but also responds to HTTP requests from clients for content, so a JITP will have a packaging component and an http streaming component.

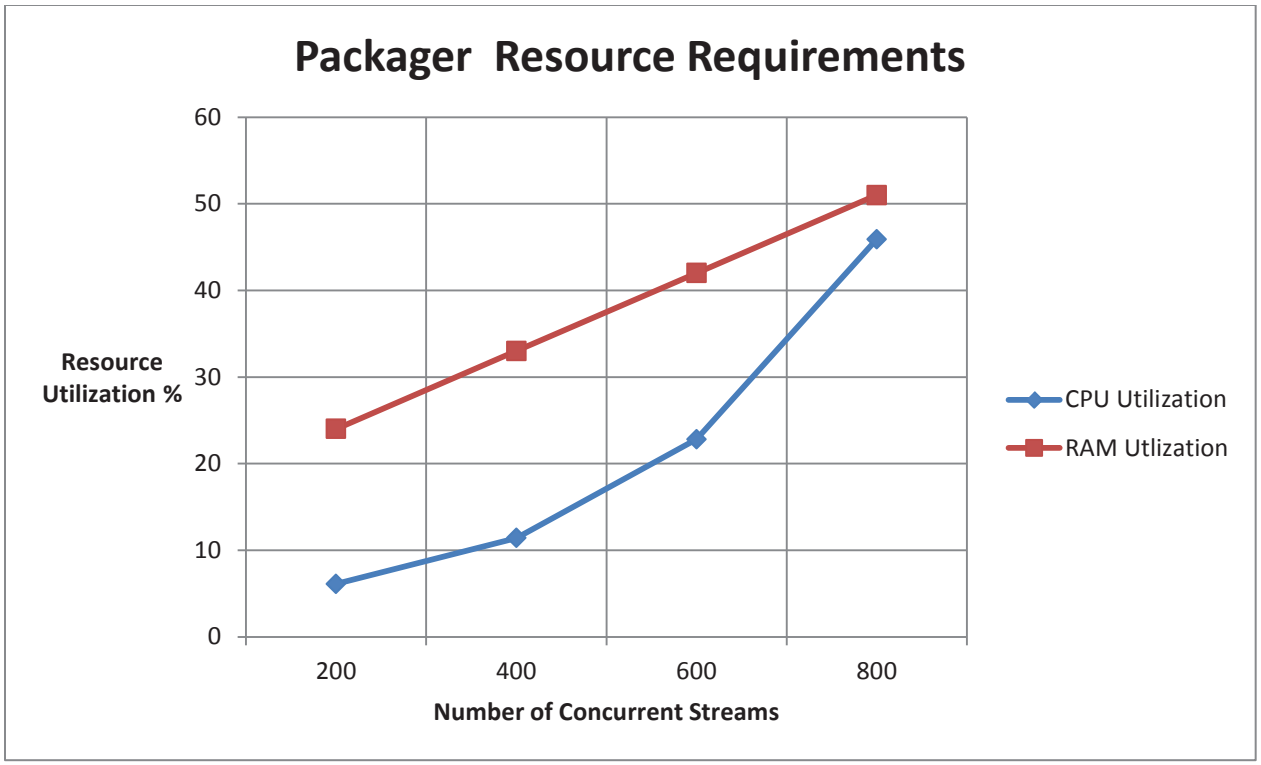

Figure 5. Packager Resource Scaling with Number of Streams

***Transcoding**:*

Video transcoding is the most computationally intensive part of the ABR processing pipeline and hence requires a substantially large amount of hardware resources compared to other ABR processing functions. Therefore successful virtualization of ABR requires a careful strategy to implement video transcoding in software. Based on the architectures presented in the previous section, one can see that ABR service delivery requires both Live and File-based video transcoding.

***Live and File based Transcoding**:* Video transcoding uses different work-flows depending upon the type of services. Two major and commonly used work-flows are Live and File based transcoding. These two work-flows pose different sets of requirements and constraints. Live transcoders ingest video in either baseband or pre-compressed formats and are required to process the video in real time, generating transcoded output streams. State of the art video head-ends at most MSOs have already migrated to ingesting video in pre-compressed format (either MPEG-2, AVC or a Mezzanine input format) and therefore do not require any custom interfaces apart from the standard IP interfaces. Live transcoders are required to operate 24x7 and therefore do not lend themselves to sharing server resources with other applications or leverage the peak vs average loads. However, all the other advantages of virtualization

mentioned earlier such as application isolation, hardware independence, high availability, and operational simplicity can still be leveraged for live transcoding. File based transcoding workflows are required for services such VoD and nDVR and file based transcoding lends itself naturally to virtualization. Here the video input that needs to be transcoded is available as a file and the work-flow requires the transcoder to operate in real-time, faster than real-time or slower than real-time to complete the transcode operation. Typically, each transcode operation is instantiated as a "job" in the workflow manager. The number of jobs submitted by the workflow manager at any given time can vary depending on the time of the day or day of the month. When several titles become available at the beginning of a view window for a subscription or pay-per-view based service, the number of concurrent jobs can be high resulting in a peak utilization of the file-based transcoder. Lack of requirement for 24x7 operation and real-time constraints, and varying loads makes file-based transcoding leverage all the benefits of software based implementation and virtualization.

***Software and ASIC based Transcoding**:* Historically, video transcoding and encoding have been implemented using both software and application specific IC (ASIC). In the early days of video compression using MPEG-2, general purpose CPUs did not provide enough processing capability or specialized instructions for video processing. Therefore, most -if not all- commercial, real time encoding was performed using custom ASICs. Over the last decade processing power on the x86 CPUs has increased dramatically. The increase in processing power first came in the form of higher clock speeds, followed by improved and specialized VLIW (Very Long Instruction Word) instructions for handling multiple video pixels in a single instruction, and more recently in the form of increasing number of cores in a single CPU. With this increased processing capability, CPU based solutions are now capable of offering video quality comparable to the ASIC based solution with the flexibility of using a general purpose processor for video processing.

	CPU	ASIC	Hybrid
Processing Technology	Multi-core CPUs	Fixed-function blocks with ARM/MIPS core system control	CPU cores, GPU Execution units and Fixed function blocks
Programmability	Highly flexible programming architecture	Limited programmability but offers tunable controls for various video processing functions	CPU cores and GPU are programmable. Fixed function blocks are tunable
Power requirements	High	Low	Medium
Processor upgrade cycle	High	Low	High
Software support	CPU vendor, Third party vendors and Open Source	ASIC vendor	CPU, Third party vendors and Open Source

Table 2. Criteria Comparison of CPU, ASIC, and Hybrid Based Transcoding Platforms

A more recent trend in x86 based CPUs is to add GPU (graphical processing unit) execution units to the CPU cores and fixed-function blocks to perform selected video processing functions [R11]. GPU cores are well suited to perform highly parallel, repetitive operations without frequent conditional exits. Fixed function blocks are used to perform operations such as entropy coding (CABAC), pre-processing and pre-filtering. Table 2 compares various performance/functional criteria against CPU, ASIC and Hybrid based platforms to be considered when making transcoding decisions. These "hybrid" CPUs that combine the CPU, GPU, and fixed function blocks are becoming very efficient in producing tunable video quality transcoders at modest to low power requirements. The higher processing throughput and flexibility offered by the newer generation of processors are expected to drive the implementation of transcoding functionality in a virtualized environment.

References

[R1] Pantos, R. P. & May, W. M., "HTTP Live Streaming", IETF Internet Draft Specification; Apple Inc.
[R2] MS-SSTR: Microsoft Smooth Streaming Protocol; Microsoft Corporation.
[R3] ISO/IEC 23009-1: Media Presentation Description & Segment Formats, Dynamic Adaptive Streaming over HTTP (DASH)
[R4] ISO/IEC 14496-12: Coding of Audio-Visual Objects – Part 12: ISO Base Media File Format
[R5] ISO/IEC 13818-1: Generic Coding of Moving Pictures and Associated Audio Information, Part-1: Systems
[R6] ISO/IEC 14496-10: Coding of Audio-Visual Objects – Parts 10: Advanced Video Coding
[R7] ISO/IEC 13818-7: Generic Coding of Moving Pictures and Associated Audio Information, Part-7: Advanced Audio Coding (AAC)
[R8] ISO/IEC 23001-7: Common Encryption in ISO Base Media File Format Files
[R9] ISO/IEC 23001-9: Common Encryption Format for MPEG-2 TS
[R10] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1", IETF RFC 2616.
[R11] Dunphy, R., Lei, R., Liu, Ping., "Utilizing Intel Quick Sync Video for High Density Video Transcoding in Communications Servers", Intel Developer Forum, 2013.