Virtualized Software Transcoding for Cloud TV Services

Yasser F. Syed Ph.D., Comcast Distinguished Engineer
Weidong Mao Ph.D., Comcast Senior Fellow & IEEE Fellow

Abstract

This paper will examine what will be needed for Cloud TV video services to utilize a virtualized software platform for orchestrating distributed transcoding workflows.

There are many encoding constructs that are now becoming realizable in this environment. Until recently, customized IC and hardware-based designs had been a practical approach to getting the job done using general-purpose servers. But these general processors are getting faster, more efficient, and more cooperative in multi-processor configuration groups.

Encoding/transcoding technologies, and especially those designed for Multiple Bit Rate (MBR) technologies using H.264/ Advanced Video Coding (AVC), as well as High Efficiency Video Coding (HEVC), are increasingly designed to become more flexible, adaptable, and scalable.

Furthermore, encoding/transcoding quality continues to increase, resulting in higher density growth at a much lower cost using general-purpose servers.

This paper discusses how transcoding workflows can be adjusted, so as to to take advantage of new server architectures, better networking, and distributed storage -- through a concept called a Transcoding Resource Manager.

OVERVIEW OF MULTI-SERVICE CLOUD TV ARCHITECTURE

Though all video services may look the same to a viewer, in actuality, cloud-enabled services differ from traditional service infrastructures in several ways. Those differences attributable to cloud-based video are contributing to increased service velocity, operational efficiencies, and infrastructure utilization. This will increase as the footprint for cloud-enabled services grows.

For the video industry, candidate services for cloud-based delivery can include:

- *Linear* - Supplying live broadcast & cable TV Channels, as well as live event programming across multiple device platforms
- *VOD (Video On Demand)*- Offering TV programs, movies, and other content on a subscription or transactional basis across multiple device platforms
- *cDVR (Cloud DVR)*- Providing services to record linear TV programs & TV series into a digital locker, to be viewed across multiple device platforms
- *cUI (Cloud UI)*- Providing single search & navigation services across all device platforms
- *EST (Electronic Sell Through)*- Providing a service to purchase TV programs, movies, and other content with storage in a digital locker and viewing across multiple device platforms.

To implement services in a cloud infrastructure, three basic building blocks are required: Computing, networking, and storage. Cloud gives the ability to process information, move it around in a timely manner, and to store the results for later use. This needs to happen both at a macro level (across a network) and at a micro level (within a device). To make the cloud platform optimized and realizable, the infrastructure needs to be suited to handle not just one type of information, but different kinds of information, and in a concurrent manner, in order to shift resources where needed.

The infrastructure to do this includes a hosted set of networked, general purpose servers located in a collection of Data Centers that are tied together through an IP network physically running over fiber, Ethernet, coax, and sometimes even wireless mediums.

Storage, in this sense, is a federated architecture, with large, centralized server farms, CDNs, and caching edge servers that implement the file migration strategies that assure data is available at the right place and time. The servers, in turn, house processors that can be used in collective, re-combinable sub groups, so as to perform tasks that vary from simple to complex.

The networking element allows data to be transferred within and between servers, as well as from network ingress points to network egress points, in an expedited manner, and thus making the task meaningful.

Consumers are already showing their enthusiasm for the benefits of increased bandwidth capacity and speed. The number of devices that can handle cable services has expanded recently, from set top boxes to PCs, game consoles, tablets, connected TVs, and cell phones. A DVR service no longer requires a hard drive in the home, because the content can reside in the cloud. A new UI or feature can be rolled out without testing exhaustively on each device platform, and the UI or feature can be designed to exhibit the same “look and feel” or “experience” on every screen. This can be accomplished by handling most of the processing in the cloud, with only rendering happening on the client device.

Peak hours for requested content can be scaled to handle larger demands through load balancing on processors, so as to better handle requests, and to strategically copy/migrate content closer to the viewer, which localizes bandwidth demands.

To achieve this performance, processors need not only to be fast, but also to work in a multi-core configuration. Bandwidth on both the server and on the core network needs to increase, and has -- from 10 Gbps to 100 Gbps. Now, 1 Tbps speeds are being approached [2]. Similarly, storage and accessibility to storage needs to improve its block accessibility for read/ write fetches, so as to create a federated storage/caching architecture with bulk access and retrieval of data.

These types of improvements will also lend themselves well to other tasks beyond just delivering services to a client/customer. In this paper, we will examine how content transcoding workflows, through virtualization, can adapt to and take advantage of a cloud infrastructure.

Virtualization, in a software transcoder sense, works by executing content workflows in a way that is agnostic to specific hardware, processing assignment, storage location or resource capacity. The operator simply interacts with the task/workflows with the monitoring and rendered results presented to them on an interface device. Using the cloud for virtualization means that resources through the network can be dynamically allocated to transcoding workflows, according to demands and schedule.

DESCRIPTION OF TRANSCODING WORKFLOWS

In early cable Linear/VOD workflows, content was targeted to be played back on a STB device through a VOD streamer or a

QAM-based live linear feed. For trusted content providers/ aggregators/ broadcasters, these distributable assets could be created and delivered to the service provider according to specifications including CableLabs ADI/VOD [5] and the SCTE's video specs [11]. This was good for scheduled material that would often come as a content refresh of a VOD catalog, or as live linear content targeted for the just the STB and no additional networked services.

For resource planning, the capacity of the transcoding system needed to be based on a peak transcoding rate covering file transcoding equipment for VOD and the recording of live feeds. That's because the peak transcoding rate is more dependent on the time constraints of processing the material than on catalog volume [4].

Another issue was getting the mezzanine assets or contribution feeds to the transcoding systems. Traditionally, this was done through a satellite pitcher/catcher infrastructure, or, more recently, with IP distribution delivery taking place over FTP-like connections (e.g. Aspera).

To get expanded Linear/VOD services to the STB, transcoding resources need to be continually increased to keep up with demand. Today, the demand for transcoding resources is exponentially exploding due to:

- VOD services being expanded to COAM (Customer Owned & Managed) retail devices (Tablets, PC, Cell Phones)[1]
- Turn-around demands getting tighter (to hours instead of days/weeks, C3/C7)
- Expanded Current Program content (over 200 TV shows for Fall 2013! [12])
- Expanded Backlog content
- Cloud DVR Services based on linear feeds
- EST (Electronic Sell Through) Assets w/ DLNA
- Higher Resolution/ Frame rate [1080p, UHD, 60 Fps] versions for content
- More diverse and increasing set of content providers

To handle this expanded scale of encodings, content workflows need to evolve to be faster. They also need to consider integrating priority of the job as part of the content workflows. This integration of job priority could be categorized into 4 content workflow types:

1. Just-in-Time (JIT) Content Workflow

A JIT content workflow requires a quick-turnaround service (immediate, minutes, hours). Immediate workflows may be for linear live services, and recordings of linear services. Examples of a "minutes & hours workflow" are highest priority VOD transcoding services, such as live events, or unscheduled VOD assets that require a mezzanine/contribution asset as a source.

2. Near-Time (NT) Content Workflow

A NT content workflow could be a high priority transcode for unbroadcasted material to handle live turn-around events (e.g. non-broadcasted Olympics events), broadcasted events that require a transcode, C3/C7 re-encodes, and repairs to pre-existing VOD assets.

3. Catalog Content Workflow

A catalog content workflow could handle scheduled assets that are a part of a VOD content catalogue refresh, EST offering, or catalog migration. This is typically not a high priority transcode unless a VOD presentation time is quickly approaching.

[1] To address CO&AM devices, adaptive streaming technologies generate 5-10 new encodes for each content asset. Each then needs to be wrapped in 1-3 DRMs and packaging technologies, dependent on device and delivery agreements.

4. Assembly Content Workflow

Assembly Content Workflows do not require additional transcoding processes but may need things like JIT packaging, manifest generation, manifest conditioning, or dynamic ad replacement. This may be initiated by a customer request, and ties in with using the Cloud for delivery of the service to the viewer.

How can a transcoding service take advantage of being virtualized in a cloud-based infrastructure to handle a large scale of content workflows? The benefits of being virtualized are multiple: 1) a service instance can be created or destroyed based upon transcoding demand, 2) a service instance needn't be dependent upon dedicated hardware but general purpose servers, and 3) a service instance can be moved anywhere in the infrastructure, for network bandwidth or storage optimizations. In the near term, it can handle more of the lower priority workflows and reduce the anticipated site-based transcoding equipment needed. In addition, cloud-based resources can be shared with other task-based services to reduce transcoding resources.

What is needed to evolve in this direction? First, a move to using software based transcoding, followed by higher bandwidth for movement of data at both the network level and server level. Lastly, a more defined breakdown of transcoding processes into more granular, operable atomic units.

REALIZING A SOFTWARE TRANSCODER ON GENERAL PURPOSE SERVERS

Software video transcoding requires a significant amount of calculations around the stages of pre-processing, transcoding, and wrapping/packaging. Pre-processing functions are things like decoding, pre-filtering, cropping, scaling, and watermarking. Transcoding processes include transforms (DCT/Integer), quantization, block-based operations, motion search/prediction, and entropy operations. Wrapping/packing functions exist to prepare the content for streaming, as well as chunking/manifest creation for adaptive streaming, or file formats wrapped in MXF. Some of these operations are coefficient based and can scale up to the number of sampled pixels. Others are more blocked-based, repeatable operations. Some other operations don't happen at this scale and require more sequentially-based threads. The types of operations for video encoding are basically a combination of sequential processing, quick paralizeable calculations, and fetching/putting from fast cache memory.

Video transcoding can be implemented in the following architectures with each type having tradeoffs in the areas of programmability (new code), platform rework (redesign based on server hardware type), quality (is it fixed or can it continuously improve?), costs per stream, cloud fit (deployability in Cloud infrastructure), density (rack compactness), and power [See Table 1].

Architecture	Program mability	Platform Rework	Quality	Costs/ stream	Cloud Fit	Density	Power
Hardware Based	Hardware Update	None	Fixed	High	Low	High	Low
Software w/CPU (single/multi-core)	Flexible	Low/High	Improve	Low	Mid/High	Low/Mid	Mid/High
Software w/ CPU & GPU	Flexible	High	Improve	Low	High	High	Mid

Table 1:Video Transcoding Architectures & Tradeoffs

Hardware architectures can be the most efficient and integrated for high-density transcoding -- but because of their permanence, specialization, and high costs, they represent a difficult fit into cloud architectures. The higher costs are attributable to specialization, and an inability to drive costs down through volume. Also, transcoding hardware may not be optimized for doing other task, which is a key to unlocking cloud architecture potential.

A general-purpose server that is suitable for multiple tasks, including transcoding, is a stronger option for distributed infrastructures with large volume scalability.

A key component of its effectiveness is the performance of a single CPU. A single CPU is capable of running 1-2 threads, using time division multiplexing. Each thread can run a single set of sequential instructions. The faster the performance of the CPU, the more threads and more complex set of instructions can be used. The limitation on the performance of a single CPU is due to an upper bound on clock frequency, which in turn stems from silicon die limitations.

An alternative is to implement a multi-core architecture. This allows for multiple processors to be put on a single silicon die. When designed properly (optimizing sequential threads across each core), this can lead to higher performance at lower processor speeds, with improved power management and cooling. Early multicore architectures involved 2 cores, but have been evolving to 2/4/8/16/32 cores. For processing purposes, performance optimization may not be directly proportional to number cores, because each individual process needs to be independent but sequentially assembled. An additional factor to consider is the amount of code rework needed to optimize over different types of multi-core architectures, which can vary over type as well as number [6]. Some experiments of multi-core architectures in HEVC studies have seen performance speed-ups of 25 times that of a single core architecture [7].

Pure software encodes can run on this platform, but do not always take full advantage of the multicore architecture -- even though lower level toolkits may provide ways of taking some advantage of this. Implementation can be further accelerated through the use of Graphical Processing Units (GPUs) and fast access memory. GPUs became popular for the computer gaming industry as specialized PC cards used in creating graphics operations, such as textures and shading. GPUs harness a parallel throughput architecture that is suitable for processing many concurrent threads slowly, and can be suitable for some transcoding operations of a highly scalable nature -- such as coefficient level/ pixel-based operations or block-based operations. GPUs need to have a full input pipeline to be most efficient, and this is where fast access to memory is needed, with optimizations in the code to allow for large block fetching. GPUs can exist as a separate chip on a board (e.g. GT 60xx), in which can handle large volumes but also need to be tied into larger bandwidth and faster memory access that is off the die. GPUs and memory can also coexist on the die, but process less volume and memory while optimizing data management between CPU and on-board GPUs. Originally, GPUs were an afterthought on the die for basic computer graphics purposes. If used for video transcoding, the encoding time speeds up, but the picture quality could suffer.

As chip architecture developed, more room became available on the die (to accommodate I/O demands), which allowed the inclusion of more and better GPUs, memory, and other specialized processors, to address the growing importance of customer-facing media applications [8,9]. This has greatly improved the performance of GPUs on the die -- in some cases up to 75 times over the span of 4-5 years [9] . The popularity of this architecture is evident, with 9 out of 10 PCs already shipping CPU and GPU on the same piece of silicon. [8]

Today, specialized processors are built on the die or GPU chip that is dedicated to perform minute operations involved in video coding. This aims to improve quality while increasing processing speed. GPU/CPU architectures designed to share more memory can further increase performance. There are several new specialized media structures used to help accelerate video encoding [such as Quicksync (Intel – Haswell), and VCE logic (Nvidia Radeon)]. These use video middleware toolkits such as Handbrake and OpenCL as efficient access layers to the hardware. As GPU performance improves [See Fig. 1], it increasingly appears that the use of the evolving general-purpose servers that have these capabilities will be an ideal architecture for transcoding operations. [6]

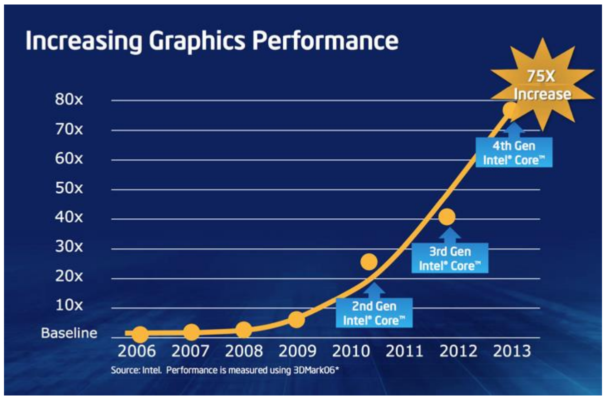

Figure 1: Evolution of Chip Graphics Performance [9]

Software video encoding can be realizable on server architectures, but ultimately needs to be optimized for the multicore architecture that can be accelerated through GPUs, cache memory, and specialized processors using video conversion toolkits. Again, the three components needed for the cloud are computing , networking, and storage [See Fig. 2] -- at a macro and micro level. A multicore architecture with mixed CPUs/GPUs integrates well with this concept [See Fig. 3]. Building servers for this should consider a combination of architecture & capabilities, BOM costs, and data center operational costs.

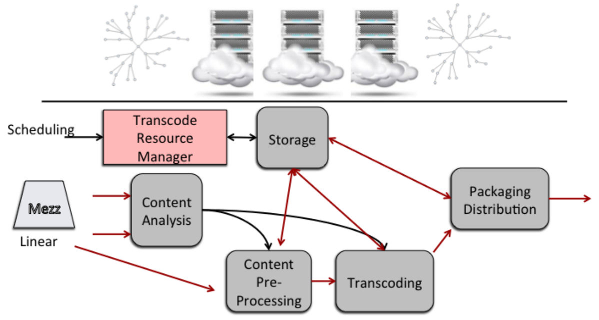

Figure 2: Transcoding Workflows using the Cloud

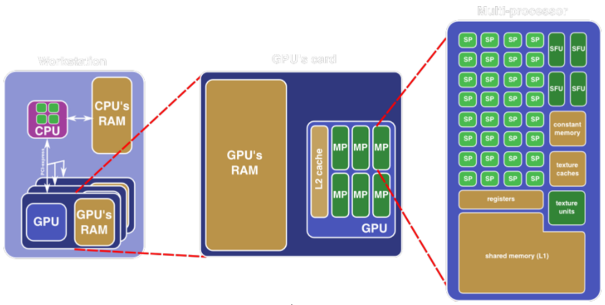

Figure 3: Mixed CPU/GPU architecture [11]

TRADEOFFS OF QUALITY, DENSITY, AND PERFORMANCE

Bringing up a transcoding instance on a general-purpose server allows the flexibility of performing tasks in more than one way. Different approaches may be optimized to converse resources, speed-up transcodes, or improve quality. The advantage of this over dedicated hardware is to be able to look on a larger scale (beyond just a box) to assign and adjust resources (often across different locales) as needed to produce an acceptable output volume.

In dedicated transcoding hardware, some algorithms may be fixed to avoid the complexity of putting multiple approaches in the hardware. An example of this is motion search algorithms. This can affect the accuracy and number of motion vectors generated, which can in turn affect the size of predicted pictures. In dedicated hardware, motion search approaches and range may be limited. Under certain circumstances, using an alternative approach may actually improve picture quality. This opportunity (and others such as dynamic encoding, dithering, and edge enhancement) can now be available using this type of architecture. Another example is the ability to assign cache memory resourced for a single instance. Expanding the amount of on-cache memory and allowing large block fetch movements can optimize the interaction between CPUs and GPUs, but optimization can differ depending on the size of the picture [1].

Lastly, in a software/general server-based environment, more than one instance can now be invoked to handle a task. A "split and stitch" process could use the multiple instances to split the content up, process each section in parallel, then reassemble the compressed stream [See Fig. 4]. This can be helpful when dealing with near time or JIT content workflows.

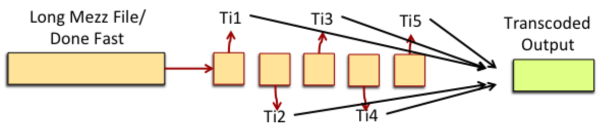

Figure 4: Split & Stitch Workflow

With a more actively managed orchestration of a workflow, there exist many yet-to-be-explored advantages that may be useful for content transcoding workflows using software on general-purpose servers. It does, however, require changes in traditional content workflows.

DESIGNING CONTENT WORKFLOWS FOR THE CLOUD

A. Adjusting Traditional Content Workflows

A traditional approach in transcoding is to make use of "hot folders" to process content in each step of the workflow [See Fig. 5]. Basically, content and/or metadata is put into a storage folder. The transcoding operation would then see the content and start processing it. At the end of the task, it places the content in an output folder, which then may act as an input folder for another task.

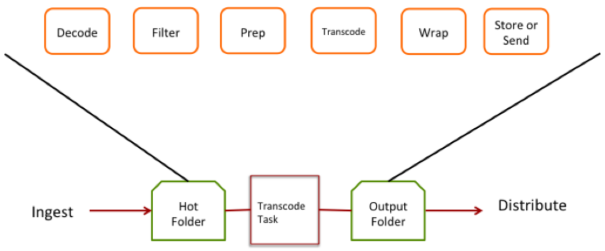

Figure 5: Hot Folder Workflow

There are several items to be aware of when using hot folders:

- Content needs to be processed in the same way (issues arise when the content gets placed in the wrong folder)
- Content needs to be complete before a task is started
- It is difficult to indicate priority, and thus harder to adjust the workflow because of priority
- Resources can idle, due to empty folders, making it all the more difficult to share resources

An alternative approach is to associate tasks with the content as a "job". In this sense, the content is acted upon, rather than placed in a folder. The transcoding instance is not designed to do a dedicated set of tasks, but instead looks to see what task needs to happen on a piece content at the moment. This is similar to the use of a general-purpose server rather than a dedicated hardware device. With an associated ID and metadata (such as due date, aspect ratio, etc.) , the concept of a job can be a powerful concept and can enable:

- Workflows that adjust to content
- Reduction of storage

- Scheduling of resources even before content arrives
- Prioritizing workflow to either handle voluminous or bursty traffic
- Reducing transcoding resources when not needed and applying them to other cloud tasks

Additionally, another change that needs to be made to improve adaptability to cloud and data is to break down tasks into more atomic units. Instead of just transcode, it can be broken down further into operations such as:

- Job verification
- Schedule
- Unwrap
- Verification (syntax, file size, semantic)
- Baseband processing (e.g. Cropping, Scaling, prefiltering)
- Decode
- Watermark, Fingerprint
- Encode
- Delete
- Wrapping, Distribution

Each of these tasks has their own combination of resources and can vary in complexity and time-to-finish. These three factors may not be deterministic, but can be bounded. These tasks can be combined in different manners to adjust the workflow to the types of source content, types of job, available resources, and time constraints. Creating smaller task modules and using a job workflow will allow better processing in the cloud in a resource optimized way.

B. Orchestrating Workflows Using A Transcoding Resource Manager

In order for a virtualized software transcoding instance to operate in a cloud infrastructure, the following are required: A balance of resources, network bandwidth demands, and timely output volume of assets. In order to keep these three aspects in balance, while exploiting the advantages of non-dedicated, general-purpose resources, the concept of a transcoder resource manager needs to exist [See Fig. 6]. This is more than just a workflow manager, because it balances server resources, storage, and bandwidth in a scheduled manner.

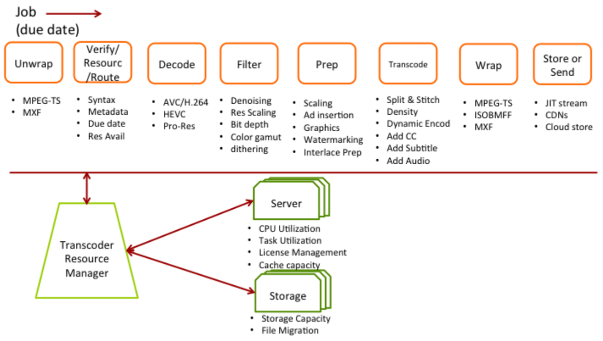

Figure 6: Workflow Using a Transcoder Resource Manager

The transcoder resource manager determines when and where an instance(s) will be created to address a job and when that instance will be destroyed/taken down. It will consider what server(s) to use and where to locate storage. It will schedule jobs based upon priority, type of content workflow and estimated time to completion (JIT, Near-Time, Catalog, Assembly). It will also ensure that jobs will be done agnostic to any types of equipment failures.

Some of the decisions that the transcoder resource manager could make can be exemplified by a mezzanine workflow to create a UHD-1 output as well as a set of MBR (Multi-Bitrate) streams:

- Determine how instances can exist at site(s) closest to the content mezzanine
- Verify source characteristics through metadata and syntax validation
- Create an instance to convert color gamut from BT 2020 to Rec 709
- Convert 10-bit input into an 8-bit version

- Split content into 5 sections and create 30 instances to process each section, for both 4K and MBR outputs
- Determine task assignment on servers using CPU utilization as a guideline
- Record linear programming
- Index file generation
- Manifest generation
- CDN distribution
- Destroy task instances and copies

OPERATIONAL IMPACTS

The basic building blocks for cloud infrastructure (computing, networking, storage) need to support the maximum performance needs of individual tasks for virtualized software transcoding. Cloud-based software and virtualized transcoding platforms will also impose unique operational requirements for the service providers.

These may include: (1) Capacity planning of compute and networking resources to enable simultaneous transcoding streams (dependent on types of services - linear, VOD, cDVR) and mode of transcoding (real-time, just-in-time); (2) Distribution topology of transcoding resources to national or regional data centers, based on attributes of source origination, network connectivity, national or local channels, and content delivery network; (3) Remote operational monitoring of software transcoding configuration, usage, performance, and availability; (4) Redundancy with automatic failover; and, (5) Transcoding software upgrade strategy.

CONCLUSION

Shifting to cloud-based transcoding services will help address the ever-increasing demand for video services with flexible resources. As algorithms continue to be optimized for multi-core architectures, and as networking bandwidth increases, and storage gets more accessible, cloud-based transcoding services can scale to meet the demand without scaling the costs in the same manner. Modification in the content workflows that are needed are: 1) implementing a job-based priority workflow, 2) breaking down the transcoding processes into smaller, atomic tasks, and 3) creating a transcoding resource manager to coordinate tasks and distributed equipment resources.

This approach can open up vast new types of business and operational models, where transcoding resources can be leased rather than purchased, and used to off-load peak transcoding demands. Virtualized software transcoders can create a flexible transcoding process that will be needed as the types and volumes of content for linear, VoD, and cDVR services grow.

REFERENCES

[1] F. Lee, “implementing H.264 encoding algorithms”, Embedded Systems, March 2006, pp.34-37.

[2] J. Baumgartner, “Comcast’s Unsung Hero: The Network”, Multichannel News, Feb. 3rd, 2014, p.2.

[3] M.Ferrell, “Why Comcast Is Winning Subscriber War (for Now)”, Multichannel News, Feb 3., 2014, p.29

[4] Conversations with Austin Vrbas, Brian Taft, Chris Cunningham, Feb 2012.

[5] www.cablelabs.com/specifications: MD-SP-ADI1.1, MD-SP-CONTENTv3.0, MD-SP-VOD-CONTENT1.1.

[6] Cyril Kowaliski, A look at hardware video transcoding on the PC: Performance and image quality with black boxes and Open CL. Tech report,., www.techreport.com, July 30th , 2012.

[7] M. Horowitz, F. Kossentini, and H. Tmar, " Informal Subjective Video Quality Comparison Between the eBrisk-UHD HEVC and x264 AVC Encoders", JCTVC-P0158, 16th Meeting of JCT, San Jose , Ca Jan 9-17, 2014.

[8] I. Cutress, and R. Garg, " AMD Kaveri Review: A8-7600 and A10-7850K Tested" AnandTech, Anandtech.com, Jan. 14, 2014

[9] A. L. Shimpi, "Intel Iris Pro 5200 Graphics Review: Core i7-4950HQ Tested" Anandtech, Anandtech.com, June 1, 2013.

[10] www.aurelian.plyer.fr/phd/gpu-programming

[11] www.scte.org/standards: ANSI/SCTE 128-1 2013, ANSI/SCTE 128-2 2013, ANSI/SCTE 54 2009, ANSI/SCTE 35 2013a

[12] tvbythenumbers.zap2it.com /2013/08/01/list-of-2013-fall-tv-show-premires-dates/195183