code{4}lib JOURNAL

Issue 22, 2013-10-14 ISSN 1940-5758

For Video Streaming/Delivery: Is HTML5 the Real Fix?

The general movement towards streaming or playing videos on the web has grown exponentially in the last decade. The combination of new streaming technologies and faster Internet connections continue to provide enhanced and robust user experience for video content. For many organizations, adding videos on their websites has transitioned from a "cool" feature to a mission critical service. Some of the benefits in putting videos online include: to engage and convert visitors, to raise awareness or drive interest, to share inspirational stories or recent unique events, etc. Along with the growth in the use and need for video content on the web; delivering videos online also remains a messy activity for developers and web teams. Examples of existing challenges include creating more accessible videos with captions and delivering content (using adaptive streaming) for the diverse range of mobile and tablet devices. In this article, we report on the decision-making and early results in using the Kaltura video platform in two popular library platforms: CONTENTdm and DSpace.

by Elías Tzoc and John Millard

Introduction

The video market share on the web is such that some tech companies such as Cisco are now predicting that online video will surpass social media in popularity by 2017. For universities, creating and delivering video content has also gained attention in the last few years, especially for those moving into hybrid (online and face-to-face) types of classes and e-learning initiatives. As for library content, the digitization of collections originally stored on reel-to-reel, VHS or DVD along with the creation of new video content like oral histories, has created new opportunities to deliver video content on the web.

Flash was once the preferred video container file format used over the Internet; however, the new version of HTML5 offers new possibilities for managing and delivering video content. Web developers can create customized video players and create more accessible captioned videos, end-users can stop installing third-party browser plugins including Flash. Flash is particularly problematic due to generally high CPU usage for video playback and no support for iOS devices. A recent study on *Video Performance Analysis between Flash Video and HTML 5 Video* suggests that a combination of both technologies may still work better [1]. In the next sections, we report on the decision-making and early results in using the Kaltura video platform, the workflow and tools used in migrating and re-formatting several video collections, the integration of a Flash with HTML5 fallback in popular library platforms, and an early evaluation of the time/cost of creating videos with captions.

Evaluation of video delivery options

In late 2012, Miami University Libraries faced a perfect storm of technical developments that forced the re-evaluation of our delivery of media-based digital collections. On the one hand, we were forced to migrate our content due to hardware failures locally and decommissioning of a centrally managed statewide repository. On the other, we had the opportunity arising from tremendous gains in network bandwidth both on campus and in "last mile" providers and of course the continued transition to mobile devices in a post-pc landscape. Our previous solution for delivering media involved a local QuickTime Streaming server for some content and a Flash-based progressive download system delivering other content from the Statewide repository running DSpace.

As we began the process of evaluating options, we developed a short list of required capabilities that addressed perceived shortcomings in our previous configuration.

1. **Standards compliant and non-proprietary:** whatever method we chose had to provide high quality video and audio and be standards compliant; something based on open source technologies would be a highly sought after plus.
2. **Not flashy:** the selected solution needed to not rely on Flash or installation of a plugin. Flash excludes iOS devices and plugins would most likely be platform specific.
3. **Keep it simple:** be simple to implement and maintain; like many organizations, our staff is small and we have multiple

competing obligations; thus, the solution needed to “just work” for both novice and experienced staff.

4. **Embeddable:** ability to embed content into multiple existing or future digital delivery platforms; right now we use CONTENTdm, DSpace , and Omeka, but that may change tomorrow.

In general, media files can be delivered in one of three ways, via streaming, progressive download, or adaptive bitrate streaming. Each has its purpose and for the most part, regular users will not notice the difference.

Streaming involves delivering the media to the client via a server process using specific streaming protocols (such as RTMP). Video playing begins almost immediately, especially if the video file was encoded at a data rate similar to the effective bandwidth of the target viewer. Streaming video is also often not cached by the client so a local copy of the video is not held in its entirety on the client machine. While it is not impossible for an enterprising person to capture and hold a copy of the stream, it takes more effort than the casual viewer may be willing to take on. To adapt for the slowest common denominator in regard to end-user bandwidth, streaming videos are often encoded at lower quality and data rates.

Progressive download simply delivers a media file via traditional webserver technologies. The file begins playing on the client as soon as enough data has been buffered to provide a smooth uninterrupted viewing experience. Progressive downloaded files are easier to capture since an entire copy of the file is downloaded to the local machine. Also, the quality of the file can be higher simply because a user on a slower connection will just have to wait longer for the viewing to begin. Progressive download, at least in our experience, is also easier to graft into systems that don’t natively support video and audio such as DSpace and CONTENTdm.

Adaptive bitrate streaming is a kind of best of both worlds. As the name implies, adaptive bitrate is a streaming technology and generally requires a dedicated streaming server. In this case, media files are transcoded into multiple bitrates with the appropriate streaming being delivered to the user based on their available bandwidth. Adaptive streaming servers can also dynamically change the bitrate as network conditions dictate.

Early in our evaluation, we considered adopting whatever platform was being supported by our local IT organization for E-Learning and copyright compliant streaming. However, until recently, that service had been tailored for restricted access viewing of commercial content and as such was limited to on-campus use by authenticated university users. The Libraries needed a platform that allowed open, unregistered access to copyright free materials. The environment changed this year with the adoption by the university of the hosted Kaltura Video Platform service. The Kaltura hosted service is built on the Kaltura Open Source Online Video platform. As a service, Kaltura met all our requirements. It is an adaptive bitrate solution that automatically transcodes uploaded video to optimized output formats. The video player provides automatic device detection and delivers either an HTML5 or Flash player depending on capabilities. Kaltura claims support for Android, Blackberry, and iOS devices. Published pricing starts at $750/month so it’s definitely in the range of enterprise level as far as cost. However, the method we describe works with similar services like Vimeo which can be hosted more cheaply for smaller scale projects. In our case, we piggy-backed on an institutional subscription.

Migrating and re-formatting video collections

Much of our open access video content was stored on analog media. One of the biggest challenges was creating digital media from obsolete and rapidly disappearing technologies like reel-to-reel tape, BetaMax, VHS, Digital Audio Tape, audio-cassette, and even some radio NAB cartridges. With help from a now defunct local radio station and other sources, we’ve acquired machines to play back most of the media we encounter, including broadcast reel-to-reel, DAT, Sony MiniDISC and Betamax.

We have assembled a Rube Goldberg-esque system of analog to digital convertors and media adapters to connect our analog devices to a Mac Pro for processing. Video devices connect through a FireWire DA converter or a digital MiniDV deck. Audio devices are connected through a Roland USB audio interface. We use the USB device to filter processor and hard drive noise and to allow more connection types. The Roland, for example, can accommodate 3pin XLR and ¼ inch inputs commonly found on professional audio equipment. We use Audacity for audio capture and editing and Final Cut Pro X for video capture and editing. We also employ HandBrake and MPEG Streamclip to perform format conversions and transcoding, although Final Cut’s Compressor can also be effective, especially for large batch processes. Adopting Kaltura has allowed us to eliminate a time intensive step in our workflow, mainly the transcoding of video for streaming. Kaltura takes whatever format we have (within reason) and does the transcoding on their cloud platform.

We have also recently begun capturing 16mm sound movies using a custom built unit from the Texas-based Moviestuff LLC. Their Sniper HD telecine is composed of a modified 16mm film projector with optical sound. Video is captured by an

HD camera and sent along with analog audio to a BlackMagic capture card in a custom built Windows PC. The resulting video quality is quite good considering that most of the films we have are approaching 50 years old.

Integration of video players in library platforms

As mentioned in section two in this article, one of the four requirements identified as important for our new video solution was the ability to embed content into multiple digital platforms with automatic device detection and fallback. One of the key features we found with Kaltura was the ability to support both Flash and HTML5 in the video player, satisfying our need to provide video to devices that do not support Flash. Although we did not use it, Kaltura also provides an HTML5 Video Library and a Video Player API to allow for the development of custom players. We were satisfied with the included player and used the embed settings available on the Kaltura control panel. Figure 1 indicates the options we selected. Once confirmed, Kaltura generates the appropriate embed code.

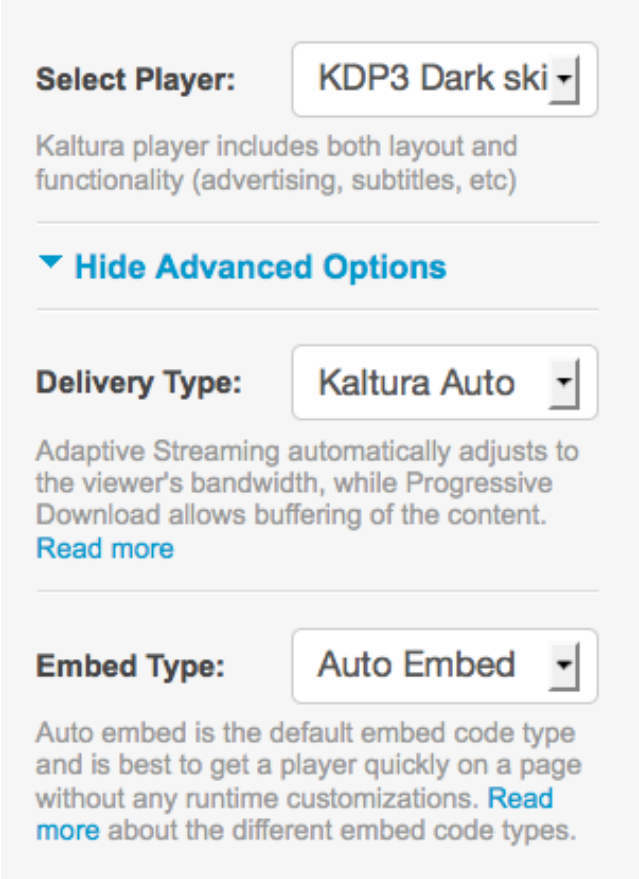

Figure 1. Kaltura embedding options.

A typical Kaltura embed generated code will look something like the next following lines:

```
<div id="kaltura_player_1376588661" style="width: 400px; height: 333px;" itemprop="video" item
<span itemprop="name" content="{$video-title;}"></span>
</div>
<script src="http://cdnapi.kaltura.com/p/886232/sp/88623200/embedIframeJs/uiconf_id/16115322/p
</script>
```

Our three target systems were DSpace, CONTENTdm, and Omeka. Each of these provides the ability to serve up video files, but do not provide native streaming.

In the OhioLINK solution, developers were successful in simulating a streaming service by embedding a flash video player that served up MP4 video bitstreams as progressive downloads. We considered re-implementing that player in our local install of DSpace and perhaps upgrading it to an HTML5 player. A similar technique could have been adapted to work with other platforms as well. The problem we saw was that we would have had to continue to contend with the other limitations of progressive video with that solution. We wanted something more robust that could, among other things, provide an optimized viewing experience at multiple target bandwidths. Kaltura provides this functionality and a relatively

straightforward method to embed video in a foreign system.

Examples of how this code was integrated in our three digital platforms are below:

CONTENTdm provides a way to add records with a URL as the primary item; this alternative allows the system to display webpages or links to streaming media servers. In this case, we just needed to add URLs for every video item with a valid Kaltura video ID. Then, the next step is to add a few lines of PHP and HTML code in the loader.php file or basic_view file located at:

/Content6/public_html/ui/cdm/default/collection/default/viewers/showLink

/Content4/docs/collection_alias/includes/basic_view.php

```
<?php
//echo retrieving kaltura video ID
if (strpos($parent->itemLinkUrl,'http://www.kaltura.com/') !== false) {
   $kaltura-video-id = $parent->itemLinkUrl;
   $kaltura-video-id = substr($kaltura-video-id, 23, 10);
?>
<!-- BEGIN Kaltura video code -->
<div id="kaltura_player_1376588661" ... itemtype="http://schema.org/VideoObject">
<span itemprop="name" content="<?php echo $video-title;?>"></span>
</div>
<script src="http://cdnapi.kaltura.com/ ... &entry_id=<?php echo $kaltura-video-id;?> ... ">
</script>
<!-- END Kaltura video code -->
```

Example: The Freedom Summer Digital Collection available at: http://digital.lib.muohio.edu/fs-videos/

DSpace can handle video files as a primary bitstream within the digital object and simply sends the bitstream to the user as a download. OhioLINK's video player provides an inline viewer by locating the bitstream's physical location in the DSpace assetstore and feeding that asset back to the flash player. That approach requires a fair amount of hacking with the METS metadata that DSpace keeps internally. Alternatively, integrating the Kaltura hosted video was much simpler, requiring the addition of an extra metadata field in DSpace with the Kaltura video ID and using that as a reference to embed the Kaltura player. The final step was to add some XSL and HTML code in the theme.xsl file located at: /webapps/xmlui/themes/your-theme/lib/xsl/aspect/artifactbrowser/

```
<!--  Adding mp4 call in itemSummaryView-DIM -->
 <xsl:template name="itemSummaryView-DIM">
    <xsl:apply-templates select="./mets:fileSec/mets:fileGrp[@USE='CONTENT']/mets:file[@MIMET
</xsl:template>

<!-- BEGIN Kaltura video code -->
<xsl:template match="mets:fileGrp[@USE='CONTENT']/mets:file[@MIMETYPE='video/mp4']" mode="fla
<xsl:variable name="v1" select="(dim:field[@element='relation' and @qualifier='isversionof'])
<div id="kaltura_player_1376588661" ... itemtype="http://schema.org/VideoObject">
<span itemprop="name" content="{$video-title;}"></span>
</div>
<script src="http://cdnapi.kaltura.com/ ... &entry_id={$kaltura-video-id} ... ">
</script>
<!-- END Kaltura video code -->
```

Example: The CAWC Lecture Series Digital Archive available at: http://digital.lib.muohio.edu/cawc/

Omeka can also support items with URLs. The trick is to create items and add a URL in one of the fields. We chose the IDENTIFIER field but other fields will work as well, especially if your metadata profile requires a different use of identifier such as for a DOI or accession number. With the Kaltura video ID in the Identifier field, we then added the viewer code in the show.php file located at: /themes/your-theme/items

```
<?php
// get Kaltura video ID
$kaltura-video-id = metadata('item', array('Dublin Core', 'Identifier'));
?>
<!-- BEGIN Kaltura video code -->
<div id="kaltura_player_1376588661" ... itemtype="http://schema.org/VideoObject">
<span itemprop="name" content="<?php echo $video-title;?>"></span>
</div>
```

```
<script src="http://cdnapi.kaltura.com/ ... &entry_id=<?php echo $kaltura-video-id;?> ... ">
</script>
<!-- END Kaltura video code -->
```

Our current example will be launched later this year.

One of the greatest benefits of using the Video Player SDK is that we only needed to embed 4-5 lines of code in our local library platforms. Using the HTML5 Video Library self-hosted option would require an explicit list and access for all the files available for every video; an example for two different bitrates and two iOS devices is below:

```
<video title="{$video-title}"
    data-entryid="{$kaltura-video-id}"
    data-durationhint="3271"
    poster="http://cdnsecakmi.kaltura.com/p/886232/sp/88623200/thumbnail/entry_id/{$kaltura-v
    controls="controls"
    style="width: 640px;">

<source data-bitrate="2832"
    data-width="640"
    data-height="480"
    src="http://cdnbakmi.kaltura.com/p/886232/sp/88623200/playManifest/entryId/{$kaltura-vide
    data-flavorid="iPad"
    type="video/h264">

<source data-bitrate="2408"
    data-width="480"
    data-height="360"
    src="http://cdnbakmi.kaltura.com/p/886232/sp/88623200/playManifest/entryId/{$kaltura-vide
    data-flavorid="iPhone"
    type="video/h264">

<source data-flavorid="iPadNew"
    type="application/vnd.apple.mpegurl"
    src="http://cdnbakmi.kaltura.com/p/886232/sp/88623200/playManifest/entryId/{$kaltura-vide

<source data-flavorid="iPhoneNew"
    type="application/vnd.apple.mpegurl"
    src="http://cdnbakmi.kaltura.com/p/886232/sp/88623200/playManifest/entryId/{$kaltura-vide

</video>
```

Cost of creating videos with captioning

A great introduction to this topic along with some quick facts and stats on the importance of video captioning was given by the moderator of a recent panel "Strategies for Deploying Accessible Video Captioning" that was part of the 2013 Streaming Media East conference. Creating captioned videos can be time-consuming but there are legitimate reasons to do so. Four reasons and benefits include: a) complying with organizational or legislated accessibility requirements b) to provide a good alternative to videos with poor quality audio, c) to make videos text searchable to search engines, and d) to increase comprehension. From a pragmatic perspective, our campus recently mandated that all educational video be provided with accessible captions, especially in online learning contexts. In response, in Summer 2013, we began an effort to investigate and benchmark captioning for our video content.

The two main activities in creating captions are: transcribing the video and synchronizing the text and video. The transcription process remains the most labor intensive activity. Although there are now some software tools that claim to auto-transcribe speech to text, the results from these software tools are still not that accurate. Therefore, outsourcing the work to video transcription services can be a viable solution; organizations such as Verbal Ink, Speechpad, CastingWords, or SpeakerText charge between $1 and $2.50 per minute with a delivery service in 24-48 hours and with options for time-stamped files. Of course, another solution for transcribing videos is to do it in-house. In our case, back in 2008-2009, a team of two librarians and student assistants led the video transcription work for two video collections. The average time for transcription was between 5-7 times the runtime of each video.

One of the challenges we found with our solution is that the workflow did not include the synchronization of text and video. Users have been able to view and download the transcript in PDF files, but with the new demand for captioned videos, we would like to add captions to as many videos as possible. We are currently testing the MovieCaptioner software, which

allows us to import the transcripts as text files; the export options provides a set of file formats including SRT, SCC, SUB, SMIL, TT, and many others. Kaltura supports SRT/DFXP files for video captioning. Another example of a university creating captioned videos is the Stanford Captioning project; they outsource the transcription process and the captioning team does the synchronization part.

As for the actual labor and cost of creating captioned videos, our current estimate indicates that for 1 hour of video, we would need an average of 6 hours for transcribing and an additional 3 hours for synchronizing; in other words, for every minute of video, we would need to pay 9 minutes of work. The actual in-house cost would depend on how much of the work is done by whom -e.g. undergraduate students, graduate students, library staff, or professional librarians. But assuming that 75% of the work is done by a student and 25% of quality control and editing is done by a librarian, it should be fair to assume that every hour of work would cost a library about $12; therefore, creating video captions for a 1 hour video -which would require about 9 hours of work- can cost about $108. For comparison, one of the transcription services quoted a cost for captioning a 1 hour video to be about $120 with no guaranteed turnaround time. Rush options such as “next day” or “2-3 business days” were more expensive. The two prices are close enough that it is hard to suggest or choose one over the other.

Lessons learned

When dealing with videos, delivering the best possible video experience is the ultimate goal for every web team. We all have seen good evidence that several HTML5 compliant video players (VideoJS, HTML5Video, JW Player, etc.) are becoming excellent alternatives when a fallback for non-flash support is required. Perhaps the answer for the title/question of this article “is HTML5 the real fix for videos?” remains unanswered; but the hybrid solutions (Flash and HTML5) can be good examples of how to deliver video content, regardless of which device or browser users are using. After all, users do not necessarily care about how their videos are being delivered or processed. One surprising observation we made as a result of this work was that there was a relative dearth of documented work on the topic of video streaming in the library community. As for the Kaltura platform, it was definitely good to learn that it is also currently used by the Internet Archive to manage its digital video library to seamlessly support HTML5 and Flash. Last but not least, for those interested in learning and contributing to the Kaltura community supported project (hey code4libers), check the free community-supported site. If you are looking to meet other open source video developers and share ideas, then Kaltura.org may be the place for you.

Endnotes

[1] Ilias, I. S. H. C., Munisamy, S. B., & Rahman, N. A. A. (2013). A study of video performance analysis between Flash video and HTML 5 video. In Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication (p. 30). ACM. Available at: http://dl.acm.org/citation.cfm?id=2448586 Accessed on September 15, 2013.

About the Authors

Elías Tzoc is the Digital Initiatives Librarian at Miami University Libraries. He assists the head of the Center for Digital Scholarship to provide Miami University scholars with the facilities, services, and expertise to support the creation and use of digital scholarships. His recent work includes: co-developing a framework for creating and publishing open access e-books; prototyping and implementing web interfaces for digital projects using CONTENTdm, DSpace, OJS, and Omeka; researching and publishing on technical issues and open source applications for libraries; researching for new access points and mobile apps for digital library programs using the jQuery mobile framework; and developing/publishing web plug-ins using PHP, HTML5, XSLT, CSS, and jQuery.

John Millard is the head of the Miami University Libraries Center for Digital Scholarship. A graduate of the University of Illinois at Urbana-Champaign with a Master of Science in Library and Information Science. His research interests and activities focus on developing digital library systems to serve distinct user communities. He is a former NASA and USGS funded investigator working for the OhioView project, a congressionally sponsored project to further the use of U.S. civilian satellite data by the public. At Miami University, he also serves as an adjunct faculty member in the interactive media studies program, teaching a course on Information Studies in the Digital Age and on the faculty of the Learning Technologies Summer Institute.

Subscribe to comments: For this article | For all articles