FIVE TECHNIQUES TO SAVE ENCODING AND STREAMING COSTS
# CONTENTS

**SAVING ON ENCODING: ADJUST ENCODING CONFIGURATION TO INCREASE CAPACITY** .......................................................................................................................................................... 3  
  x264 Presets - Increase Capacity by 73% ........................................................................................................................................ 3  
  x265 Presets - Triple Your Capacity and Improve Quality ........................................................................................................... 5  
  Cut Reference Frames to Boost Capacity with Minimal Quality Loss .......................................................................................... 9  
  Action Steps: .............................................................................................................................................................................. 10  
  Where to Go from Here ............................................................................................................................................................. 11  

**SAVING ON ENCODING AND STREAMING: DEPLOY CAPPED CRF** ........................................................................................................... 12  
  Bandwidth Savings, Increase QoS, or Both? .................................................................................................................................. 12  
  What is Capped CRF ................................................................................................................................................................. 13  
  Bitrate Control Concerns ............................................................................................................................................................. 14  
  Careful With Screencams ............................................................................................................................................................ 15  
  Deploying Capped CRF ............................................................................................................................................................... 17  
  Capped CRF in FFmpeg ............................................................................................................................................................... 18  
  What About Duplicate Resolutions ...................................................................................................................................... 18  
  Where to Go From Here ............................................................................................................................................................. 19  

**SAVING ON ENCODING AND DELIVERY: DYNAMIC PACKAGING** ........................................................................................................... 20  
  About Dynamic Packaging ........................................................................................................................................................... 20  
  Getting Dynamic Packaging ......................................................................................................................................................... 21  
  What About CMAF ................................................................................................................................................................. 22  
  Where Dynamic Delivery is Contraindicated ............................................................................................................................. 23  
  Where to Go From Here ............................................................................................................................................................. 23  

**SAVE ON ENCODING COSTS: CUT CLOUD ENCODING CHARGES** ........................................................................................................... 24  
  Transmux Fees ........................................................................................................................................................................... 26  
  When the Cost of Switching is High? ......................................................................................................................................... 26  
  Otherwise, Consider other Encoding Shops ............................................................................................................................ 26  
  Where to Go From Here ............................................................................................................................................................. 27  

**SAVING STREAMING COSTS: ADDING A NEW CODEC** ....................................................................................................................... 28  
  Introduction ................................................................................................................................................................................. 28  
  Encoding Cost/Per Hour ............................................................................................................................................................. 28  
  Bandwidth Savings Per Hour ...................................................................................................................................................... 29  
  How Much Savings? ................................................................................................................................................................. 29  
  Putting it All Together ............................................................................................................................................................... 31  
  Final Thoughts .......................................................................................................................................................................... 31  
  Where to Go From Here ............................................................................................................................................................. 32
SAVING ON ENCODING:
ADJUST ENCODING CONFIGURATION TO INCREASE CAPACITY

This article details how to cut x264 encoding costs by 73% without noticeable quality loss and triple your x265 capacity while actually improving video quality.

A key focus of my book Video Encoding by the Numbers was to isolate the qualitative impact of critical encoding decisions, particularly when they significantly impact encoding time. For example, if a configuration improves quality by 1%, it likely wouldn't be visible and wouldn't improve the viewer's quality of experience. If this configuration option boosts encoding time by a factor of ten, it's probably not worth it.

Clearly, configuration options like this impact cost in some instances and don't in others. For example, if you're encoding on-premise and you're approaching full capacity, you may need to purchase another encoder. If you can accelerate your existing encoder without significantly degrading quality, you could save significant CAPEX.

If you're encoding in the cloud with a service like Hybrik, where you pay a fixed monthly fee and pay for your own compute hours, cutting encoding time means cutting encoding costs. Ditto for pricing plans like encoding.com's reserved cloud, where you rent the encoder for a monthly fee and process as much video as possible during that month.

Conversely, if you're on a per-minute or per-gigabyte plan in the cloud, cutting encoding time won't impact costs, and it makes sense to optimize quality to the max, irrespective of encoding.

Which configuration options should you check? There are two parameters with huge encoding time swings but only minor changes in encoding time or quality. These are presets and reference frames.

**x264 Presets – Increase Capacity by 73%**

For presets, I'll address x264 and x265, since these are the codecs that I work with in FFmpeg. Both codecs offer ten presets with the same names, from Ultrafast to Placebo, which provide video producers with a simple mechanism for trading off quality against encoding time. If your encoder uses the x264 and x265 codecs, which most do, you should be able to select from among these presets. Note that the default is Medium, so if you don't specify otherwise in your encoder, that's the preset you'll use.

In my Numbers book, I tested all encoding parameters with eight different files, including real-world videos, animations, movie clips, and screencams to account for variations in file types. For the chart shown in Figure 1, I encoded each clip ten times using identical configurations save the x264 preset. Then I measured the quality of each clip using Netflix's VMAF metric calculated with the Moscow State University Video Quality Measurement Tool (VQMT). For perspective, values above 93 typically don't show any artifacts, and a difference of six points constitutes a just noticeable difference (JND) that viewers will likely observe.
All videos but Haunted cross 93 when encoding with the Faster preset, after which VMAF increases on average from 94.52 to 95.62, or about 1.1 VMAF points, well under the 6-point JND threshold. This means that for most videos, the Faster preset delivers sufficient quality to eliminate artifacts and that the additional quality delivered by the higher-quality presets likely wouldn’t be noticed by most viewers.

So, if you’re currently using the Medium preset and switch to Faster, you increase capacity by 73% while dropping the average VMAF score from 95.11 to 94.52, a change that would be imperceptible to the vast majority of your viewers.

Table 1 shows the average encoding time in seconds for all test clips using all ten presets on my HP Z840 workstation. Line 2 shows the impact on encoding capacity if you switch from the default Medium preset to another. So, if you’re currently using the Medium preset and switch to Faster, you increase capacity by 73% while dropping the average VMAF score from 95.11 to 94.52, a change that would be imperceptible to the vast majority of your viewers.

<table>
<thead>
<tr>
<th></th>
<th>Ultrafast</th>
<th>Superfast</th>
<th>Veryfast</th>
<th>Faster</th>
<th>Fast</th>
<th>Medium</th>
<th>Slow</th>
<th>Slower</th>
<th>Veryslow</th>
<th>Placebo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average encoding time</td>
<td>11</td>
<td>12</td>
<td>15</td>
<td>16</td>
<td>27</td>
<td>28</td>
<td>42</td>
<td>48</td>
<td>66</td>
<td>213</td>
</tr>
<tr>
<td>Capacity from Medium preset</td>
<td>154%</td>
<td>138%</td>
<td>95%</td>
<td>73%</td>
<td>7%</td>
<td>0%</td>
<td>-32%</td>
<td>-41%</td>
<td>-57%</td>
<td>-87%</td>
</tr>
</tbody>
</table>

Table 1. Average encoding time for the x264 presets and impact on capacity of change from Medium preset
If you’re not using the x264 codec, you still can benefit from this type of analysis. For example, Elemental encoders use their own codec, with a numerical switch (1-4 I believe) that trades off quality against encoding time. The MainConcept H.264 and H.265 codecs use values ranging from 1-28 to do the same. Using the analysis above as a template you could easily create your own test suite and find the optimal tradeoff between encoding time and quality. Click here to download the handout from a presentation on HEVC quality where I ran such an analysis for the MainConcept HEVC encoder and Google's VP9 codec.

x265 Presets – Triple Your Capacity and Improve Quality

The HEVC preset situation presents a completely different set of choices. Figure 2 shows the results for the eight test files encoded with the ten presets. Where the x264 presets produce a reasonably steady increase in quality from left to right, the x265 presets achieve a mini-peak at Superfast (average VMAF of 95) that isn't surpassed until the Slow preset (95.43).

Note that I checked my FFmpeg encoding script twice to make sure that I encoded correctly. In addition, in the Numbers book, which presents similar test results using PSNR rather than VMAF, the same dynamic was generally true; that is a peak at Superfast, then a drop in quality and slow increase that didn't surpass Superfast quality until the Medium preset. So, I'm pretty sure the numbers are correct, even though they appear counter-intuitive.

Figure 2. VMAF values by clip and x265 preset

Figure 3 parses the data by clip type, separating animations, real-world videos, and synthetic videos, with Tears of Steel in the real-world category. Interestingly, the synthetic videos show the expected pattern of gradual increases in all presets. With real-world videos, Superfast produces the highest quality through the Slow preset, and with animations, it’s the highest quality through Medium. In all cases, the quality jump between Medium and Slow is alluring, though at about 1 VMAF on average, it would go unnoticed by the vast majority of viewers.
For example, if you change from the Medium x265 preset to the Superfast preset, you pick up 218% of capacity, more than tripling your throughput. For real-world video clips, you'll actually improve the quality of your encoded videos.

Whichever chart you look at, it's hard to justify encoding with the Slower, Veryslow or Placebo presets since the quality stays the same for animation and synthetic videos, and actually drops slightly for real-world clips. The big question is, should you use the Superfast, Medium, or Slow preset? To make that call let's look at encoding time.

**Figure 3. VMAF values by clip type and x265 preset**

Table 2 presents the time element showing the average number of seconds for all eight clips for all ten presets encoded on my HP Z840 workstation. The second line shows the impact on capacity if you change from the default Medium preset. For example, if you change from the Medium to the Superfast preset, you pick up 218% of capacity, more than tripling your throughput. For real-world video clips, you'll actually be improving the quality of your encoded videos.

**Table 2. Average encoding time for the x264 presets and impact on capacity of change from Medium preset**
That said, whenever you use a very low-quality preset you should check for transient quality glitches using a tool like the Result Plot from the Moscow State University VQMT tool (Figure 4). This plots the VMAF score of the Sintel clip encoded using the Medium (in green) and Superfast presets (in orange). The top plot shows the entire clip while the plot on the bottom shows the highlighted region in the top. In this case, that's the orange downward spike about 25% into the clip, which represents the much larger downward spike near the center of the bottom graph. To view the frames at the selected location, you would click the Show frame button on the bottom right, which displays the frames from the source and both encoded videos.

![VQMT Result Plot](image)

Figure 4. Looking for transient quality issues with the Moscow State University VQMT Result Plot

In this case, the Superfast preset produced the distortion shown on the bottom of the frame in Figure 5 which obviously is a concern even though it's transient and only occurred once in the clip. The distortion was even worse in the Screencam clip and occurred several times, a caution sign against encoding screencams with the Superfast x.265 preset. If my library included many animated sequences, I would test to see if this type of distortion occurred frequently.
On the other hand, there were no similar issues with any of the four real-world video clips. Though Superfast produced some significant download spikes, they were very short in duration and the quality deficits were not visually discernible. In the Talking Head clip, shown in Figure 6, Superfast delivered higher quality throughout the clip with no downward spikes at all.

Given the potential savings at stake, it makes sense to test your footage to see if the Superfast preset produces transient glitches in your clips. If not, you should consider deploying the Superfast preset for your x265 encodes.
Cut Reference Frames to Boost Capacity with Minimal Quality Loss

The reference frame setting controls the number of frames the encoder analyzes to find redundancies with the frame being encoded, and this value can range up to 16 frames. Interframe compression fueled by these redundancies is much more efficient than the interframe compression used where no redundancies are found, so the more redundancies the better. That's why some compressionists jack reference frames to the max, though this obviously boosts encoding time.

Intuitively, however, you'll find most redundancies in frames proximate to the one being encoded which makes the reference frame setting another configuration option that can significantly boost encoding time without a concomitant benefit in quality. However, if you haven't customized your reference frame settings, there's probably little efficiency to be gained.

To explain, if you don't customize your reference frame setting in your FFmpeg command line argument or your encoding interface, most encoders use the setting for the selected preset, which is Medium if you haven't chosen otherwise. The reference frame setting for the Medium preset is 3, and it's 2 for Faster, which doesn't leave a lot of room for improvement since the lowest setting is 1.

On the other hand, the Veryslow preset uses a reference frame setting of 16. As you'll see, if you're using this preset and the previous section didn't convince you to switch to Faster, you can reduce the reference frame setting to a lower value and pick up lots of speed with a minimal hit in quality.

Increasing the reference frame setting from 1 to 5 boosts encoding time by about 35% with a 0.04 bump in PSNR. Not a good investment.

Table 2 shows the quality side, with PSNR values for 720p files encoded using the reference frame values shown at the top. Red backgrounds indicate the lowest quality score and green the highest. Max delta, which averages 0.31%, is the maximum difference between the highest and lowest scores. Not much quality difference to be found.

<table>
<thead>
<tr>
<th>Average Quality</th>
<th>1 Ref</th>
<th>5 Ref</th>
<th>10 Ref</th>
<th>16 Ref</th>
<th>Max Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tears of Steel</td>
<td>39.34</td>
<td>38.99</td>
<td>39.47</td>
<td>39.49</td>
<td>1.28%</td>
</tr>
<tr>
<td>Sintel</td>
<td>38.45</td>
<td>38.54</td>
<td>38.58</td>
<td>38.59</td>
<td>0.35%</td>
</tr>
<tr>
<td>Big Buck Bunny</td>
<td>39.99</td>
<td>40.09</td>
<td>40.11</td>
<td>40.11</td>
<td>0.31%</td>
</tr>
<tr>
<td>Talking Head</td>
<td>44.27</td>
<td>44.36</td>
<td>44.39</td>
<td>44.40</td>
<td>0.29%</td>
</tr>
<tr>
<td>Freedom</td>
<td>40.68</td>
<td>40.80</td>
<td>40.85</td>
<td>40.87</td>
<td>0.47%</td>
</tr>
<tr>
<td>Haunted</td>
<td>42.24</td>
<td>42.32</td>
<td>42.35</td>
<td>42.36</td>
<td>0.26%</td>
</tr>
<tr>
<td>Screencam</td>
<td>43.59</td>
<td>43.73</td>
<td>43.76</td>
<td>43.70</td>
<td>0.38%</td>
</tr>
<tr>
<td>Tutorial</td>
<td>48.58</td>
<td>48.65</td>
<td>48.68</td>
<td>48.68</td>
<td>0.22%</td>
</tr>
<tr>
<td>Average - 720p</td>
<td>42.14</td>
<td>42.18</td>
<td>42.27</td>
<td>42.27</td>
<td>0.31%</td>
</tr>
</tbody>
</table>

*Table 3. PSNR quality for different reference frame settings*
Table 4 shows the encoding time difference associated with the same reference frame settings. Increasing the reference frame setting from 1 to 5 boosts encoding time by about 35% with a 0.04 bump in PSNR. Not a good investment. In fact, you could argue that irrespective of the preset you’re using, you should manually reduce the reference setting to 1 to accelerate encoding.

<table>
<thead>
<tr>
<th>Encoding Time</th>
<th>1 Ref</th>
<th>5 Ref</th>
<th>10 Ref</th>
<th>16 Ref</th>
<th>Max Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tears of Steel</td>
<td>39</td>
<td>49</td>
<td>72</td>
<td>91</td>
<td>133%</td>
</tr>
<tr>
<td>Sintel</td>
<td>40</td>
<td>53</td>
<td>71</td>
<td>76</td>
<td>90%</td>
</tr>
<tr>
<td>Big Buck Bunny</td>
<td>41</td>
<td>53</td>
<td>68</td>
<td>85</td>
<td>107%</td>
</tr>
<tr>
<td>Talking Head</td>
<td>37</td>
<td>47</td>
<td>61</td>
<td>77</td>
<td>108%</td>
</tr>
<tr>
<td>Freedom</td>
<td>99</td>
<td>142</td>
<td>200</td>
<td>263</td>
<td>166%</td>
</tr>
<tr>
<td>Haunted</td>
<td>47</td>
<td>65</td>
<td>93</td>
<td>123</td>
<td>162%</td>
</tr>
<tr>
<td>Screencam</td>
<td>56</td>
<td>90</td>
<td>100</td>
<td>111</td>
<td>98%</td>
</tr>
<tr>
<td>Tutorial</td>
<td>23</td>
<td>24</td>
<td>24</td>
<td>26</td>
<td>13%</td>
</tr>
<tr>
<td>Average - 720p</td>
<td>48</td>
<td>65</td>
<td>86</td>
<td>107</td>
<td>123%</td>
</tr>
</tbody>
</table>

Table 4. Encoding times for different reference frame settings

In summary, while we all want the best quality video possible, you also want to shave costs where they don’t impact QoE. In the case of x264 presets and reference frames, you can do so very easily. Paradoxically, with real-world footage, you may be able to improve encoding quality and decrease encoding cost by changing your x265 preset.

**Action Steps:**

If you’re encoding with the x264 codec, check your encoding configuration and identify the preset used. If it’s Medium or higher, consider changing to Faster. Before doing so, run test encodes on your typical footage to make sure that your results are similar to mine.

If you’re not encoding with the x264 preset, run tests similar to that shown above on the whatever quality/encoding time slider your encoder uses. For example, the Adobe Media Encoder has five settings from Lowest (fastest) to Highest (slowest). Table 5, which is from the Numbers book and hasn’t been updated since 2016, shows that the Lowest preset is probably only an option for draft encodes and that either the Lower or Good increases quality with minimal impact on encoding time. From there however, encoding times hit the proverbial hockey stick and quality improvements get smaller. Since you’ll likely only be using the Adobe Media Encoder to create mezzanine files for uploading or archiving, you’re better off increasing the data rate for more quality rather than using a higher quality encoding preset.

<table>
<thead>
<tr>
<th>TOS-Main10</th>
<th>Lowest</th>
<th>Lower</th>
<th>Good</th>
<th>Higher</th>
<th>Highest</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality</td>
<td>35.78</td>
<td>37.03</td>
<td>37.75</td>
<td>38.36</td>
<td>38.70</td>
<td>8.15%</td>
</tr>
<tr>
<td>Encoding Time</td>
<td>14</td>
<td>21</td>
<td>35</td>
<td>167</td>
<td>1246</td>
<td></td>
</tr>
</tbody>
</table>

Table 5. Adobe Media Encoder HEVC presets; encoding time and PSNR quality

If you’re encoding with x265, check your encoding configuration. If you’re using the Medium preset or higher, run tests to confirm my findings above, including testing for both overall and transient quality with your different footage types.

Irrespective of your codec or encoder, check the number of reference frames in your encoding configuration. If more than two, run tests to gauge the impact of switching down to one on both encoding time and quality.
Where to Go from Here

If you’re unfamiliar with video quality metrics, or other concepts presented above, I suggest you pick up Video Encoding by the Numbers, which details how to create test clips and measure quality (see coupon for 30% off PDF version below). It also presents data similar to that shown above for a variety of configuration options, from bitrate control to I-frame interval.

30% Discount on PDF Version

Ties all key configuration decisions to quality metrics like PSNR and SSIMPLUS, eliminating the guesswork. You’ll learn how to apply these metrics to your own videos and how to choose and deploy ABR technologies like HLS and DASH.

And you’ll get a primer on encoding and packaging with FFmpeg.

30% off PDF version only (NOT on Amazon, PDF only). Click bit.ly/VEN_cart and enter coupon code five_techniques (no caps no spaces) for 30% discount ($27.97 after discount). Not available for physical book or on Amazon. Click bit.ly/numbers_book for more information on the book.

Discount expires 1/31/2020
SAVING ON ENCODING AND STREAMING: DEPLOY CAPPED CRF

Article summary: Capped CRF encoding is a single-pass technique that can save encoding costs compared to two-pass VBR. Capped CRF is also a simple per-title encoding method that can reduce your bandwidth costs and improve viewer QoE. Capped CRF is currently used by several large OTT publishers and JWPlayer in the company’s online video platform for both H.264 and VP9.

Per-title encoding customizes each encode for the complexity of the video footage, producing hard-to-encode clips at higher data rates than your normal ladder, and easier-to-encode clips at lower data rates. Since most encoding ladders are conservative, in most cases, per-title encoding reduces the data rates for most clips.

You can access per-title encoding in many forms from many different vendors. You can license optimization technology from Beamr, Crunch Media, Euclid IQ, and ZPEG, deploy on-premise encoders from Capella Systems, Harmonic, Elemental, and others with per-title capabilities, or access per-title encoding in the cloud from Bitmovin, Brightcove, JWPlayer, and Mux. Or, depending upon your encoding platform, you can roll your own via a technique called Capped CRF.

Bandwidth Savings, Increase QoE, or Both?

For some companies, deploying Capped CRF will reduce bandwidth costs. For others, it will improve the quality of experience (QoE) for their viewers. For some, it will reduce bandwidth and improve quality. It all depends upon which streams in your encoding ladder that you’re delivering to your existing customers.

To explain, consider Table 1, which shows an encoding ladder and three different stream distribution patterns, A, B, and C. Each pattern shows the percentage of each stream actually delivered from the adaptive group, as you should be able to derive from your server’s log files.

<table>
<thead>
<tr>
<th>Data Rate</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>145 kbps</td>
<td>11%</td>
<td>0%</td>
<td>2%</td>
</tr>
<tr>
<td>365 kbps</td>
<td>11%</td>
<td>0%</td>
<td>3%</td>
</tr>
<tr>
<td>730 kbps</td>
<td>12%</td>
<td>0%</td>
<td>5%</td>
</tr>
<tr>
<td>1100 kbps</td>
<td>22%</td>
<td>0%</td>
<td>5%</td>
</tr>
<tr>
<td>2000 kbps</td>
<td>22%</td>
<td>0%</td>
<td>5%</td>
</tr>
<tr>
<td>3000 kbps</td>
<td>22%</td>
<td>0%</td>
<td>10%</td>
</tr>
<tr>
<td>4500 kbps</td>
<td>0%</td>
<td>0%</td>
<td>10%</td>
</tr>
<tr>
<td>6000 kbps</td>
<td>0%</td>
<td>0%</td>
<td>20%</td>
</tr>
<tr>
<td>7800 kbps</td>
<td>0%</td>
<td>100%</td>
<td>40%</td>
</tr>
<tr>
<td>Total</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 1 Three stream delivery patterns.
In pattern A, all of the streams delivered are 3000Kbps or below, perhaps representative of distributing in a third-world country. In this case, switching to capped CRF would have no impact on bandwidth cost because you’d just be switching one 3000 kbps stream (or lower) for another. The quality would likely be improved, of course, but you’d be distributing the same bandwidth stream, so bandwidth savings would be modest.

In distribution pattern B, 100 percent of the delivered streams are the 7800Kbps stream, perhaps representative of distributing via direct fiber to the home in Scandinavia. Here, deploying capped CRF would likely reduce the bandwidth of most of your highest bandwidth streams, which would translate directly to bandwidth savings. For exceptionally hard-to-encode clips, it would also improve the QoE of your viewers.

Pattern C shows a high concentration in the top rungs and a decent spread in the other rungs, perhaps a mix of mobile and broadband. Again, deploying Capped CRF would drop the data rate of most streams in your ladder, reducing your delivery bandwidth. And, it would also improve the quality of some streams watched by your customers, improving QoE.

The obvious point is that your bandwidth savings depend upon your distribution pattern, which is data you’ll have to mine from your log files. It also depends upon how aggressive you are with your existing ladder. If your top bitrate for 1080p video is 6000 Kbps or higher, and you’re distributing lots of those streams, you’ll probably save quite a bit. If it’s 4200 Kbps, you’re already pretty aggressive and the savings will be more modest.

Note that all these observations are true for any per-title technology, not just capped CRF. They are also true for the benefits of implementing a new codec like HEVC or AV1.

**What is Capped CRF**

Constant rate factor (CRF) is an encoding mode that adjusts the file data rate up or down to achieve a selected quality level rather than a specific data rate. CRF values range from 0 to 51, with lower numbers delivering higher quality scores. Multiple codecs support CRF, including x264, x265, and VP9.

On its own, CRF is unusable for adaptive bitrate streaming, where data rates in the ladder rungs need to be limited. However, by adding a “cap” to CRF, you limit the data rate to that cap. An FFmpeg argument implementing capped CRF would look like this:

```
ffmpeg -i input_file -crf 23 -maxrate 6750k -bufsize 6750k output_file
```

This tells FFmpeg to encode at a quality level of 23, but to cap the data rate at 6750 kbps with a VBV buffer of 6750 kbps. For easy-to-encode clips, the CRF value would limit the data rate, as the required quality could be achieved at data rates lower than the cap. For hard-to-encode clips, the cap would kick in to control the data rate.

Capped CRF lacks some of the features of more sophisticated per-title technologies, like the ability to change the number of rungs in the ladder or change the resolution of some of the rungs. Still, it has always performed well in comparisons with other technologies (see [One Title at a Time: Comparing Per-Title Video Encoding Options](#) and is essentially free if your encoding tool supports it.

For Streaming Media East, I compared Capped CRF with per-title technologies from Capella Systems and Brightcove. You can see a video of my presentation and download the handout [here](#). Table 2 shows the key results.
Capped CRF Capella Systems Brightcove

<table>
<thead>
<tr>
<th></th>
<th>Capped CRF</th>
<th>Capella Systems</th>
<th>Brightcove</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage saved</td>
<td>42,259</td>
<td>39,717</td>
<td>48,824</td>
</tr>
<tr>
<td>Streaming saved</td>
<td>23,818</td>
<td>34,959</td>
<td>31,195</td>
</tr>
<tr>
<td>Net impact on VMAF</td>
<td>66.21</td>
<td>63.52</td>
<td>64.12</td>
</tr>
<tr>
<td>Saves</td>
<td>98</td>
<td>25</td>
<td>25</td>
</tr>
</tbody>
</table>

Table 2. A per-title scorecard from Streaming Media East 2018.

In the table, you see that capped CRF ranked second in saved storage, last in streaming bandwidth saved, but first in net impact on VMAF. Essentially, this means that while capped CRF didn’t reduce the data rate as much as the two other technologies, this had the beneficial result of improving the viewer quality of experience over the other two. If your goal is more streaming savings you can use a higher CRF value that lowers the data rate and decreases quality slightly. For example, I used CRF 22 for my tests, where JWPlayer uses CRF 23, which delivers more bandwidth savings.

One key benefit of capped CRF is that it’s a single-pass technology. If you’re currently using a two-pass technique, capped CRF will also significantly increase your capacity or cut costs. In contrast, most other per-title technologies actually require an additional analysis pass prior to the actual encode, which may boost your encoding cost or decrease your capacity.

The single-pass nature of capped CRF is reflected in the 98 “saves,” representing one pass for each of the seven rungs in the fourteen test files. Capella and Brightcove got their saves by eliminating rungs from the encoding ladder for easy-to-encode clips, though this doesn’t factor in the analysis pass both systems use for their per-title encodes (it will next time).

**Bitrate Control Concerns**

One concern about capped CRF is that because there are no bitrate controls other than the cap, there could be huge data rate swings within the file that potentially disrupt the switching algorithm used by your selected ABR technology. The file shown in Figure 1 contains a mix of ballet (the peaks) combined with a talking head video (the valleys) causing the data rate within the file to vary from under about 3 Mbps to 6 Mbps.

![Figure 1. Significant data rate swings in this file encoded using Capped CRF.](image-url)
In truth, most other VBR technologies deliver a similar file. For example, Figure 2 shows the data rate of the same file encoded with FFmpeg using 200% constrained VBR. In this file, the valleys are about the same but the peaks slightly higher. So, if you’re using 200% constrained VBR now, capped CRF shouldn’t cause any concerns.

![Bitrate Viewer](image)

*Figure 2. Even worse data rate swings in this file encoded using 200% constrained VBR.*

On the other hand, if you’re using CBR because you believe it maximizes file deliverability, then capped CRF is definitely not for you. From my perspective, the fact that JWPlayer continues to use capped CRF after several years of deployment allays most of those concerns.

**Careful With Screencams**

I test per-title technologies with about 20 test clips including three or so screencams or similar synthetic clips. While writing this article, I tested to see if CRF seriously degraded the quality of any clip, which is simplified by the Moscow State University VQMT’s Result plot, shown in Figure 3.
Figure 3. The result plot shows significant qualitative differences between capped CRF and 200% constrained VBR.

Briefly, I used PSNR on these analyses (rather than VMAF) since it computes so much faster and is a great canary in the coal mine for quality issues. Here, I'm analyzing 200% constrained VBR (orange) and capped CRF (green), with the top graph showing the PSNR values for the two files over the entire file, and the bottom showing the highlighted region from the top (about 55% – 65%). The significant deltas between the values often points to very noticeable qualitative differences.

If you click Show Frame on the bottom right of the Result Plot UI, you can toggle through the source frame and frames from the two analyzed clips. Figure 4 shows a portion of the screen from the capped CRF clip, which is clearly degraded.

Figure 4. Indicating a few frames like this.
Note that this was, by far, the biggest qualitative difference I saw in the three synthetic clips, and I saw no meaningful differences in the real-world clips or animations. The comparisons for most real-world clips looked like Figure 5, a high motion clip where CRF delivered a slightly higher data rate than 200% constrained VBR and slightly higher quality, but no major deltas from the 200% constrained VBR plot.

Figure 5. The CRF clip was consistently slightly higher than the 200% constrained VBR.

So, while I would not recommend capped CRF for screencam and similar synthetic footage without additional testing, I’m comfortable recommending it for real-world videos and animations.

Deploying Capped CRF

Deploying capped CRF encoding is simple so long as your encoder allows you granular control over your encoding parameters. For example, Figure 2 is a screenshot from the browser-based user interface of the Hybrik cloud encoder. As you can see, you choose the CRF bitrate mode, then enter the max_bitrate and vbv_buffer size values (the entry for CRF value is further below and isn’t shown). If you were using the API, you would configure the same parameters via JSON. Most cloud encoders are built around FFmpeg, so you may be able to access CRF encoding if another per-title method isn’t available.

Figure 2. Selecting capped CRF in the Hybrik cloud encoder.
If your desktop encoder doesn’t allow you to select crf as a bitrate, you may be able to enter x264 commands directly within the user interface, which was a feature of Telestream Vantage last time I checked. If you can access CRF controls, you’ll substitute these for your previous bitrate control method, whether CBR or VBR.

### Capped CRF in FFmpeg

The batch file below shows the test ladder from the Streaming Media comparison, absent the GOP, preset, audio, and other commands you’d normally see in an FFmpeg batch. I’ve changed the CRF value to 23 to match JWPlayer. I set the max rate and buffer size at 1.5x times the original target data rate, which was 4500 kbps for the 1080p stream. JWPlayer also sets the same value for data rate and buffer size, though I’ve seen other documentation where the buffer was twice as high as the target.

```
ffmpeg -i Input.mp4 -c:v libx264 -crf 23 -maxrate 6750k -bufsize 6750k Output_1080p.mp4
ffmpeg -i Input.mp4 -c:v libx264 -crf 23 -s 1280x720 -maxrate 4050k -bufsize 4050k Output_720p.mp4
ffmpeg -i Input.mp4 -c:v libx264 -crf 23 -s 960x540 -maxrate 2850k -bufsize 2850k Output_540p.mp4
ffmpeg -i Input.mp4 -c:v libx264 -crf 23 -s 852x480 -maxrate 2025k -bufsize 2025k Output_480p.mp4
ffmpeg -i Input.mp4 -c:v libx264 -crf 23 -s 640x360 -maxrate 1350k -bufsize 1350k Output_360p.mp4
ffmpeg -i Input.mp4 -c:v libx264 -crf 23 -s 480x272 -maxrate 750k -bufsize 750k Output_272p.mp4
ffmpeg -i Input.mp4 -c:v libx264 -crf 23 -s 320x180 -maxrate 375k -bufsize 375k Output_180p.mp4
```

*Batch 1. Encoding a full ladder with capped CRF.*

Note that you can adjust all these parameters to achieve your specific delivery and quality of experience goals. Lower CRF values like 21-22 will deliver higher bitrates and higher quality, while higher values like 24-25 will do the opposite.

### What About Duplicate Resolutions

Conveniently, Batch 1 contains seven rungs with different resolutions. This simplifies things, because so long as you use the same CRF value in all rungs, larger resolutions should always have higher data rates, preserving the necessary data rate progression for effective stream switching.

However, what happens if you have three rungs at 720p, say at CRF 21, 23, and 25? How can you be sure that the 720p@CRF 25 rung has a higher data rate than the next lower rung, say at 540p@CRF 21. You’ll almost certainly encounter this issue with 4K video footage where ladders can have 9 – 11 rungs.

I’ve run into this situation once when working with VR 4K footage. In that case, I ran test encodes on multiple clips at different resolutions and CRF values. With this data, I created a ladder that utilized several resolutions (like 4K, 1080p, and 720p) multiple times with different CRF values. I then tested the ladder with multiple clips that ranged in encoding complexity from simple to insane to make sure the ladder maintained the necessary data rate spread between all rungs.
With extremely simple clips, the lower rungs tended to get very close together, simply because you don’t need 11 rungs if the top rate is 5 Mbps. Still, the ABR groups were workable. The harder-to-encode files that I tested seemed to work perfectly.

**Where to Go from Here**

You should have everything you need to start experimenting with capped CRF. If you’re unfamiliar with video quality metrics, or other concepts presented above, I suggest you pick up *Video Encoding by the Numbers* (see offer on PDF version on page 11). If you’d like to start experimenting with FFmpeg, consider picking up *Learn to Produce Videos with FFmpeg: In 30 Minutes or Less (2018 Edition)* or the FFmpeg course shown below.

---

**30% off PDF Version and Course**

This focused 158-page book teaches you how to use FFmpeg to create full adaptive bitrate ladders and package them into HLS or DASH formats.

30% off PDF version (NOT on Amazon, PDF only). Click bit.ly/FFmpeg_cart and enter coupon code *five_techniques* (no caps no spaces) for 30% discount (~$21 post discount). Not available for physical book or on Amazon. Click bit.ly/ozerr_ffmpeg for more information on the book.

Learn to use FFmpeg, Bento4, and Apple's HLS tools to produce optimized videos for delivery to HLS and DASH clients. Each lesson covers theory and practice so you can choose the best option and use the optimal command syntax.

Click bit.ly/learn_ffmpeg for more info or to buy the course. Enter coupon code *five_techniques* (no caps no spaces) for 30% discount (~$21 after discount).

Both discounts expire 1/31/2020
SAVING ON ENCODING AND DELIVERY: DYNAMIC PACKAGING

You can dramatically reduce net encoding and storage costs by implementing dynamic packaging for your live or VOD video. This article defines dynamic packaging, explores its benefits, and outlines some implementation options.

The image above is from Bitmovin’s Video Developer Report 2019, which incorporates 542 survey submissions from 108 countries. The graph reflects the reality that if you’re distributing video with DRM, you likely need to support at least two ABR formats, HLS and DASH. If you’re supporting older gaming platforms, you may also need to support Smooth Streaming.

If this reflects your reality, the most cost-effective way to support these multiple formats is via dynamic packaging, also called just-in-time packaging. In this article, I’ll describe what dynamic packaging is, how it works, and how it can reduce your overall streaming expense. I’ll also address how quickly the Common Media Application Format (CMAF) will obsolete dynamic packaging (not soon) and instances where dynamic packaging may be contraindicated.

About Dynamic Packaging

The traditional way to serve multiple ABR formats was to encode and package each format separately, then upload all files to the origin servers for delivery. From a cost perspective, with some cloud encoding vendors, each encode would be a completely separate charge, though many vendors now change less for outputting multiple ABR formats. Since you’re storing multiple ABR versions of each video in the cloud, each additional ABR format that you support boosts your storage cost by 100%.
Dynamic packaging works for live and VOD video and is shown in Figure 2 from Microsoft Azure documentation. You create the content in two phases. You encode your encoding ladder once, and upload the MP4 files to a streaming endpoint. From there, a server on the endpoint detects the playback capabilities of the client and dynamically produces the necessary ABR format. Since the server is merely changing the container format of the file (also called transmuxing) and perhaps adding DRM, this is a very fast and lightweight operation with minimal latency.

![Dynamic Packaging Diagram](image)

Because you’re only encoding the content once, you save on encoding costs, and because there's only one version of the file at the origin server, you save on storage costs. Balanced against this is the cost of the server component.

Obviously, if you pay Microsoft Azure to package and deliver for you, the cost is bundled into the service. However, if you license the Wowza Streaming Engine or Nimble Streamer to package your content, you have to incorporate the cost of both the server software and the cloud machine that you’ll run it on. To calculate whether dynamic packaging will save you money, identify the encoding and storage savings and offset that against the cost of the server.

Savings can be significant. During the transition year, dynamic packaging reduced the encoding costs of the library transcode and ongoing encodes by close to $90,000 and cut storage costs by around $34,000. Add back the $20,000 or so it would cost to run Wowza Streaming Engine, and total savings exceeded $100,000.

Savings can be significant. For example, I worked with a client who was converting a large existing library to ABR streaming, while producing significant hours of new videos. During the transition year, dynamic packaging reduced the encoding costs of the library transcode and ongoing encodes by close to $90,000 and cut storage costs by around $34,000. Add back the $20,000 or so it would cost to run Wowza Streaming Engine, and total savings exceeded $100,000.

**Getting Dynamic Packaging**

In addition to the vendors already identified, you can access dynamic packaging from multiple vendors with at least one open source alternative. For example, AWS Elemental Technologies offers a software solution, AWS Elemental Delta, and an SaaS solution, AWS Elemental MediaPackage. Harmonic offers ProMedia Package, while Brightcove deploys Dynamic Delivery on their Video Cloud platform. The NGINX Plus server can dynamically package live video into HLS, DASH, and RTMP format, and convert VOD files in MP4 format to HLS and HLS. The bottom line is that you’ll have multiple options for dynamic packaging, whether you’re seeking to buy it, license it, or access it as part of a SaaS.
What About CMAF

By way of background, when initiated, Apple HLS only supported files in the MPEG-2 transport stream format (.ts) while DASH supported fragmented MP4 files (fMP4). This prevented one set of files from being distributed to both DASH and HLS clients. In 2016, Apple started supporting fMP4 files compatible with the Common Media Application Format spec that Apple co-authored with Microsoft. CMAF uses the ISO Base Media File Format (ISOBMFF) container—with common encryption (CENC); support for H.264, HEVC, and other codecs; and WebVTT and IMSC-1 captioning. For compatibility, CMAF can be called by both HLS playlists (.m3u8 files) and DASH manifest files (.mpd files). For HLS delivery, CMAF will replace files currently packaged in the MPEG-2 container format.

The problem is that CMAF also enables two incompatible common encryption modes: cipher block chaining (CBC) for Apple’s FairPlay digital rights management technology (DRM), and counter mode (CTR) for PlayReady, Widevine, and other DRMs. Content encrypted with CBC can’t be decrypted by PlayReady- and Widevine-compatible clients, while content encrypted with CTR can’t be decrypted by FairPlay clients.

Since 2016, Google has added CBC support to Widevine while Microsoft has announced support for CBC in PlayReady 4.0. This means that one set of CMAF-formatted content encrypted using CBC should be able to play in all modern DASH and HLS players.
The problem is that there are many generations of Apple devices that aren’t compatible with CMAF and can’t play content that’s not in an MPEG-2 container. So, any service supporting legacy devices won’t be able to move to a totally CMAF-based solution for several years.

Where Dynamic Delivery is Contraindicated

There are a couple of scenarios where dynamic delivery is contraindicated. One is very large events. Specifically, though dynamic packaging works well for hundreds or even thousands of simultaneous users, Will Law, chief architect of media cloud engineering at Akamai, points out that when streaming gets to a truly massive scale, like tens of millions of connections, the extra work associated with dynamic packaging is magnified. According to Law, this reduces the throughput of Akamai’s edge servers, and the extra complexity associated with dynamic packaging increases the opportunity for workflow issues to arise.

The other potential problem are involves live streams with advertising insertion, where dynamic packaging has caused problems with at least one executive that I spoke with. Otherwise, if you’re still creating separate file stores for each video format for either live or VOD delivery, you should at least explore whether dynamic packaging can reduce your net encoding and storage charges.

Where to Go From Here

You’ve got the background, now it’s time to start exploring the class of alternatives that are right for you. Note that Dynamic packaging is covered (along with 12 other topics in the Encoding and Packaging for Multiple Screen Delivery 2019 course shown below (with 30% off).

This course teaches you the applicable standards and best strategies for delivering live and VOD adaptive video to viewers on all platforms, with and without DRM. You’ll learn dynamic and static options for producing H.264 and HEVC; the status of standards like the Common Media Application Format (CMAF), Media Source Extensions (MSE) and Encrypted Media Extensions (EME); and how and when to utilize them.

Enter coupon code five_techniques (no caps no spaces) for 30% discount.

[Link to course]

Discount expires 1/31/2020
SAVE ON ENCODING COSTS: CUT CLOUD ENCODING CHARGES

Cloud encoding is a great alternative for producers who don’t want to invest CAPX for their own encoders and pay for the in-house expertise necessary to keep the encoders up and running. But it’s very easy to overpay for cloud encoding; here’s a guide to help you avoid doing so.

Cloud VOD encoding is gaining popularity for many streaming producers for many reasons (this article will only discuss VOD, not live). This demand has spawned many different product offerings, and it can be very difficult to understand what you’re getting and what you’re paying for, and what you may not be getting.

A great place to start is the new price list for the [Professional Tier of Amazon Media Convert](https://aws.amazon.com/media-convert/pricing/). I’ve copied the H264 pricing data into the table below and added some columns and rows to identify the premiums for extra services. I included this information because AWS Elemental is one of the few sites that clearly delineated what you pay for the various options and because it illustrates the cost side of the encoding equation. All cloud vendors run on AWS or a different cloud platform and pay similar costs to produce your output; what each vendor passes on to the customer is up to them.

<table>
<thead>
<tr>
<th>H264</th>
<th>1 pass (speed optimized)</th>
<th>2 pass (quality optimized)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&lt;=30 fps</td>
<td>&gt;30 and &lt;=60 fps</td>
</tr>
<tr>
<td>SD</td>
<td>$0.0120</td>
<td>$0.0150</td>
</tr>
<tr>
<td>HD</td>
<td>$0.0240</td>
<td>$0.0300</td>
</tr>
<tr>
<td>UHD</td>
<td>$0.0480</td>
<td>$0.0600</td>
</tr>
<tr>
<td>Premium</td>
<td>125%</td>
<td>150%</td>
</tr>
</tbody>
</table>

Table 1. AWS Elemental Professional Tier pricing for H.264.
Let's start with resolution. Like many vendors, AWS Elemental doubles the price for HD and doubles it again for UHD. Obviously, this is because more pixels take longer to encode. AWS Elemental does the same for higher frame rates, again for the same reason; more frames mean more pixels to encode. There’s also a 175% premium for two-pass encoding, which in most cases delivers higher quality than single pass. Interestingly, not all vendors offer two-pass encoding; most notably, the Amazon Elastic Transcoder does not.

Questions you should know the answer to for your cloud encoding facility are:

- What’s the premium you’re paying for HD and UHD?
- Is two-pass available, and if so, is there a premium?
- If you’re using single-pass encoding (say for capped CRF), is there a discount?
- Can you customize encoding parameters to improve quality if it extends encoding time; for example, using the veryslow x264 preset rather than fast? If so, is there a premium?

Table 2 shows AWS Elemental pricing for HEVC. Single-pass encoding pricing is double that of H264, which is reasonable, and premiums for HD, UHD, and high frame rates are the same as they were for H264. However, you’ll pay 4x for balanced single-pass encoding and 7x for quality optimized 2-pass encoding, peaking at about $1/minute for high frame rate UHD content.

<table>
<thead>
<tr>
<th>HEVC</th>
<th>1 pass (speed optimized)</th>
<th>1 pass HQ (balanced)</th>
<th>2 pass HQ (quality optimized)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&lt;=30 fps</td>
<td>&gt;30 and &lt;=60 fps</td>
<td>&gt;60 and &lt;=120 fps</td>
</tr>
<tr>
<td>SD</td>
<td>$0.024</td>
<td>$0.030</td>
<td>$0.036</td>
</tr>
<tr>
<td>HD</td>
<td>$0.048</td>
<td>$0.060</td>
<td>$0.072</td>
</tr>
<tr>
<td>UHD</td>
<td>$0.096</td>
<td>$0.120</td>
<td>$0.144</td>
</tr>
<tr>
<td>Premium</td>
<td>125%</td>
<td>150%</td>
<td>400%</td>
</tr>
</tbody>
</table>

Table 2. AWS Elemental Professional Tier pricing for HEVC.

More questions you should know the answer to include:

- What’s the premium for higher quality codecs like HEVC, VP9, and AV1?
- How will this vary by encoding configuration?

Having experimented with encoding HEVC, Elemental’s pricing accurately reflects the cost side of the equation; two-pass encoding can easily take five – seven times longer to encode than single pass, particularly if you use a speed-optimized configuration for one-pass and quality-optimized for two-pass. The question isn’t whether it’s reasonable or not, but what you’re getting and what you’re paying for.

If you’re producing multiple ABR formats via a cloud encoder, the obvious question is whether you’re paying full price for each, or whether you’re getting a discount to reflect the actual processing involved. If you’re paying full price for each format you’re very likely paying too much.
Transmux Fees

Another area of price differentiation surrounds transmuxing fees or converting files in an encoding ladder from one ABR format, say HLS, to another, like DASH or Smooth Streaming. Where encoding to H264 or HEVC is very CPU intensive, converting from HLS to DASH is not. Some vendors, like Telestream (shown in Figure 3), Bitmovin, and Zencoder, charge less for transcoding; many do not.

<table>
<thead>
<tr>
<th>Others</th>
<th>Multiplier</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio</td>
<td>0.25x</td>
<td>10 minutes of Audio output count as 2.5 SD minutes</td>
</tr>
<tr>
<td>Transmuxing</td>
<td>0.25x</td>
<td>10 minutes of Transmuxing output count as 2.5 SD minutes</td>
</tr>
</tbody>
</table>

Figure 3. Telestream charges less for transmuxed output.

If you’re producing multiple ABR formats via a cloud encoder, the obvious question is whether you’re paying full price for each, or whether you’re getting a discount to reflect the actual processing involved. If you’re paying full price for each format you’re very likely paying too much.

When the Cost of Switching is High?

Once you understand your pricing structure, it’s time to consider your alternatives. The first thing to assess is the cost and risk of switching vendors. To the extent that you’ve implemented pre-encoding workflows in the cloud, the cost of switching goes up, as not all cloud encoding facilities may be able to replicate these. The same is true if you’re looking to supplement your in-house encoding facility with cloud-based encoding for spikes in encoding demand. In these cases, using the same vendor for in-house and cloud means controlling all encoding from a single interface and even using the same encoding presets. That’s a lot of convenience and time saved.

If the cost of switching from your current vendor is too high, ask if your vendor offers different pricing options. For example, encoding.com offers reserved cloud pricing that lets you buy the capacity of a cloud machine for a fixed price per month. You can encode and package as much video as you can during that month. For many customers, this can present significant cost savings over encoding.com’s normal per GB pricing. Some vendors will also drop the price if you guarantee certain levels of throughput.

Other vendors, like Bitmovin, let you license their cloud encoder for use on your own computing infrastructure (see the figure on page 24). In this case, the price per minute is much less than you would pay for encoding the same output via Bitmovin’s SaaS.

Otherwise, Consider other Encoding Shops

Outside of these scenarios, understand that from a strict output format perspective, most outputs are commodities. Most cloud encoders are built around FFmpeg and use the x264 and x265 codecs. DASH is DASH and HLS is HLS. By this point, most credible encoding shops can handle multiple languages for audio and captions and multiple DRMs. Though it will take some experimentation to produce the equivalent output and programming time to support a different API, it’s not rocket science, and if the economics dictate a change, it’s a change you can and should make.
As you’ve learned, though most vendors still offer per GB or per-minute pricing, several vendors offer new pricing models for different types of clients. Cloud vendor **Hybrik** is another alternative. Briefly, Hybrik runs on the Amazon cloud, and you allocate machines using your own Amazon account. Hybrik charges a flat fee that depends on how many machines you can simultaneously assign to the system; 10 machines are $1,000/month, 100 machines are $5,000/month, and so on.

What’s great about reserved cloud and Hybrik-style pricing is you know exactly what you’re paying for and can make the quality/encoding-time trade-offs that you want to make. It really is as close to running your own encoding farm as you can get without actually running your own encoding farm. You can download a white paper I wrote for Hybrik comparing their pricing model to other vendors [here](bit.ly/hyb_pricing).

**Where to Go From Here**

Whether you’re currently using a cloud encoder or are shopping for a provider, expect price comparisons to be challenging. Not only will you have to calculate costs under both per-minute and per-GB pricing, you may have to compute costs under reserve cloud or Hybrik-style pricing. Most cloud vendors do an awful job identifying all the pricing alternatives on their websites, so expect to spend some time contacting each vendor for details.

Video quality metrics like PSNR, SSIM, VMAF, and SSIMPLUS predict how subjective viewers will evaluate digital videos. Many large producers like Netflix and Facebook use video quality metrics to guide day-to-day encoding decisions like choosing the optimal encoding settings, creating encoding ladders, and evaluating codecs and encoding tools. This course teaches you how to compute and apply these metrics.

Enter coupon code `five_techniques` (no caps no spaces) for 30% discount.

[bit.ly/SLC_VM (~$70 with discount)]

Discount expires 1/31/2020
SAVING STREAMING COSTS: ADDING A NEW CODEC

Introduction

The calculus for adding a new codec sounds simple, as in, “HEVC is 50% more efficient than H.264, so implementing HEVC will shave 50% of your bandwidth costs. That will cover encoding costs in no time.” Well, maybe yes, maybe no. We can say that adding a new codec will improve the quality of experience of your viewers, or save bandwidth costs, and maybe do a bit of both. Be we can’t say any of that without first doing a lot of analysis.

In this article, you’ll learn how to compute how many views of a video you'll need to deliver to recover the costs of encoding using another codec. This certainly isn’t the complete picture of the cost/benefit of deploying a new codec, but it’s a critical component of that analysis, and the most I could bite off in a reasonably-sized post. This analysis is for VOD only, not live, which has a complete different set of numbers.

The breakeven formula itself is simple; divide encoding cost per hour by bandwidth savings per hour, and you have the number of viewing hours necessary to recoup the encoding cost. Unfortunately, while the formula is simple, the components of the numerator and denominator may be challenging to derive.

\[
\text{Hours to recoup cost} = \frac{\text{Encoding Cost / hour}}{\text{Bandwidth Savings / hour}}
\]

*Figure 1. Computing streaming hours necessary to recoup encoding costs.*

Let’s begin with the numerator, which is the easier of the two values to nail down.

**Encoding Cost/Per Hour**

Obviously, your encoding costs will depend upon your encoding ladder. For the purposes of this discussion, we’ll use the ladder shown in Table 1, which is Apple’s recommended HEVC ladder less the top rung, which is overkill for most producers. This also shows the HEVC encoding cost from Bitmovin. To compute these numbers, I assumed the lowest per-minute cost published on the Bitmovin site ($0.025), which is a tier that will cost you $1,299/month and includes 60,000 minutes.

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Bitrate</th>
<th>Multiple</th>
<th>Cost/hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>640x360</td>
<td>145</td>
<td>2</td>
<td>$3.00</td>
</tr>
<tr>
<td>768x432</td>
<td>300</td>
<td>2</td>
<td>$3.00</td>
</tr>
<tr>
<td>960x540</td>
<td>600</td>
<td>2</td>
<td>$3.00</td>
</tr>
<tr>
<td>960x540</td>
<td>900</td>
<td>2</td>
<td>$3.00</td>
</tr>
<tr>
<td>960x540</td>
<td>1600</td>
<td>2</td>
<td>$3.00</td>
</tr>
<tr>
<td>1280x720</td>
<td>2400</td>
<td>4</td>
<td>$6.00</td>
</tr>
<tr>
<td>1280x720</td>
<td>3400</td>
<td>4</td>
<td>$6.00</td>
</tr>
<tr>
<td>1920x1080</td>
<td>4500</td>
<td>4</td>
<td>$6.00</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td><strong>$33.00</strong></td>
</tr>
</tbody>
</table>

*Table 1. Computing HEVC and VP9 encoding cost for an hour of video.*
The per-minute cost is for H.264 encoding, which Bitmovin doubles for HEVC and VP9. There's a 2X premium for HD videos (1280×720 – 1920×1080), producing the multiple of 4 for those resolutions. Multiply the per-minute cost ($0.025) times the multiple (2x or 4x) times 60 minutes to get per-hour cost for each rung, with the total at the bottom.

If you’re using a different cloud facility, you can run the same analysis as Table 1. If you run your own encoding facilities, and you know the per-minute cost for H.264 encoding, multiply that by four for both HEVC and VP9. If you’re encoding AV1, multiply that by 20 to calculate the encoding cost.

A couple of observations here. First, the obvious—the higher the cost per hour to encode, the more views necessary to recoup that cost. In addition, since you may be converting existing libraries into the new format, you could be looking a huge up-front expense. The bottom line is that if you decide to start encoding with a new codec or codecs you should make absolutely sure that you're encoding at the lowest possible cost. I covered cloud encoding costs in the previous article here. Second, don't forget transmux charges. For example, if you decide to deploy HEVC, you'll need HLS formatted video for Apple devices, and DASH formatted video for Smart TVs and other devices. You might be able to use a single set of files in CMAF format, but then again, you may not, and you certainly won't be able to if your encoder/packager doesn't support it.

There can be huge cost differentials for producing the same set of files in multiple formats. When computing encoding costs, be sure to fully flesh out the costs involved with supporting all the formats necessary to deploy the new codec.

Bandwidth Savings Per Hour

This is where things start to get hairy. There are two components to this number, how efficient the new codec is compared to H.264, and how much of this efficiency you’ll actually be able to utilize. Let’s start with quality.

The Quality Differential Between H.264 and Newer Codecs

In their HEVC Video Codecs Comparison, Moscow State University found that VP9 and the best performing HEVC codecs were about 40% more efficient than x264 when encoding 1080p files. Using a different set of clips and procedures (SD and HD clips), Facebook found that AV1 produced the same quality as x264 at about a 50% bitrate saving. Not all the clips in your encoding ladder are 1080p, but typically the higher resolution clips tend to get viewed the most.

So, let’s keep things simple. In the HEVC-related analysis below I’ll use an overall bitrate savings of 30%. I would use 30% for VP9 and 40% for AV1, but I’m not going there in this analysis.

Now that we know the potential savings, the big question is how much of these savings can you actually harvest. For that, you’ll have to visit your log files.

How Much Savings?

Table 2 shows our encoding ladder with a column for percent viewed, which is the percentage this particular rung is watched by a viewer, a statistic you’ll have to harvest from your log files.

Here's the funky part; you only achieve bandwidth savings from viewers who are watching your lowest quality and your highest quality streams. I’ll explain. In Table 2, assume that today, 15% of your viewers are watching your 540p H.264 stream encoded at 2,200 kbps. Tomorrow, after switching to HEVC, these viewers will almost certainly be watching the 720p HEVC stream encoded at the same bitrate. The quality of the video should improve, but the outbound bandwidth will be exactly the same. Higher QoE, but no bandwidth savings.
Table 2. Bitrate savings from encoding ladder with a general distribution pattern.

Up and down the encoding ladder the result is the same; perhaps some small decreases or increases in the switch, but no wholesale benefits except for the lowest and highest rung.

Viewers connecting at 200 kbps will most likely connect at 140 kbps with HEVC, but the data rate is very low so the savings are minuscule. At the top rung, however, we’re substituting the 4,550 HEVC stream for the 6,500 H264 stream, a 30% true savings. However, since only 15% of viewers are watching that stream, the overall savings is minor, only about 130 MB/hour, or about 13% of a GB.

Compare that to Table 3 which has a top-heavy distribution pattern, and shows a bandwidth savings of about half a GB per hour. Interestingly, as the bandwidth savings increase, the QoE improvement decreases, at least as measured solely by video quality. That is, a viewer previously watching the 1080p H.264 video encoded at 6,500 kbps is now watching the 1080p video encoded at 4,550, which should be nearly identical in quality. There may be other quality-of-experience benefits from retrieving a lower bitrate stream, like reduced stream switching or more available bandwidth for other users, but the video quality should be about the same.

Table 3. Bitrate savings from an encoding ladder with a top-heavy distribution pattern.
Note that the encoding ladders that you actually deploy for HEVC will be different than for H.264 for reasons I explain in Apple Got It Wrong: Encoding Specs for HEVC in HLS. However, as presented, Tables 2 and 3 simplify the point I’m trying to make in this section.

Note also that your savings will reflect the actual rungs that you deploy with the new codec. To save encoding costs, some producers are adding HEVC rungs only at 720p and higher resolutions, an approach I explored here, while others are creating two completely different ladders with H264 and HEVC (like Apple here). Most producers that use VP9 seem to deploy a completely separate ladder. Obviously, your bandwidth savings/QoE improvements will reflect these decisions.

The takeaway from this section? You need to compute the GB saved per hour by switching to the new codec. This combines the estimated bandwidth savings from the codec and how much of that savings you’ll achieve based upon your encoding ladder analysis.

**Putting it All Together**

Table 4 computes the hours to breakeven at multiple cost per hours and bandwidth costs. The only input is the bandwidth savings per hour of video on the top left (.5 GB in this case). Find your encoding cost per hour and bandwidth cost and you’ll see the hours of videos necessary to recoup the cost of encoding a single hour.

<table>
<thead>
<tr>
<th>Bandwidth savings (GB/hour)</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bandwidth Cost (per GB)</strong></td>
<td>$0.085</td>
</tr>
<tr>
<td>$5</td>
<td>118</td>
</tr>
<tr>
<td>$15</td>
<td>353</td>
</tr>
<tr>
<td>$20</td>
<td>471</td>
</tr>
<tr>
<td>$25</td>
<td>588</td>
</tr>
<tr>
<td>$30</td>
<td>706</td>
</tr>
<tr>
<td>$35</td>
<td>824</td>
</tr>
<tr>
<td>$40</td>
<td>941</td>
</tr>
<tr>
<td>$1,500</td>
<td>35,294</td>
</tr>
</tbody>
</table>

*Table 4. Cost per hour at the stated encoding costs and bandwidth costs.*

You can download an Excel file with this table and other tables here.

Your ability to achieve and surpass these hours depends upon a number of factors, including which platforms support each new codec and how that maps with your typical viewers. If you’re distributing solely to Smart TVs, HEVC should make a lot of sense. If distributing primarily to computers for browser-based playback, VP9 is a natural. For a discussion of this and other codec implementation issues, check out Return of the Codec Wars: A New Hope—a Streaming Summer Sequel.

**Final Thoughts**

Some final thoughts. First, this isn’t a total break-even analysis, it’s a hopefully useful computation to figure out whether deploying a new codec makes sense of not. But there are many additional expenses to add into the breakeven analysis, including player development costs, encoder selection costs, ladder development costs, and debug and quality control costs. For many OTT providers, the math should be pretty compelling. For the vast majority of smaller sites that distribute each video in the low hundreds of hours, it probably isn’t.
Second, the $1,500 cost per hour for encoding in Table 4 isn’t a joke, it’s a rough estimate of the encoding cost of AV1 and shows why AV1 makes sense for Netflix, Hulu, Facebook, Amazon, and others who distribute videos in the millions of copies. Famously, over 16 million people watched the premiere of Game of Thrones season 7 on the first night it was available; had AV1 been around back then, it would have been the gift that keeps on giving.

Third, the HEVC royalty situation has slowed adoption since the codec first became available in 2013. While two of the three royalty groups have abandoned royalties on non-physical content (i.e. streamed or downloaded), the Velos pool refuses to make the same decision. I checked the Velos FAQ today, and it still states “As it relates to content, we will take our time to fully understand the dynamics of the ecosystem and ensure that our model best supports the advancement and adoption of HEVC technology.”

If Velos didn’t intend to pursue content royalties, you would assume they would have said so by now, since it’s such a substantial issue for so many producers. So, at the very least, you need to get your CFO and/or corporate counsel involved in the decision to implement HEVC. If you need to estimate royalty costs to include in your break even analysis, note that MPEG LA charges $0.02/title for H.264 with a 9.75M annual cap, while HEVC Advance charges $0.0225 for HEVC video shipped on physical media with a $2.5 million annual cap.

**Where to Go From Here**

Adding and configuring a new codec involves many steps, from analysis to encoder selection to ladder creation. If you’d like some help running this analysis, contact me (Jan Ozer) at janozer@gmail.com. The course below includes modules on encoding with HEVC as well as dynamic packaging.

This course teaches you the applicable standards and best strategies for delivering live and VOD adaptive video to viewers on all platforms, with and without DRM. You’ll learn dynamic and static options for producing H.264 and HEVC; the status of standards like the Common Media Application Format (CMAF), Media Source Extensions (MSE) and Encrypted Media Extensions (EME); and how and when to utilize them.

Enter coupon code *five_techniques* (no caps no spaces) for 30% discount.


Discount expires 1/31/2020