Codec Testing and Comparison

Warning

Message Date: 04/12/2018

The alliance for Open Media froze bitstream format on March 28, 2018 and announced version 1.0 of the codec on the same day. It is not clear if the bitstream work has actually been finalized or if this was done to present a "finished" product at NAB.

VLC has support for AV1 playback but it doesn't work with the latest build of the encoder reference implementation. I have yet to play an AV1 file (that I encoded or downloadded from third parties) successfully in a release (3.0.1 on MacOS) or nightly (041218 on MacOS).

As of the writing of this article FFMPEG does not support AV1. I expect this to change now that the bitstream is frozen, however with all the active work going on with the codec, I don't know when/if that will happen.

For AV1, I run the code from the https://aomedia.googlesource.com/aom/ and update it before every experiment using the instructions on the repository. Because I'm running the code from Git I'm playing with fire as the code may have bugs or other unexpected issues. The flip-side is that I get the most complete implementation of the codec and its tools available.

Note:

The Repository at https://github.com/caraya/video-encoding-tests contains the scripts used on this test along with 2 example videos for you to duplicate the tests on your system.

Hardware Specs

- Model Name: MacBook Pro
- OS Version: MacOS 10.13 (High Sierra)
- Processor Name: Intel Core i7
- Processor Speed: 3.1 GHz
- Number of Processors: 1
- Total Number of Cores: 4
- L2 Cache (per Core): 256 KB
- L3 Cache: 8 MB
- Memory: 16 GB

Codecs to test

The idea is to test the following codecs, both old and new, to get a better idea of they answer the questions

- x264
    - Run through ffmpeg
- x265
    - Run through ffmpeg
- vp9
    - Run through ffmpeg
- aom for av1 support
    - Compiled and installed from source

AVC/H264 is the current generation codec and what most people use to create video on the web.

HEVC/H265 is the successor to H264 and produces smaller files of equivallent quality.

VP9 is the successor to VP8 and its primary use is in Youtube and some specific settings for HTML5 video (HDR video).

AV1 is a open source, royalty-free video codec developed by the Alliance for Open Media, a large consortium of software and hardware companies. The most attractive characteristi are:

- It’s royalty free

- It claims at least 20% better quality than HEVC at the cost of slower encode speeds

Once we've decided on the codecs we can start setting objectives to measure. Most of these are subjective and some need time before they can be fully tested since AV1 still doesn't play in VLC and the versions that will play embedded in Firefox are, as of this writing, tied to specific commit hashes for the reference encode/decoder implementation (explained in this Mozilla Hacks Article).

The basic questions that I seek to answer with this experiment:

1. Is there's any pereceptible difference between codecs?
    1. Do they look different? Is there a subjective difference?
2. How the storage requirements change between formats
3. How long do they take to encode?
    1. VP9 encoder is slow, is the speed acceptable?
    2. AV1 is the slowest of all encoders, is the speed acceptable?

Encoders

Rather than use the native encoders which would make the test system specific, I've chosen to run as much as possible through FFMPEG; a cross-platform tool to work with video enconding and transcoding.

The only exception is AV1. For this encoder I've downloaded the source code and compiled the reference implementation since the codec is not officially supported by FFMPEG yet.

To run the encodes through FFMPEG in MacOS (High Sierra) we need to install it. I chose to install it via Homebrew with the following flags:

- **–head** (install from the HEAD of the Git Repository)
- **–with-tools** (enable additional FFmpeg tools)
- **–with-x265** (adds x265 support)
- **–with-libvpx** (addslibvpx support)
- **–with-opus** (adds support for the Opus audio codec)

The source

To generate an uncompressed YUV420 baseline of the video we'll use in the other

encodings I ran the following command:

```
filename=$1

ffmpeg -i ${filename} -r 24 -vf format=yuv420p \
${filename}-source.y4m
```

`filename=$1` assigns the first parameter to the variable filename. This is what allows the other parts of the script to use `${filename}` instead of hardcoding the name everywhere. With this little trick we can run multiple tests with different videos.

The first video we'll encode is X264. This will create an AVC version of the video. I've chosen to use the slow preset for both AVC and HEVC encodings. You may want to play with the `-preset` value to see if faster or slower presets still make a difference.

```
ffmpeg -i ${filename} \
-c:v libx264 -preset slow -crf 22 \
-c:a copy -b:a 9k \
${filename}-h264.mp4
```

The next encoding is for X265. This should produce a higher quality video at the same bitrate. I don't believe the difference is noticeable to the naked eye.

```
ffmpeg -i ${filename} \
-preset slow \
-c:v libx265 -crf 28 \
-c:a aac -b:a 44.1k \
${filename}-h265.mp4
```

VP9 is the current version of Google's VPx line of codecs (acquired when they purchased On2 Corporation). It takes longer to encode but it provides smaller files at the same bitrate.

```
ffmpeg -i ${filename} \
```

```
-c:v libvpx-vp9 \
-b:v 512K \
-c:a libopus -b:a 44.1k \
${filename}-vp9.webm
```

The final test is for AV1. This takes a really long time so it may work better if you leave the encode running over a long period of time (I've left mine running overnight).

```
aomenc  \
${filename} \
--passes=1 --pass=1 \
--fps=24/1 \
--end-usage=cq \
--target-bitrate=512  \
--width=640 --height=360 \
-o ${filename}-av1.webm
```

Results And Final Notes

I picked a short 5 minute clip to validate the concept and the scripts I'm using to run the comparisons and to create clips short enough that won't run afoul of Github's 100MB size limit. I know that using Git LFS would solve the problem but the free tier is limited in space and I don't want to pay Github for the next tier of storage.

The results for encoding the Footloose clip are somewhat surprising in the Source to h264 conversion. If the files are encoded using the same codec and settings, how do you explain the 800 KB difference?

I also wasn't expecting the VP video to be larger than h265. I think that's because the audio bitrate is 48Khz rather than the 44.1Khz used in the other formats. Need to adjust the scripts to run the same audio bitrate throughout.

Encoding results for Footloose clip

Format	File Size	Notes
h264 (high Profile)/AAC in MP4 Container	28.5 MB	Source for conversions
h264 (high Profile)/AAC in MP4 Container	27.7 MB	Cannot explain the size difference
HEVC/AAC in MP4 Container	13 MB	
VP9 (profile 0)/Opus in WebM container	24.1 MB	Default audio bitrate is higher than the ones chosen for x264 and x265. I'm still puzzled on the difference
AV1/Opus in WebM container		

In my local environment I'm testing encodes with Tears of Steel. I'm expecting the results of these tests to better represent what full movie encoding in each format would look like.

Encoding results for Tears of Steel movie

Format	File Size	Notes
H264 High Profile/AAC in MOV container	738.9 MB	Source File for all conversions
H264 High Profile/AAC in MP4 container	343.6 MB	
HEVC Main Profile/AAC in MP4 container	110 MB	
VP9 (profile 0)/Opus in WebM container	58.9 MB	
AV1/Opus in WebM container		

All our testing has been subjective. What we think looks best and uses the least amount of resources (bandwidth and file size) but it may not be an accurate assessment. There are *objective* quality measurement tools that may be good to use in addition to the subjective evaluation I've done here. Posible subject for a future post. :)

Playback is another issue I have yet to tackle for all formats. Unlike Bitmovin,

who uses a specific version of the Encoder and Decoder tied to specific Git commits, I have been unable to use VLC or the HTML5 video element to play back AV1 content. It shouldn't be long before major browsers (all members of the Alliance for Open Media) start implementing AV1 support (and in the case of Firefox to remove the restriction that goes along with Bitmovin's implementation).

The last item that will drive decision for me is encoding speed. VP9 and AV1 take significantly longer than x264 and x265 to create content so we'll have to decide if these slower speeds are worth the file size and bandwidth savings.

As it stands right now AV1 is not usable. The encoding takes too long for it to work effectively in any sort of compressionn pipeline. VP9 is slower than either x264 or x265 but the file size reduction is worth the slower encoding speeds for video on demand.

I'm evaluating the command and parameters I'm using with AV1 to see if there are ways to optimize it for faster encodings. I'm also looking at 2-pass encoding to see if it improves encoding speed.

Links and Resources

- Code comparisons
  - AV1, VP9, x265, x264 Video Codec Comparison - 562kb/s
  - AV1, VP9, x265, x264 Video Codec Comparison - 1Mb/s
  - AV1 beats x264 and libvpx-vp9 in practical use case (Facebook)
  - Multi-Codec DASH Dataset: An Evaluation of AV1, AVC, HEVC and VP9 (Bitmovin)
  - AV1 Beats VP9 and HEVC on Quality, if You've Got Time, says Moscow State (from streamingmedia.com)
- Encoding Guides
  - FFmpeg and VP9 Encoding Guide
  - FFmpeg and H.264 Encoding Guide
  - FFmpeg and H.265 Encoding Guide
  - CRF Guide (Constant Rate Factor in x264 and x265)
- DASH and HLS
  - What is MPEG-DASH? (2011)
  - MPEG DASH
  - Guidelines for Implementation: DASH-IF Interoperability Points
  - HTTP Live Streaming 2nd Edition
- Mozilla specific

        - DASH playback of AV1 video in Firefox
        - https://demo.bitmovin.com/public/firefox/av1/ (only works in Firefox Nightly)
        - AV1 Is Finally Here, but Intellectual Property Questions Remain
        - AV1 & Media Codecs
- Objective Video Quality Assessment
    - Objective Video Quality Assessment Methods: A Classification, Review, and Performance Comparison
    - Content-aware objective video quality assessment
- Miscelaneous
    - Understanding Rate Control Modes (x264, x265, vpx)
    - Commentary: It's Not AV1 vs. HEVC
    - The Future of Video Codecs: VP9, HEVC, AV1 from Streaming Media West 2017