Codec Comparison: 2-pass encoding

We've set up 1-pass encoding for all our target codecs and we have the resulting videos to compare. Does a 2-pass enconding changes file size and video quality?

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. Does a 2-pass encoding improve video quality?
2. How the storage requirements change between formats
    1. Does 2-pass encoding further reduce file size?
3. Does 2-pass encoding improve encoding speed?

To answer these questions I'll prepare a second round of videos using 2-pass encodings. In the 2-pass process each codec will run twice with slightly different settings to create the two pass video.

The econding scripts

The 2-pass encondings will, as much as possible, use default encoding strategies. The first pass will generate a log that the second pass will use, so don't delete the log files.

I've also created a dummy `null` file to replace `/dev/null`. For some reason Bash on MacOS wants to overwrite `/dev/null` and I don't think that's healthy for a Unix-based operating systems.

H264

```
# Encode x264
# Pass 1
ffmpeg -i $f \
-preset slow -crf 22 \
-c:v libx264 -b:v 512k \
-pass 1 \
-c:a aac -b:a 9600 \
-f mp4 null

# Pass 2
ffmpeg -i $f \
-c:v libx264 -b:v 512k \
-preset slow -crf 22 \
-pass 2 \
-c:a aac -b:a 9600 \
$f_2pass_h264.mp4
```

x265

```
# h265 Pass 1
ffmpeg -y -i $f \
-c:v libx265 -b:v 512k \
-x265-params pass=1 \
-c:a aac -b:a 9600 \
-f mp4 null

# h265 Pass 2
ffmpeg -i $f \
-c:v libx265 -b:v 512k \
-x265-params pass=2 \
-c:a aac -b:a 9600x \
$f-2pass-x265.mp4
```

VP9

```
# Pass 1
ffmpeg -i $f \
-c:v libvpx-vp9 -b:v 512K \
-pass 1 -c:a libopus \
-f webm null

# Pass 2
ffmpeg -i $f \
-c:v libvpx-vp9 -b:v 512K \
-pass 2 -c:a libopus \
$f_2pass_vp9.webm
```

AV1

```
# AV1 Pass 1
  aomenc \
    $f \
    -o $f-av1-2pass.webm \
    --width=640 \
    --height=360 \
    -p 2 \
    --pass=1 \
    --fpf=$f.fpf \
    --cpu-used=2 \
    --target-bitrate=512K \
    --auto-alt-ref=1 \
    -v \
    --minsection-pct=0 \
    --maxsection-pct=800 \
    --lag-in-frames=25 \
    --kf-min-dist=0 \
    --kf-max-dist=99999 \
    --static-thresh=0 \
    --min-q=0 \
    --max-q=63 \
```

```
    --drop-frame=0
# AV1 Pass 2
  aomenc \
    $f \
    $f-av1-2pass.webm  \
    --width=640 \
    --height=360 \
    -p 2 \
    --pass=2 \
    --fpf=$f.fpf \
    --cpu-used=2 \
    --target-bitrate=512K \
    --auto-alt-ref=1 \
    -v \
    --minsection-pct=0 \
    --maxsection-pct=800 \
    --lag-in-frames=25 \
    --kf-min-dist=0 \
    --kf-max-dist=99999 \
    --static-thresh=0 \
    --min-q=0 \
    --max-q=63 \
    --drop-frame=0 \
    --bias-pct=0 \
    --minsection-pct=0 \
    --maxsection-pct=800 \
    --psnr \
    --arnr-maxframes=7 \
    --arnr-strength=3
```

Analysis and Review

1 and 2 pass encoding comparison

Format	1 pass encode	2 pass encode
Footloose original file	28.5 MB	
Footloose x264 in MP4 container	27.7 MB	23.1 MB

Format	1 pass encode	2 pass encode
Footloose x265 in MP4 container	13 MB	23.2 MB
Footloose VP9 in WebM container	24.1 MB	22 MB
Tears of Steel Original File	738.9 MB	
Tears of Steel x264 in MP4 container	343.6 MB	56.1 MB
Tears of Steel x265 in MP4 container	110 MB	56.5 MB
Tears of Steel VP9 in WebM container	58.9 MB	55.6 MB

The differences between 1 and 2 pass encodings is puzling in some aspects. I looks like 2 pass encoding works better for larger files but not necessarily for smaller files.

Footloose presented an interesting encoding exercise. The two pass encoding didn't reduce the file size as muhch as I expected and, in the case of x265, it increaesed the file size and not reduce it.

Tears of Steal provided a more consistent result in the 2 pass encoding even for VP9, that presented a very small 1 pass encoding to begin with. However all of the 2 pass encodings offer massive side reductions without a perceptible, to me, loss of quality.

The larger file also shows significantly larger file size savings; from 739 down to 55.6 MB for VP9, 56.1 for x264 and 56.5 for x265.

Even though results for other files may vary.

For Footloose, I'd recommend using **x265 with a 1-pass encode**.

Larger files (dimensions and file size), however, benefit from a 2 pass encoding. In all formats the savings of **a 2 pass encoding** are significant enough to be worth the extra time it takes.

The examples I've used may or may not match the results. As with many things working on video I'd suggest you test all the formats and see what works best for your needs.

I have not included DASH video in these comparisons. A next step would be to take the x264 file and process it through the Shaka Packager as documented in

Revisiting video encoding: DASH.

Or maybe the video will be small enough to play locally or through Youutube or Vimeo. Your milleage may vary :)