Color Correction Performance, CD4 vs PD14

By
JL_JL (forum.cyberlink.com)

A fair warning, this write-up contains a fairly detailed look into color correction performance of PD14 and CD4 so for those that are simply glad they got a “Produced” file of their project timeline, this discussion probably not for you, a few others may have some interest to continue reading. The discussion has to do with the time it takes to encode a project after a color adjustment was made to significant timeline content duration. As many know, this can add significant time to the basic encode process.

Background: A while back I was given some very high quality 4K footage of significant duration (180min) that all needed some minor color correction done on the entire 180min. Even though the source video was 4K, the end state of the video editing project was to provide a basic ~120min BD, 1920x1080/60i 24Mbps disc for commonality of playback. What I noticed was the following:

1) If I apply the minor color correction with CD4 and then “Produce” a 1920x1080/60i 24Mbps H.264 file in PD14 this took xx hours to encode.

2) If I apply the same minor color correction with CD4 and then “Produce” a 1920x1080/60i 24Mbps H.264 file from within CD4, which I could then just import to PD14 and use as my editing “color corrected” source file for future edits, this took yy hours to encode in CD4.

3) Oddly, the ratio of xx/yy was like 4 to 1. Basically, it was 4 times faster to have CD4 “Produce” the color corrected file vs PD14 doing what appeared to be the same color correction task. Somewhat significant when I had to process the ~180min of video source files. So the obvious question, WHY?

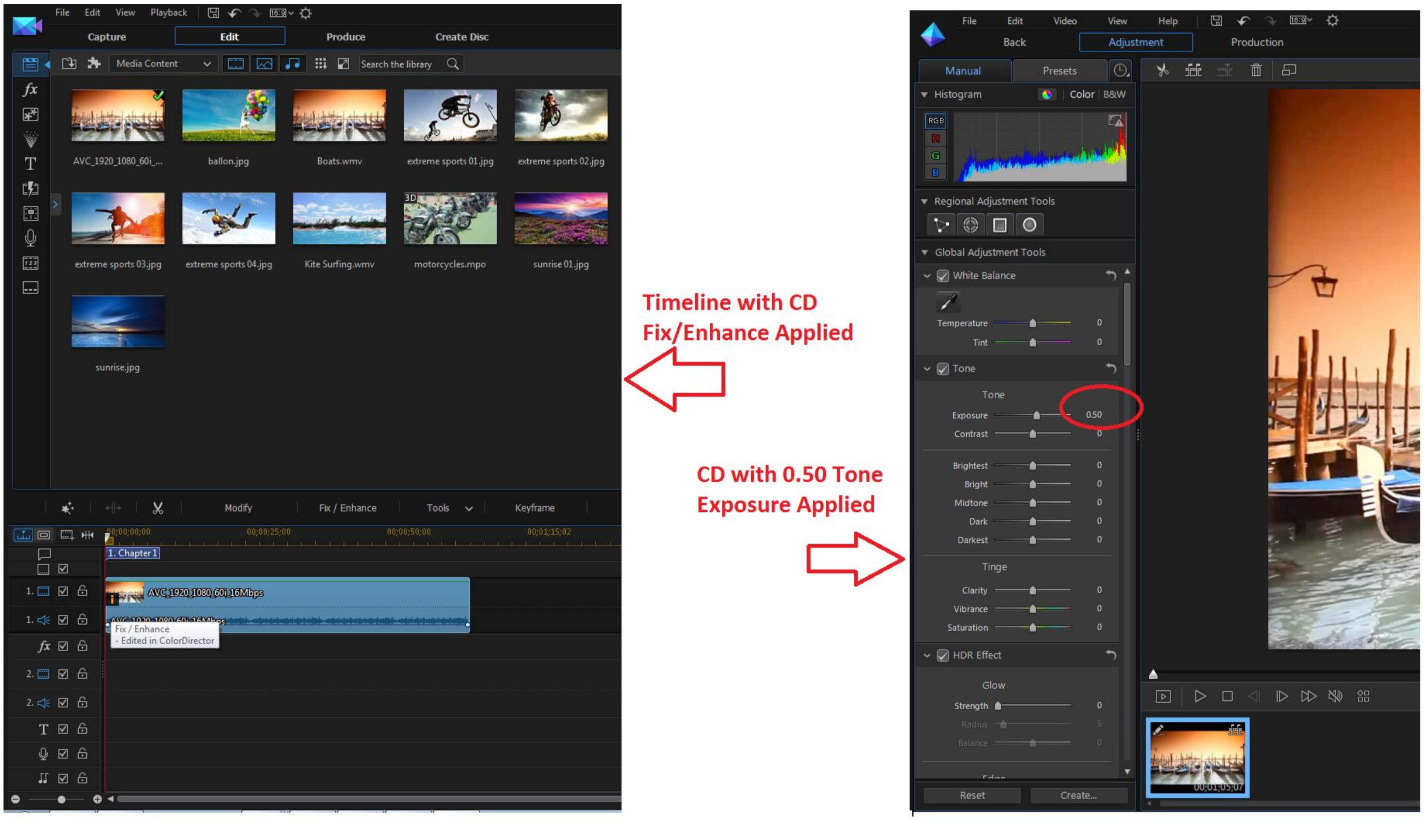
To answer the why and document the findings, I created a few “source” files to work with. I used the basic boats.wmv (PD13 and here: http://forum.cyberlink.com/forum/posts/list/25/40930.page) from CL so others could experiment with the exact same files and observations if so inclined. My source footage was simply 5 of the boats.wmv in the timeline and “Produced” to create various source files with standard PD14 profiles in a H.264 m2ts container. Table 1 below outlines the initial source files to work with and some pertinent video file information to validate that they are what the title says they are.

Table 1: Created Source Video Files

Source Footage Filename	Filesize MB	Video Bitrate Mbps	Frame Size	Duration (Ssec)
AVC_1920_1080_60i_16Mbps	128	15.5	1920 x 1080	65
AVC_1920_1080_60i_24Mbps	183	22.2	1920 x 1080	65
AVC_2048_1080_60i_40Mbps	324	39.4	2048 x 1080	65
AVC_2048_1080_60p_40Mbps	325	39.6	2048 x 1080	65
AVC_3840_2160_60i_50Mbps	396	48.3	3840 x 2160	65
AVC_3840_2160_60p_50Mbps	396	48.3	3840 x 2160	65
AVC_4096_2160_60i_50Mbps	397	48.3	4096 x 2160	65
AVC_4096_2160_60p_50Mbps	397	48.4	4096 x 2160	65

So the first task was to simply “Produce” these source files per methods discussed in items 1 & 2 above and monitor a few performance stats to get a feel for what PD14 and CD4 might be doing differently. Figure 1 shows my typical timeline with a simple 0.50 exposure adjustment applied in CD4.

Figure 1: Typical Timeline and CD4 Exposure Adjustment



Fix / Enhance
- Edited in ColorDirector

☑ White Balance

Temperature 0

Tint 0

☑ Tone

Tone

Exposure 0.50

Contrast 0

Brightest 0

Bright 0

Midtone 0

Dark 0

Darkest 0

Tinge

Clarity 0

Vibrance 0

Saturation 0

☑ HDR Effect

Glow

Strength 0

Radius 5

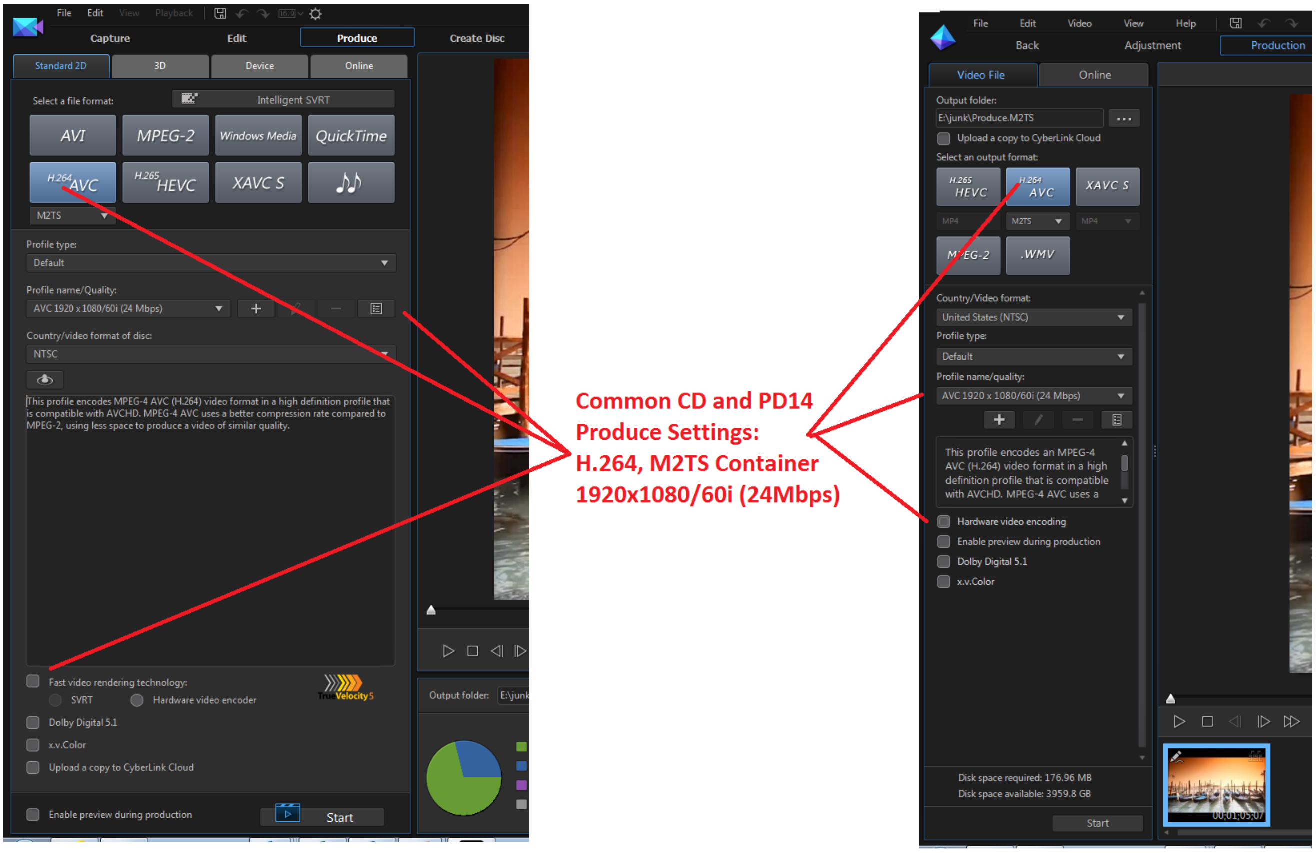
Balance 0

Timeline with CD Fix/Enhance Applied

CD with 0.50 Tone Exposure Applied

Figure 2 shows the typical "Produce" settings for CD4 and PD14. Both set to a basic H.264, 1920x1080/60i 24Mbps m2ts container with no hardware encoding, so a basic CPU encoding task for both CL products.

Figure 2: Typical CD4 and PD14 "Produce" Settings



Select a file format:

M2TS

Profile type:
Default

Profile name/Quality:
AVC 1920 x 1080/60i (24 Mbps)

Country/video format of disc:
NTSC

This profile encodes MPEG-4 AVC (H.264) video format in a high definition profile that is compatible with AVCHD. MPEG-4 AVC uses a better compression rate compared to MPEG-2, using less space to produce a video of similar quality.

□ Fast video rendering technology:
○ SVRT ○ Hardware video encoder

□ Dolby Digital 5.1

□ x.v.Color

□ Upload a copy to CyberLink Cloud

□ Enable preview during production

Output folder: E:\junk

Output folder:
E:\junk\Produce.M2TS

□ Upload a copy to CyberLink Cloud

Select an output format:

M2TS

Country/Video format:
United States (NTSC)

Profile type:
Default

Profile name/quality:
AVC 1920 x 1080/60i (24 Mbps)

This profile encodes an MPEG-4 AVC (H.264) video format in a high definition profile that is compatible with AVCHD. MPEG-4 AVC uses a

□ Hardware video encoding

□ Enable preview during production

□ Dolby Digital 5.1

□ x.v.Color

Disk space required: 176.96 MB
Disk space available: 3959.8 GB

Common CD and PD14 Produce Settings:
H.264, M2TS Container
1920x1080/60i (24Mbps)

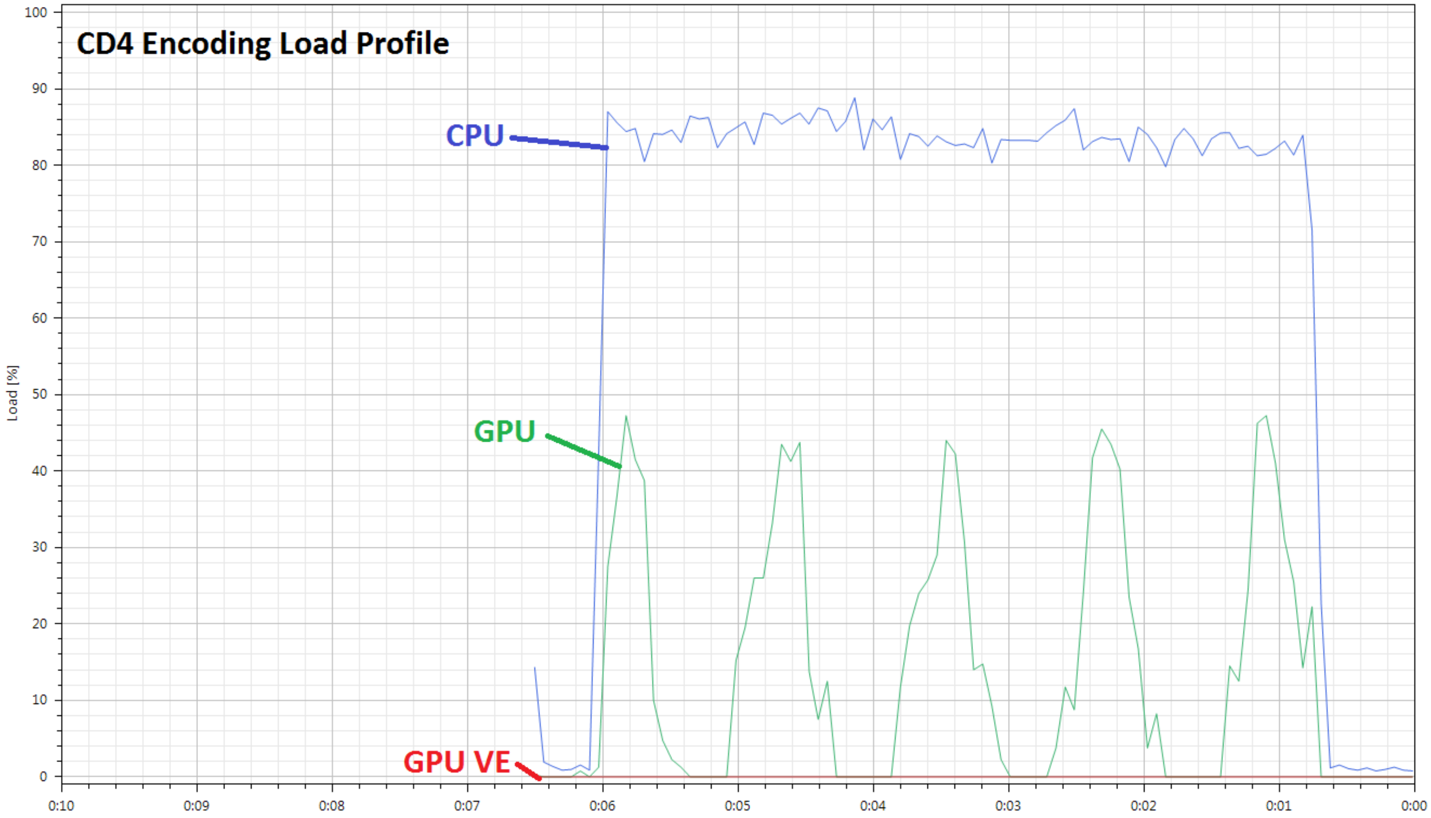
Table 2 shows the results with some other pertinent file information to make sure everything is proper. As seen by the PD14/CD4 ratio in the table, CD4, when dealing with high frame resolution content is significantly faster to apply the color correction within and "Produce" my output file specification vs PD14. All very consistent with my real footage observations and my initial question of why.

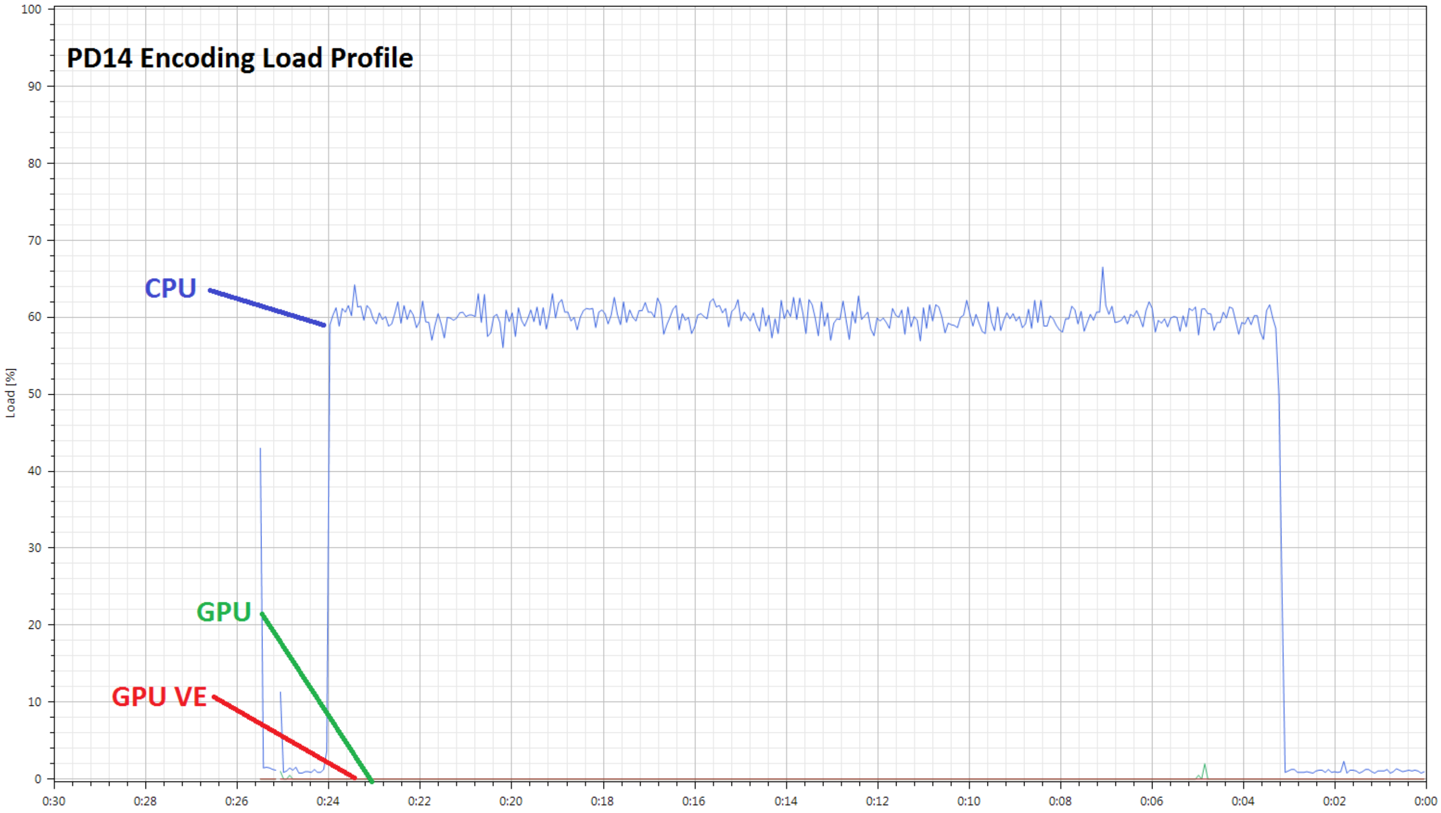
Table 2: Color Corrections Performance Results

Source Video Format	PD14 Encode Time	GPU Load %	GPU VE Load %	CPU Load %	Size MB	Produced Video Bitrate, Mbps	PD14/CD4 Encoding Time Ratio	CD4 Encode Time	GPU Load %	GPU VE Load %	CPU Load %	Size MB	Produced Video Bitrate, Mbps
AVC_1920_1080_60i_16Mbps	336	0	6	62	185	22.4	1.17	286	48	8	74	186	22.5
AVC_1920_1080_60i_24Mbps	338	0	6	62	185	22.4	1.18	286	46	8	74	186	22.5
AVC_2048_1080_60i_40Mbps	357	0	0	64	184	22.3	1.20	298	46	0	78	185	22.4
AVC_2048_1080_60p_40Mbps	359	0	0	66	184	22.3	1.21	296	44	0	78	184	22.3
AVC_3840_2160_60i_50Mbps	1200	0	0	60	185	22.4	3.72	323	44	0	82	186	22.5
AVC_3840_2160_60p_50Mbps	1173	0	0	60	185	22.4	3.64	322	44	0	84	185	22.5
AVC_4096_2160_60i_50Mbps	1278	0	0	60	185	22.4	3.99	320	44	0	84	185	22.5
AVC_4096_2160_60p_50Mbps	1245	0	0	60	184	22.3	3.94	316	44	0	84	185	22.4

A detailed look at some hardware loadings during this CD4 and PD14 encode process is shown in Figure 3 which can start to shed some insight.

Figure 3: Example CPU, GPU and GPU VE Loads





As seen in Figure 3, a few unusual observations that deserve some discussion. These color corrections need to be done on individual frames prior to encoding and currently with PD14 this is predominately a CPU function and here we are also doing CPU encoding so would expect a high CPU usage. The GPU VE (hardware encoding) is not being utilized so no GPU VE load would be expected. Even if the hardware encoder was utilized, the encode time with the color correction will nearly be the same as the process is governed by the CPU applying the color correction, not the actual encoding.

However, it appears that CD4 is probably using some aspect of OpenCL to assist in the actual encode process because the basic GPU does show significant periodic loading which would handle the OpenCL features. Although I've seen nothing published from CL about CD4 using OpenCL, and no mention of the word in the docs, it appears obvious that they are from the unique GPU loadings when CD4 is doing the encoding. Maybe something similar to the CL OpenCL capability added to PhotoDirector to speed up image processing and presented by CL here, http://www.cyberlink.com/prog/product/html/27047/Ultra/new.jsp .

As was shown in Table 2, this GPU load is always about the same for CD4 encoding and maybe accounts for ~15% performance improvement when both source video and output video are the same format, but surely it is not responsible for the extremely fast performance of CD4 with high frame resolution source files.

My hunch is that CD4 and PD14 perform the color correction in two vastly different approaches. I think what occurs is this:

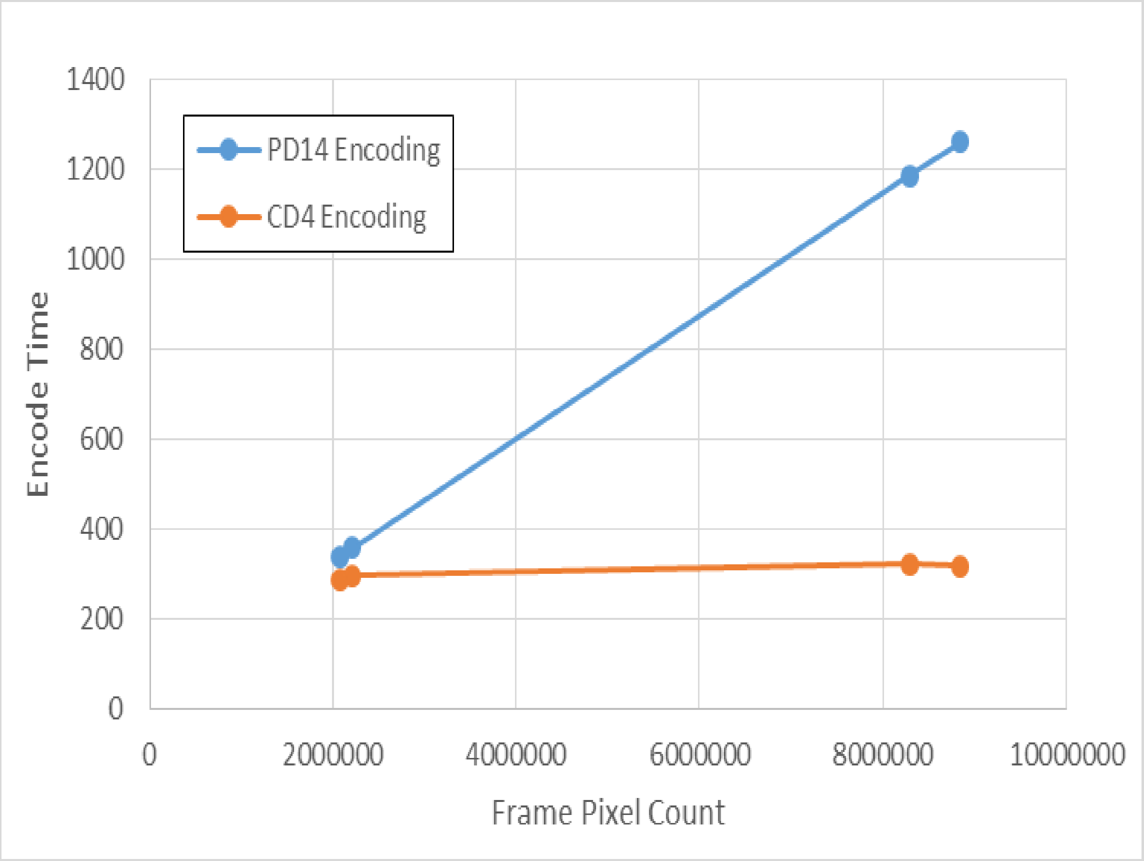
1) PD14 applies the color correction to the source raw frame size prior to downscaling to the output specification of 1920x1080. So as I work with larger source video frames, say 4096x2160, ~8.8M pixels, my processing time for the color correction on this frame goes higher resulting in the significantly longer encode times. Color correction processing time should essentially scale and be directly proportional to frame size pixel density.

2) However, CD4 does things somewhat opposite, first it reduces the source frame size to the target output frame size and then performs the color correction on this reduced frame. This would lead to the near constant encode

performance for CD4 presented in Table 2 as my output target is always identical size, namely, 1920x1080, ~2M pixels.

These two hunches are shown in Figure 4 which plots a frame pixel count on the horizontal axis and a encode time on the vertical axis. The PD14 line shows a linear increase in encode time with the size of the source frame pixel count which correlates to item 1 observation. Keep in mind this linear line is based on two frame pixel counts low, and two high. Simply governed by the basic profile options. The CD4 line shows nearly constant encode time, (horizontal line) even though pixel count of source frame changes dramatically which correlates to the item 2 observation.

Figure 4: Frame Pixel Count vs Encode Time

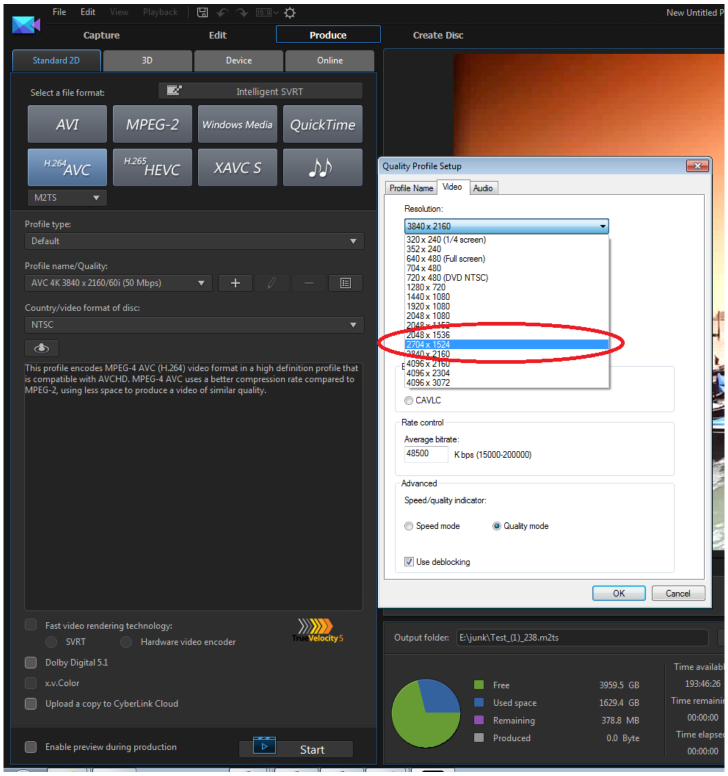


Attempt to prove/substantiate these two hunches:

1. If I create a somewhat nonstandard intermediate frame size, say something around 4M pixels, will the encode time take ~600 seconds as the massive interpolation of the PD14 line characteristics indicate it should based on Figure 4? This would imply my number 1 hunch was correct if it does.
2. If I set my output encode format in CD4 to the same frame size as the source, then no color adjustments are being made on a smaller output frame pixel size and my CD4 encode times should increase dramatically and essentially mimic PD14 encode times (minus the OpenCL benefit of ~15% discussed earlier and the higher encode task). CD4 is now essentially working with color correction being applied on the full frame pixel count like PD14 is doing hence computationally more expensive. This would imply my number 2 hunch was correct.

So for proof of item 1 above I used my 5 boats.wmv in the timeline and created a custom profile based on the standard 3840 x 2160/60i 50Mbps profile and only modified the frame size to 2704 x 1524, this produces the desired 4.12M pixel frame size. The only modification to the standard profile is shown in Figure 5.

Figure 5: Intermediate Frame Size Encoding Settings



Select a file format: H.264 AVC

M2TS

Profile type: Default

Profile name/Quality: AVC 4K 3840 x 2160/60i (50 Mbps)

Country/video format of disc: NTSC

This profile encodes MPEG-4 AVC (H.264) video format in a high definition profile that is compatible with AVCHD. MPEG-4 AVC uses a better compression rate compared to MPEG-2, using less space to produce a video of similar quality.

□ Fast video rendering technology:

○ SVRT ○ Hardware video encoder

□ Dolby Digital 5.1

□ x.v.Color

□ Upload a copy to CyberLink Cloud

□ Enable preview during production

Quality Profile Setup

Profile Name | Video | Audio

Resolution: 3840 x 2160

320 x 240 (1/4 screen)
352 x 240
640 x 480 (Full screen)
704 x 480
720 x 480 (DVD NTSC)
1280 x 720
1440 x 1080
1920 x 1080
2048 x 1080
2048 x 1152
2048 x 1536
2704 x 1524
3840 x 2160
4096 x 2160
4096 x 2304
4096 x 3072

○ CAVLC

Rate control

Average bitrate: 48500 K bps (15000-200000)

Advanced

Speed/quality indicator:

○ Speed mode ● Quality mode

☑ Use deblocking

Output folder: E:\junk\Test_(1)_238.m2ts

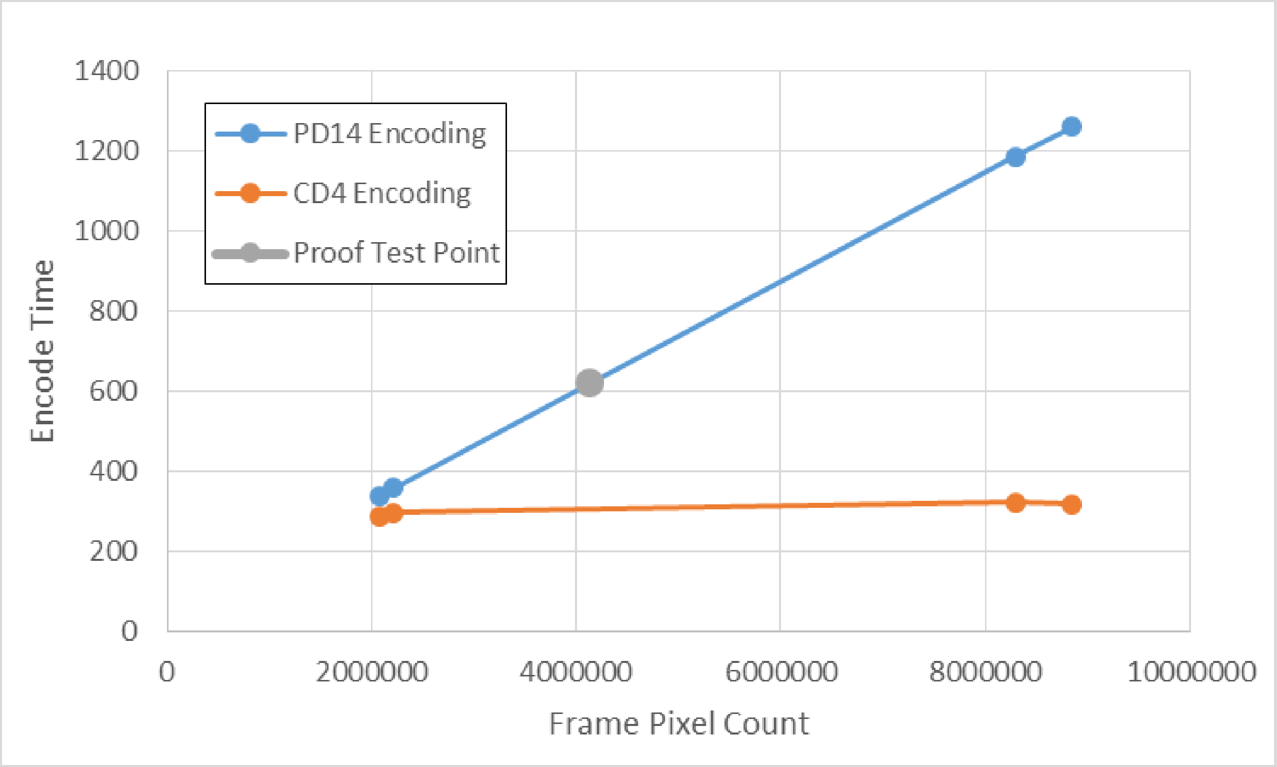
🟩 Free	3959.5	GB
🟦 Used space	1629.4	GB
🟪 Remaining	378.8	MB
⬜ Produced	0.0	Byte

Time availabl 193:46:26

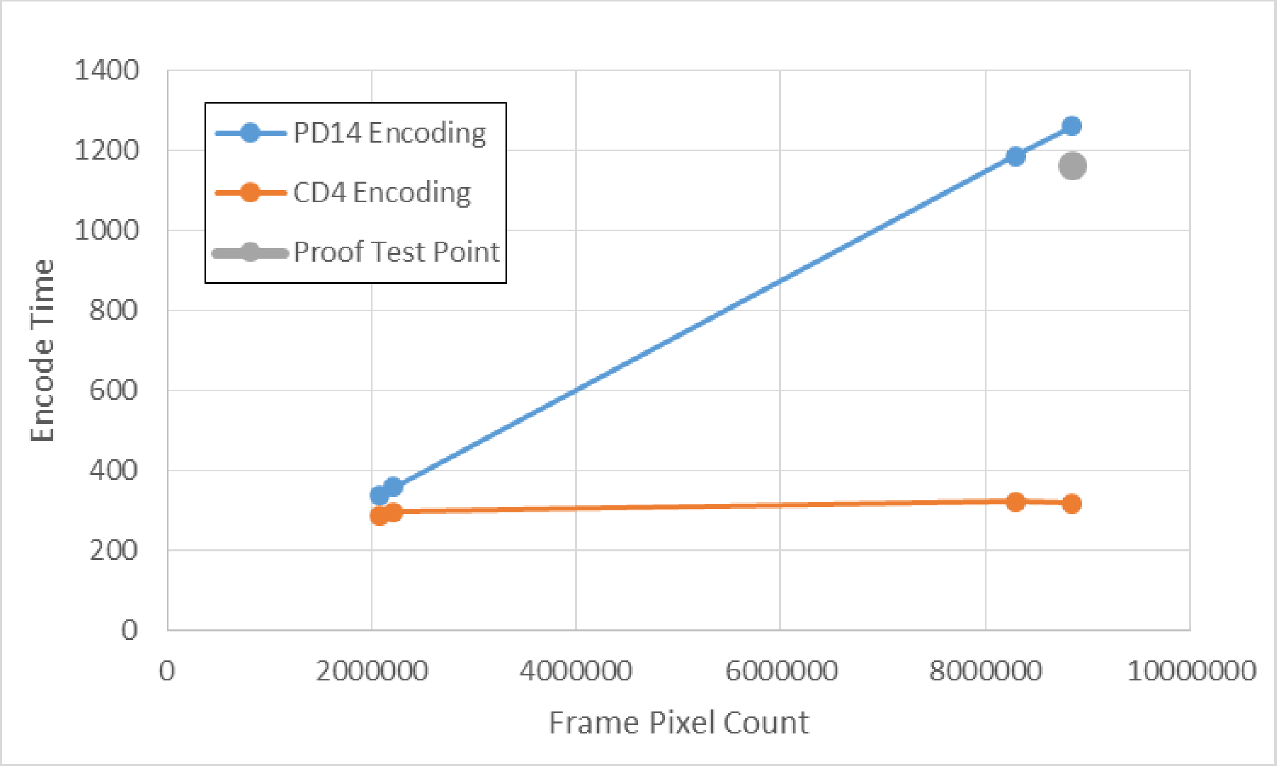
Time remainin 00:00:00

Time elapsed 00:00:00

The result data point of the encode time with the same color correction as previous with PD14 is shown below, a near perfect match on the linear relationship at a 4.12M pixel frame count.



For proof into item 2 above, I used the 4096 x 2160/60i 50Mbps source video I had created and outputted the same profile within CD4 with my applied color correction. The resulting data point of encode time is shown below, a near perfect match to what was surmised, namely, it would mimic PD14 slower performance but be about ~15% faster because of the use of OpenCL in CD4.



Conclusions: I think this confirms why color corrections in CD4 appear to perform so much better than PD14 if one has high frame resolution source but desires a down scaled output resolution. So which is better, I'm not 100% certain. In discussion with some digital photo enthusiasts, they indicate they would only color correct on a source picture frame size and then down scale to a desired output quality depending on end printing need. This would indicate one should not perform encode color correction within CD4 unless source and desired output are of the same frame size. As such, the performance gain with CD4 is negligible, ~15% because of perhaps OpenCL usage. One the other hand, if one needs to do this type of activity and has done a trial and is happy with the produced file and quality of a CD4 encoded file, it can shave hours off a long color correction project. Yes, all this rather trivial for those posting a 5 min flick to YouTube, however, many other editors produce much larger projects. For me, color correcting this 180min of 4K source footage in CD4 to desired video output profile would appear to reduce the produce time from nearly 60hrs to just 14hr for the color correction. On the other hand, I could have simply pre produced my 4K source to 1920x1080 profile in PD14 and used that for editing this project and all is equal, no gains. Therefore, I see no reason why this different approach in CD4.

So why does CL have two vastly different approaches to applying color corrections PD14 vs CD4, I have not a clue, not even a hunch, and if you have made it this far reading, you probably don't want to hear my honest opinion. Hope you gleaned at least something.

Jeff (JL_JL on forum.cyberlink.com)