How to Produce x264 and x265 Video at Maximum Quality and Maximum Efficiency

Jan Ozer

Marketing

NETINT

jan.ozer@netint.com

Gratitudes

- You the audience - thanks for coming
- Dan Rayburn - conference organizer - thanks for inviting me
- NETINT - thanks for letting me continue to pursue my passions (even when it doesn't help sell ASIC-based streaming transcoders)

Who this Presentation is For

Before lasts night's Streaming Summit Cocktail Party	Intermediate to advanced	
Ran into Alex Giladi/Dan Grois from Comcast	Novice to intermediate	
Ran into David Ronca (Meta) and Anne Aaron (Netflix)	Rank beginner to novice	
Quick call to my therapist	Novice to intermediate	

Who this Presentation is For

- Topics
  - Encoding quality - intermediate - some fun thoughts on presets and production efficiencies
  - Encoding production - AWS/etc - novice to beginner
    - Again, perhaps some interesting observations but few revelations for those already extensively producing in the cloud

What to do With This Information?

- What's the only answer that's always correct when it comes to encoding
  - It depends
  - So,
    - What's the best codec - it depends
    - What's the best encoder - it depends
    - What's the best bitrate control mechanism - it depends

- Take what I present as a map to guide your own research
  - Testing with 2-8 of my own test files could yield interesting results, but could be totally irrelevant to your scenario
- My focus - VOD (not live)
  - x264 1080p 30
  - x265 1080p 30 8-bit
  - x276 4K60 10-bit

Agenda

- x264
  - Preset
  - Reference frames
  - B-frames
  - Bitrate control
  - Best AWS CPU
  - Optimal core count
    - AWS
    - Desktop
- x265 - HD
  - Preset
  - Reference frames
  - B-frames
  - Best AWS CPU
  - Optimal core count
    - AWS
    - Desktop
- x265 - 4K
  - Preset
  - Best AWS CPU
  - Optimal core count
    - AWS
    - Desktop
- Bonus content
  - What I learned at AWS

Presets

- Overview
- Presets and quality
- Presets and bandwidth
- Computing breakeven

Exploring Presets

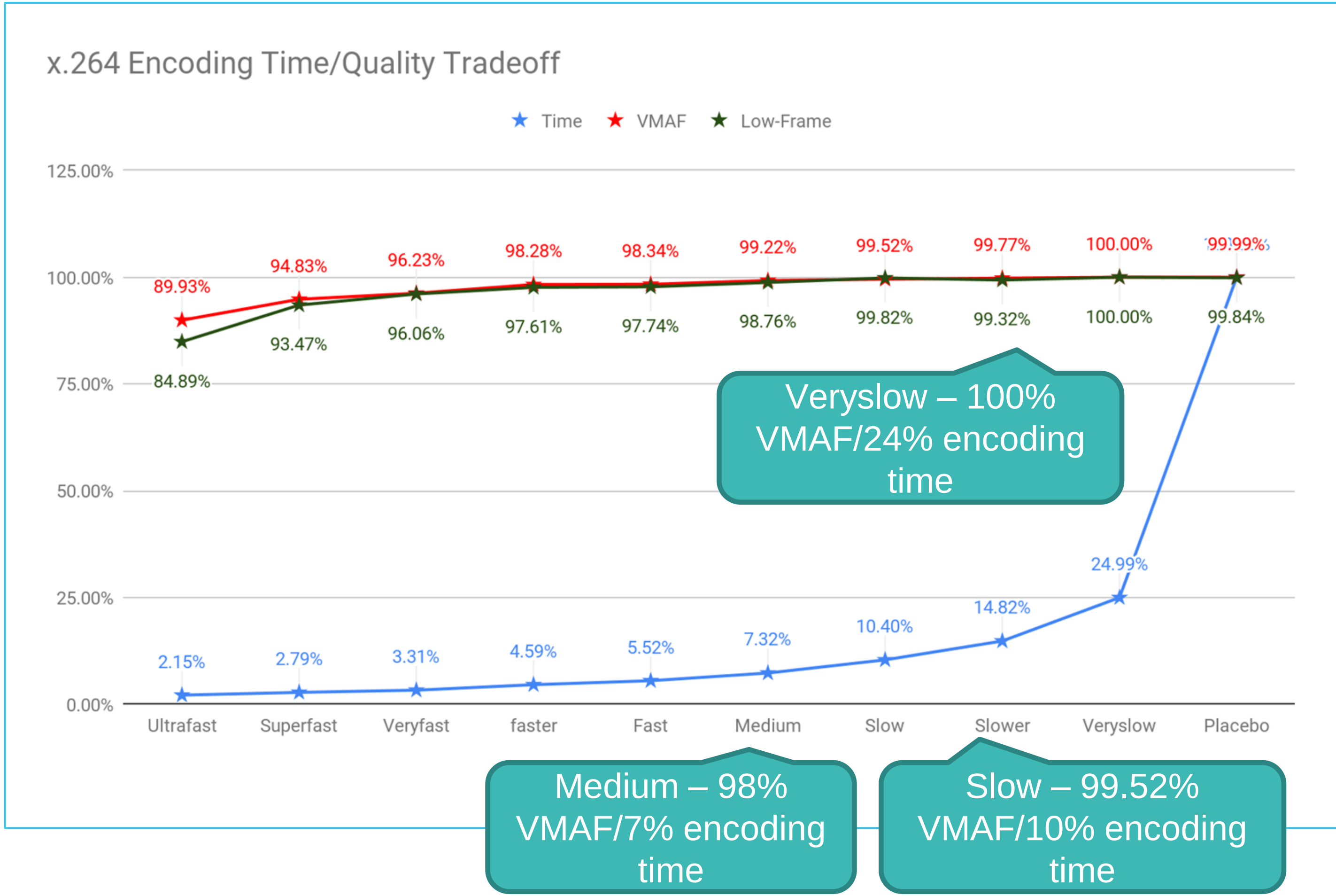
- What does the preset do?
  - Adjusts parameters to producers can choose desired quality/encoding time tradeoff
    - 10 presets - ultrafast to placebo
- Quiz: What’s our favorite x264 preset?
- Does the preset control distribution quality?
    - Yes?
    - No?

NETINT

Preset Role

- Controls encoding time
- Most producers:
  - Choose quality level (VMAF 93-95/PSNR 45) and encode to match that quality level
- If lower quality preset doesn’t achieve target quality, you boost the bitrate
  - So, preset doesn’t control ***quality***, it controls encoding ***cost***
  - Choosing a preset is ***always*** a tradeoff between encoding cost and bandwidth cost

- Two files
- Measure encoding time
- Harmonic mean VMAF
- Low-frame VMAF
- Preset and % of maximum time/score
- What's the best preset?



- Two files
- Measure enoding time
- Harmonic mean VMAF
- Low-frame VMAF
- Preset and % of maximum time/score
- Medium
- Slow

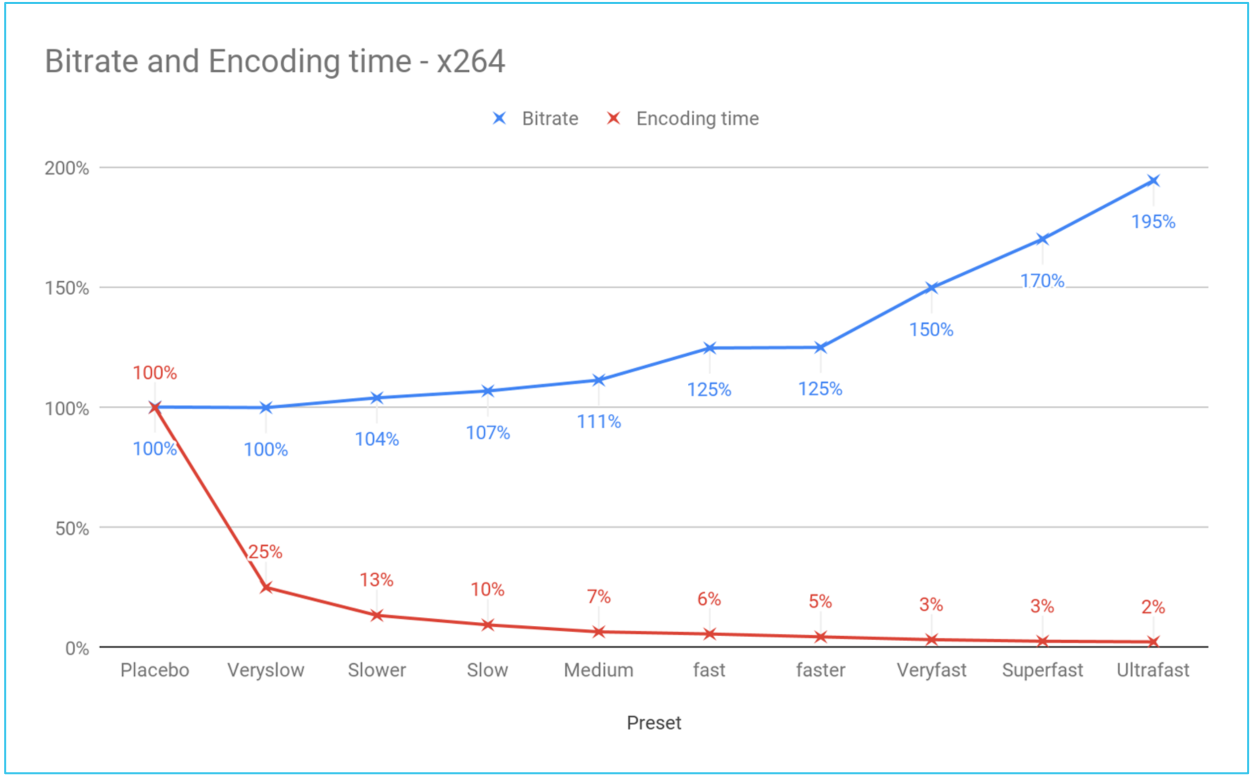
Not a suggestion but a recommendation:

- Adopt to your own test files (animation, training, sports, whatever)
- Test reasonable number of files
- Do the math
- Come to your own conclusions

Next Question

How much do you have to boost the bitrate to match 100% quality?

H.264 Preset



Preset	Bitrate	Encoding time
Ultrafast	195%	2%
Superfast	170%	3%
Veryfast	150%	3%
faster	125%	5%
fast	125%	6%
Medium	111%	7%
Slow	107%	10%
Slower	104%	13%
Veryslow	100%	25%
Placebo	100%	100%

x264 - Viewer Count Breakeven - $0.08/GB

At higher bandwidth costs, saving bandwidth matters more than encoding costs.

Will change with each scenario

		Bitrate		4000						
		MMBytes per hour		1800		Cost per GB		0.08		
		Encode/hr		0.62						
Preset	Encode	Bandwidth			50	100	250	500	1000	5000
Ultrafast	$0.06	3.50	$0.28		$14	$28	$70	$140	$280	$1,401
Superfast	$0.07	3.06	$0.25		$12	$25	$61	$123	$245	$1,226
Veryfast	$0.08	2.70	$0.22		$11	$22	$54	$108	$216	$1,080
faster	$0.11	2.25	$0.18		$9	$18	$45	$90	$180	$901
fast	$0.14	2.25	$0.18		$9	$18	$45	$90	$180	$899
Medium	$0.16	2.01	$0.16		$8	$16	$40	$80	$161	$803
Slow	$0.23	1.92	$0.15		$8	$16	$39	$77	$154	$770
Slower	$0.33	1.87	$0.15		$8	$15	$38	$75	$150	$750
Veryslow	$0.62	1.80	$0.14		$8	$15	$37	$73	$145	$721
Placebo	$2.47	1.80	$0.14		$10	$17	$39	$75	$147	$724

x264 - Viewer Count Breakeven - $0.04/GB

		Bitrate		**4000**						
		MBytes per hour		**1800**		**Cost per GB**		**0.04**		
		Encode/hr		**0.62**						
Preset	**Encode**	**Bandwidth**			**50**	**100**	**250**	**500**	**1000**	**5000**
Ultrafast	$0.06	3.50	$0.14		$7	$14	$35	$70	$140	$701
Superfast	$0.07	3.06	$0.12		$6	$12	$31	$61	$123	$613
Veryfast	$0.08	2.70	$0.11		$5	$11	$27	$54	$108	$540
faster	$0.11	2.25	$0.09		$5	$9	$23	$45	$90	$450
fast	$0.14	2.25	$0.09		$5	$9	$23	$45	$90	$450
Medium	$0.16	2.01	$0.08		$4	$8	$20	$40	$80	$401
Slow	$0.23	1.92	$0.08		$4	$8	$19	$39	$77	$385
Slower	$0.33	1.87	$0.07		$4	$8	$19	$38	$75	$375
Veryslow	$0.62	1.80	$0.07		$4	$8	$19	$37	$73	$361
Placebo	$2.47	1.80	$0.07		$6	$10	$21	$39	$75	$363

x264 - Viewer Count Breakeven - $0.02/GB

As bandwidth costs drop, encoding cost matters longer (but still not that long)

		Bitrate		4000						
		MBytes per hour		1800		Cost per GB		0.02		
		Encode/hr		0.62						
Preset	Encode	Bandwidth			50	100	250	500	1000	5000
Ultrafast	$0.06	3.50	$0.07		$4	$7	$18	$35	$70	$350
Superfast	$0.07	3.06	$0.06		$3	$6	$15	$31	$61	$306
Veryfast	$0.08	2.70	$0.05		$3	$5	$14	$27	$54	$270
faster	$0.11	2.25	$0.05		$2	$5	$11	$23	$45	$225
fast	$0.14	2.25	$0.04		$2	$5	$11	$23	$45	$225
Medium	$0.16	2.01	$0.04		$2	$4	$10	$20	$40	$201
Slow	$0.23	1.92	$0.04		$2	$4	$10	$19	$39	$193
Slower	$0.33	1.87	$0.04		$2	$4	$10	$19	$38	$188
Veryslow	$0.62	1.80	$0.04		$2	$4	$10	$19	$37	$181
Placebo	$2.47	1.80	$0.04		$4	$6	$11	$21	$39	$183

x264 - Viewer Count Breakeven - $0.02/GB

As bandwidth costs drop, encoding cost matters longer (but still not that long)

Default

		Bitrate		4000						
		MBytes per hour		1800		Cost per GB		0.02		
		Encode/hr		0.62						
Preset	**Encode**	**Bandwidth**			**50**	**100**	**250**	**500**	**1000**	**5000**
Ultrafast	$0.06	3.50	$0.07		$4	$7	$18	$35	$70	$350
Superfast	$0.07	3.06	$0.06		$3	$6	$15	$31	$61	$306
Veryfast	$0.08	2.70	$0.05		$3	$5	$14	$27	$54	$270
faster	$0.11	2.25	$0.05		$2	$5	$11	$23	$45	$225
fast	$0.14	2.25	$0.04		$2	$5	$11	$23	$45	$225
Medium	$0.16	2.01	$0.04		$2	$4	$10	$20	$40	$201
Slow	$0.23	1.92	$0.04		$2	$4	$10	$19	$39	$193
Slower	$0.33	1.87	$0.04		$2	$4	$10	$19	$38	$188
Veryslow	$0.62	1.80	$0.04		$2	$4	$10	$19	$37	$181
Placebo	$2.47	1.80	$0.04		$4	$6	$11	$21	$39	$183

Reference Frames

Writing library : x264 core 164 r3106 eaa68fa

Encoding settings : cabac=1 / ref=16 / deblock=1:0:0 / analyse=0x3:0x133 / me=umh / subme=10 / psy=1 / psy_rd=1.00:0.00 / mixed_ref=1 / me_range=24 / chroma_me=1 / trellis=2 / 8x8dct=1 / cqm=0 / deadzone=21,11 / fast_pskip=1 / chroma_qp_offset=-2 / threads=34 / lookahead_threads=5 / sliced_threads=0 / nr=0 / decimate=1 / interlaced=0 / bluray_compat=0 / constrained_intra=0 / bframes=8 / b_pyramid=2 / b_adapt=2 / b_bias=0 / direct=3 / weightb=1 / open_gop=0 / weightp=2 / keyint=60 / keyint_min=31 / scenecut=0 / intra_refresh=0 / rc_lookahead=60 / rc=2pass / mbtree=1 / bitrate=4200 / ratetol=1.0 / qcomp=0.60 / qpmin=0 / qpmax=69 / qpstep=4 / cplxblur=20.0 / qblur=0.5 / vbv_maxrate=8400 / vbv_bufsize=8400 / nal_hrd=none / filler=0 / ip_ratio=1.40 / aq=1:1.00

- Veryslow preset use 16 reference frames
    - How much encoding time does this take?
    - How much quality do they add?
    - Is it worth it/

Reference Frames

	Baseline	Ref=1	Ref=2	Ref=4	Ref=8	Ref=10	Ref=12	Delta
Freedom	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0:00:28	33.33%
Football	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0:00:44	36.00%
Average	0:00:40	0:00:26	0:00:27	0:00:27	0:00:33	0:00:34	0:00:36	35.00%
Bitrate	**Baseline**	**Ref=1**	**Ref=2**	**Ref=4**	**Ref=8**	**Ref=10**	**Ref=12**	**Delta**
Freedom	3,713	3,715	3713	3,713	3,713	3,715	3,713	0.05%
Football	4,135	4,141	4135	4,131	4,131	4,134	4,128	0.31%
Average	**3,924**	**3,928**	**3,924**	**3,922**	**3,922**	**3,925**	**3,921**	**0.19%**
VMAF	[illegible]	[illegible]	[illegible]	[illegible]	**Ref=8**	**Ref=10**	**Ref=12**	**Delta**
Freedom	[illegible]	[illegible]	[illegible]	[illegible]	93.89	93.92	93.90	0.20%
Football	[illegible]	[illegible]	[illegible]	[illegible]	93.95	93.99	94.00	0.18%
Average	[illegible]	[illegible]	[illegible]	[illegible]	**93.92**	**93.96**	**93.95**	**0.17%**
Low Frame	[illegible]	[illegible]	[illegible]	[illegible]	**Ref=8**	**Ref=10**	**Ref=12**	**Delta**
Freedom	[illegible]	[illegible]	[illegible]	[illegible]	89.71	89.59	89.22	0.62%
Football	[illegible]	[illegible]	[illegible]	[illegible]	83.64	83.42	83.66	0.76%
Average	[illegible]	[illegible]	[illegible]	[illegible]	**86.67**	**86.50**	**86.44**	**0.39%**
Standard Deviation	**Baseline**	**Ref=1**	**Ref=2**	**Ref=4**	**Ref=8**	**Ref=10**	**Ref=12**	**Delta**
Freedom	2.19	2.21	2.20	2.19	2.19	2.19	2.16	1.97%
Football	3.51	3.58	3.56	3.53	3.51		3.50	2.30%
Average	**2.85**	**2.89**	**2.88**	**2.86**	**2.85**	**2.19**	**2.83**	**24.20%**

My analysis: encode 2 files to range of paremters

Freedom - concert video
Football - Harmonic test clip
These are 1080p30

Draw some initial conclusions about impact of encoding parameters.

Encoding time

Verify bitrate

Overall VMAF

Low Frame VMA

VMAF Std. Dev.

Reference Frames

	Baseline	Ref=1	Ref=2	Ref=4	Ref=8	Ref=10	Ref=12	Delta
Freedom	0:00:30	0:00:20	0:00:20	0:00:20	0:00:26	0:00:26	0:00:28	33.33%
Football	0:00:50	0:00:32	0:00:34	0:00:34	0:00:40	0:00:42	0:00:44	36.00%
Average	0:00:40	0:00:26	0:00:27	0:00:27	0:00:33	0:00:34	0:00:36	35.00%
Bitrate	**Baseline**	**Ref=1**	**Ref=2**	**Ref=4**	**Ref=8**	**Ref=10**	**Ref=12**	**Delta**
Freedom	3,713	3,715	3713	3,713	3,713	3,715	3,713	0.05%
Football	4,135	4,141	4135	4,131	4,131	4,134	4,128	0.31%
Average	**3,924**	**3,928**	**3,924**	**3,922**	**3,922**	**3,925**	**3,921**	**0.19%**
VMAF	**Baseline**	**Ref=1**	**Ref=2**	**Ref=4**	**Ref=8**	**Ref=10**	**Ref=12**	**Delta**
Freedom	93.90	93.74	93.80	93.84	93.89	93.92	93.90	0.20%
Football	94.00	93.86	93.84	93.88	93.95	93.99	94.00	0.18%
Average	**93.95**	**93.80**	**93.82**	**93.86**	**93.92**	**93.96**	**93.95**	**0.17%**
Low Frame	**Baseline**	**Ref=1**	**Ref=2**	**Ref=4**	**Ref=8**	**Ref=10**	**Ref=12**	**Delta**
Freedom	89.36	89.78	89.54	89.77	89.71	89.59	89.22	0.62%
Football	83.79	83.16	83.59	83.79	83.64	83.42	83.66	0.76%
Average	**86.58**	**86.47**	**86.56**	**86.78**	**86.67**	**86.50**	**86.44**	**0.39%**
Standard Deviation	**Baseline**	**Ref=1**	**Ref=2**	**Ref=4**	**Ref=8**	**Ref=10**	**Ref=12**	**Delta**
Freedom	2.19	2.21	2.20	2.19	2.19	2.19	2.16	1.97%
Football	3.51	3.58	3.56	3.53	3.51	3.52	3.50	2.30%
Average	**2.85**	**2.89**	**2.88**	**2.86**	**2.85**	**2.86**	**2.83**	**2.17%**

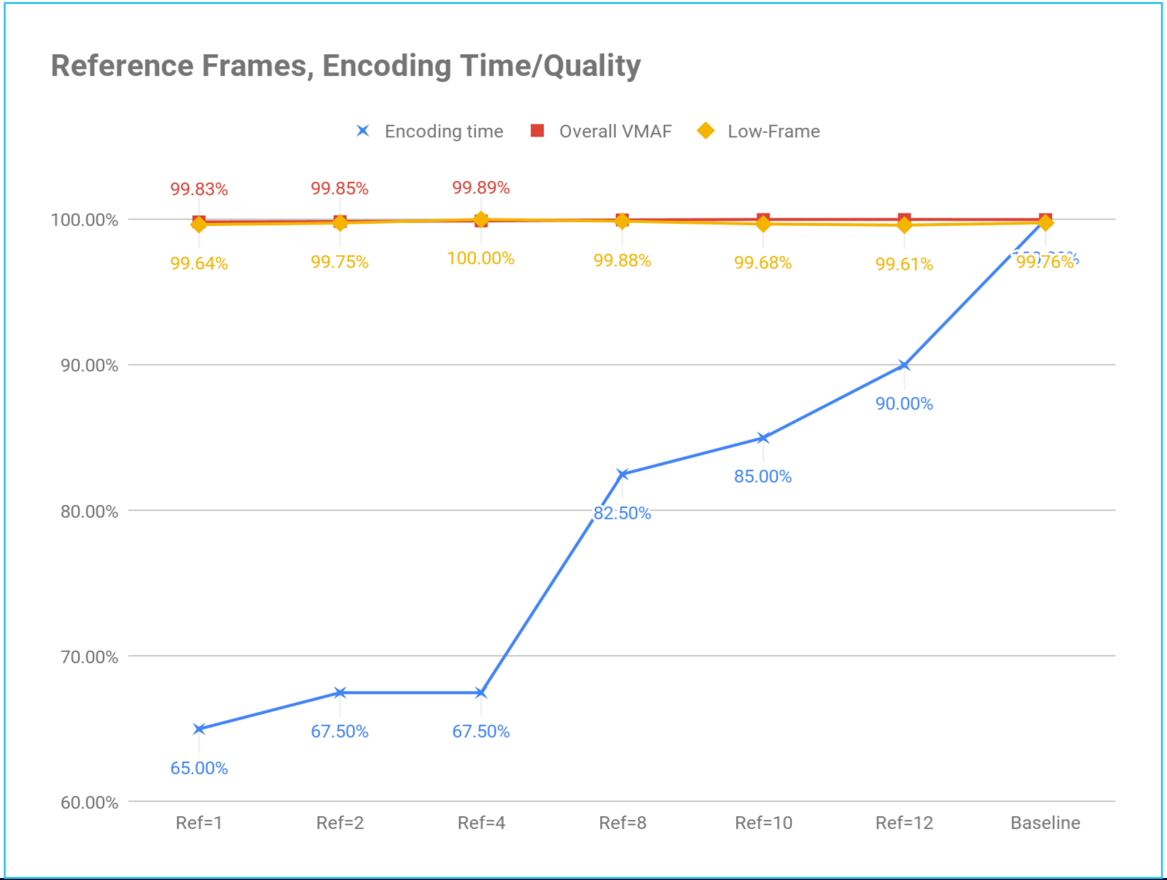
35% encoding time

Overall VMAF

Low Frame VMA

VMAF Std. Dev.

Reference Frames



- May work differently with different source clips
  - I tested football and a concert video
- Test - may be able to shave 35% from encoding costs with minimal quality impact

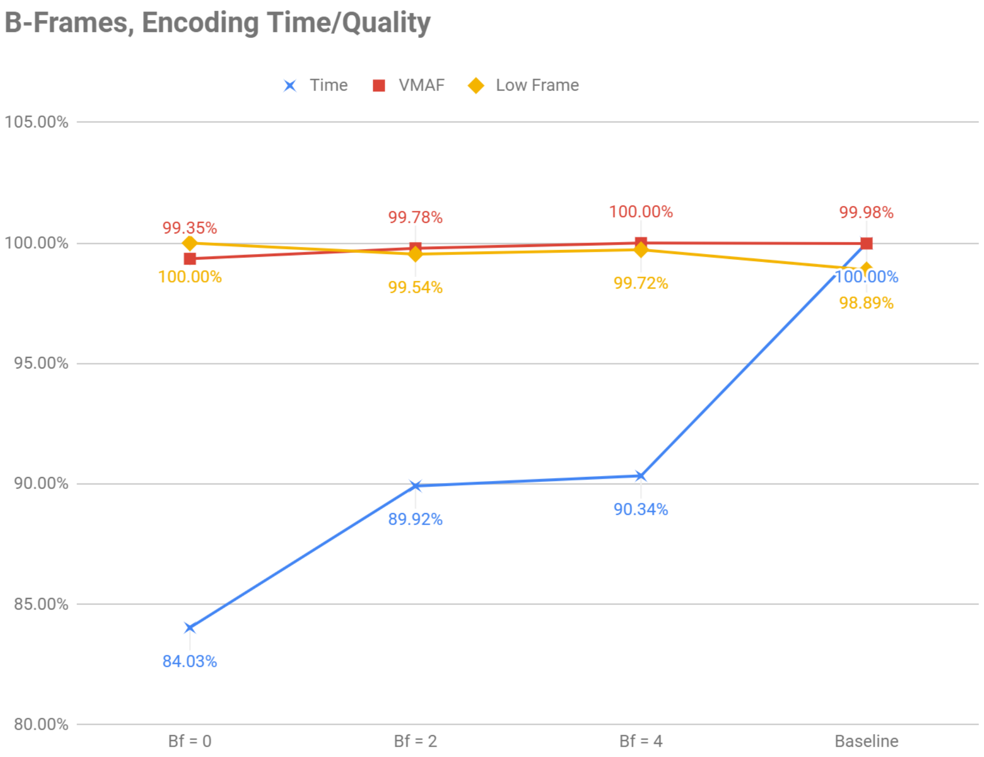
B-Frames

- B-frames - theoretically the most efficient
  - Tend to underperform
- 16% encoding time delta
- Minimal VMAF delta
- Worst:
  - Low-frame
  - Worst standard deviation

	Bf = 0	Bf = 2	Bf = 4	Baseline	Delta
Freedom	0:03:22	0:03:37	0:03:36	0:03:53	13.30%
Football	0:03:18	0:03:31	0:03:34	0:04:03	18.52%
Average	**0:03:20**	**0:03:34**	**0:03:35**	**0:03:58**	**15.91%**
Bitrate	**Bf = 0**	**Bf = 2**	**Bf = 4**	**Baseline**	**Delta**
Freedom	3,800	3,804	3,807	3,812	0.18%
Football	4,202	4,200	4,204	4,200	0.10%
Average	**4,001**	**4,002**	**4,006**	**4,006**	**0.11%**
VMAF	**Bf = 0**	**Bf = 2**	**Bf = 4**	**Baseline**	**Delta**
Freedom	94.01	93.97	94.29	94.26	0.34%
Football	90.93	91.78	91.87	91.86	1.03%
Average	**92.47**	**92.88**	**93.08**	**93.06**	**0.66%**
Low Frame	**Bf = 0**	**Bf = 2**	**Bf = 4**	**Baseline**	**Delta**
Freedom	89.72	88.35	88.57	88.03	1.92%
Football	83.88	84.45	84.55	83.64	1.09%
Average	**86.80**	**86.40**	**86.56**	**85.83**	**1.13%**
Standard Deviation	**Bf = 0**	**Bf = 2**	**Bf = 4**	**Baseline**	**Delta**
Freedom	2.02	2.26	2.32	2.31	14.58%
Football	3.38	3.11	3.08	3.07	10.29%
Average	**2.70**	**2.69**	**2.70**	**2.69**	**0.54%**

B-Frames

- If you use a high-quality preset, experiment with lower b-frame values



Constrained VBR Levels

- Minimal impact on encoding time
- VMAF about the same
- Low frame greater delta
- More significant in standard deviation
- Not shown - obviously, higher max bitrates
  - Could disrupt stream in constrained connections

Constrained VBR % - x264					
Encoding time	**200%**	**100%**	**150%**	**300%**	**Delta**
Freedom	0:00:50	0:00:48	0:00:48	0:00:50	4.00%
Football	0:00:30	0:00:32	0:00:30	0:00:32	6.25%
Average	**0:00:40**	**0:00:40**	**0:00:39**	**0:00:41**	**4.88%**
Bitrate	**200%**	**100%**	**150%**	**300%**	**Delta**
Freedom	3,712	3,714	3,713	3,711	0.08%
Football	4,132	4,124	4,133	4,134	0.24%
Average	**3,922**	**3,919**	**3,923**	**3,923**	**0.10%**
VMAF	**200%**	**100%**	**150%**	**300%**	**Delta**
Freedom	93.89	93.88	93.91	93.91	0.03%
Football	94.02	93.73	93.99	94.00	0.31%
Average	93.96	93.80	93.95	93.95	**0.16%**
Low Frame	**200%**	**100%**	**150%**	**300%**	**Delta**
Freedom	89.34	89.10	89.41	89.42	0.36%
Football	83.86	80.78	83.11	83.77	3.68%
Average	86.60	84.94	86.26	86.59	1.92%
Standard Deviation	**200%**	**100%**	**150%**	**300%**	**Delta**
Freedom	2.19	2.18	2.18	2.18	0.35%
Football	3.51	4.32	3.54	3.50	18.88%
Average	**2.85**	**3.25**	**2.86**	**2.84**	**12.50%**

VBR Constraints of the Rich and Famous

https://bit.ly/con_vbr

YouTube	Rez	Codec	Average	Max	Multiple		
TopGun	8K	AV1	18,159	44,878	2.5		
TopGun	4K	AV1	8,304	30,087	3.6		
TopGun	2K	AV1	4,958	16,800	3.4		
TopGun	1080p	AV1	1,592	7,750	4.9		
Soccer	8K	AV1	20,247	72,808	3.6		
Soccer	4K	AV1	9,823	37,143	3.8		
Soccer	2K	AV1	4,790	17,010	3.6		
Soccer	1080p	AV1	2,224	6,035	2.7	3.5	Average AV1
TOS	720p	AVC	1,977	3,911	2.0		
TOS	1080p	AVC	3,865	8,860	2.3		
Olivia Rodego - Brutal	1080p	AVC	5,239	12,969	2.5		
Selena Gomez	1080p	AVC	1,846	5,364	2.9		
TopGun	1080p	AVC	3,067	10,624	3.5		
Soccer	1080p	AVC	5,380	11,512	2.1	2.5	Average AVC
TOS	720p	VP9	1,005	2,215	2.2		
TOS	1080p	VP9	1,610	3,461	2.1		
Costa Rica	4K	VP9	26,702	52,194	2.0		
TopGun	4K	VP9	9,672	25,008	2.6		
TopGun	2K	VP9	5,292	12,965	2.4		
TopGun	1080p	VP9	1,258	4,675	3.7		
Soccer	4K	VP9	19,894	41,739	2.1		
Soccer	2K	VP9	10,617	20,659	1.9		
Soccer	1080p	VP9	4,040	7,183	1.8	2.3	Average VP9
Average					**2.8**		
Max					**4.9**		

Facebook	Rez	Codec	Average	Max	Multiple		
TOS	720p	AVC	2,196	8,637	3.9		
COVID	1080x1080	AVC	993	9,413	9.5		
Cop Cam	640x640	AVC	579	2,573	4.4		
Kimmel	720p	AVC	1,590	4,406	2.8		
Ukraine Interview	720p	AVC	1,016	3,738	3.7	4.9	Average AVC
Colber	1080p	VP9	1,376	3,609	2.6		
Biden in Poland	1080p	VP9	842	3,104	3.7		
Fatboy	1080x1080	VP9	1,577	5,772	3.7		
Cat falling	1080x1920	VP9	2,384	3,822	1.6		
Interview	720p	VP9	1,650	3,967	2.4	2.8	Average VP9
Average					**3.8**		
Max					**9.5**		

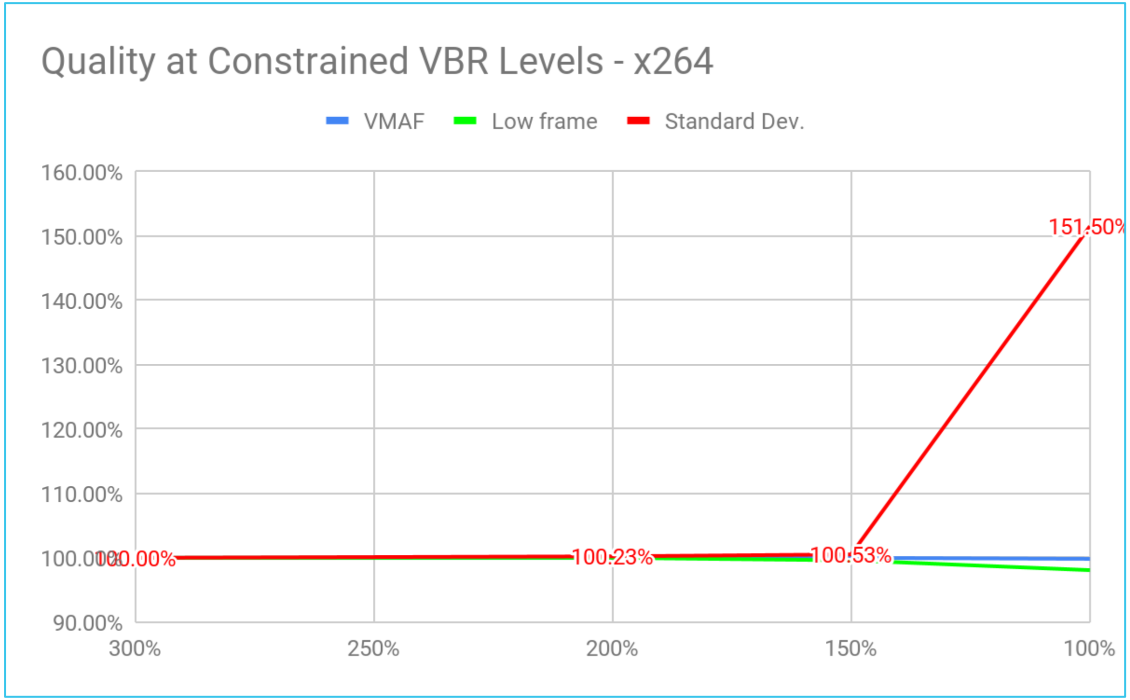
VBR Constraints of the Rich and Famous

Apple Trailer	Rez	Codec	Average	Max	Multiple
Topgun 1	816p	AVC	8,827	15,471	1.8
All the Old Knives	816p	AVC	9,326	38,862	4.2
All My Friends Hate Me	816p	AVC	9,485	13,024	1.4
Agent Game	1056p	AVC	9,242	44,803	4.8
Topgun 2	816p	AVC	9,731	16,471	1.7
Topgun 3	1080p	AVC	9,417	25,372	2.7
Topgun 4	816p	AVC	9,594	24,957	2.6
Average					**2.7**
Max					**4.8**

Bitrate Control Strategy	2x	Over 2x	Over 4x
YouTube			x
Vimeo		x	
Facebook			x
Three-letter network	x		
Apple trailers			x
CNN	x		
NY Times			x
Wall Street Journal		x	
Washington Post		x	

Constrained VBR Levels

- On these test clips
  - Minimal delta in overall or low-frame
  - BIg delta @ 100% for standard deviation
- If 100% time to rethink
- May want to experiment with 300+

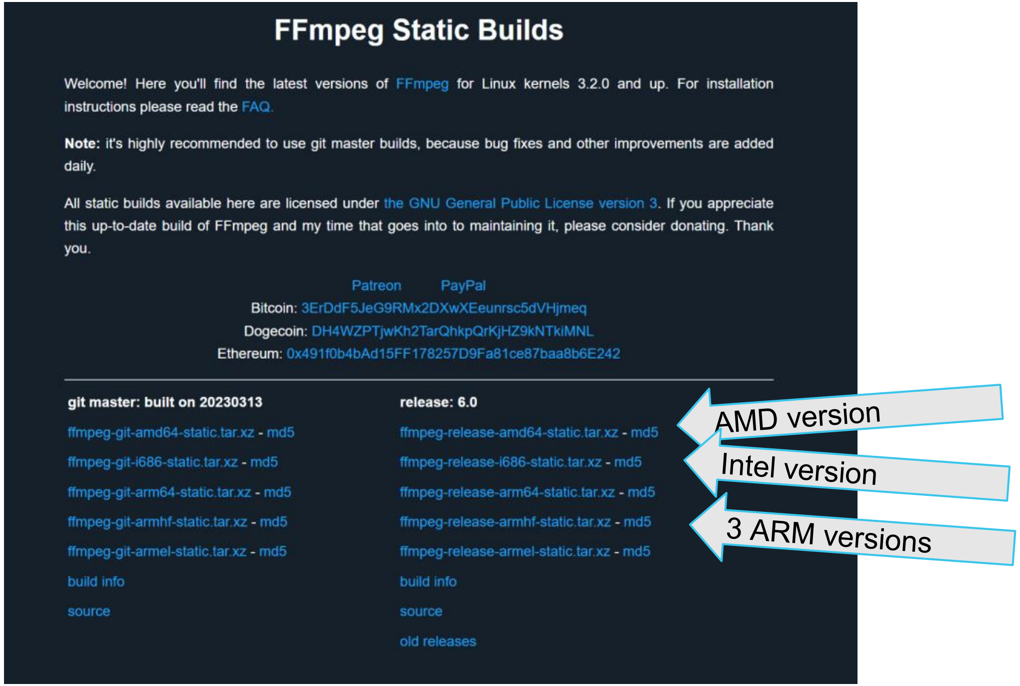


Best Instance for x264

- Three types of instances on AWS
  - Intel (c6i.xlarge) - Compute/Intel
  - AMD (c6a.xlarge) - Compute/AMD
  - AWS - Graviton - (c7g.xlarge) - Compute Graviton
- Which encodes most efficiently?
- Test methodology

CPU Specific Builds

- Different CPUs operate most efficiently (in theory) using specialized builds
- If you're going to experiment with different CPUs, should either compile independently or try third-party



FFmpeg Static Builds

Welcome! Here you'll find the latest versions of FFmpeg for Linux kernels 3.2.0 and up. For installation instructions please read the FAQ.

Note: it's highly recommended to use git master builds, because bug fixes and other improvements are added daily.

All static builds available here are licensed under the GNU General Public License version 3. If you appreciate this up-to-date build of FFmpeg and my time that goes into to maintaining it, please consider donating. Thank you.

Patreon PayPal
Bitcoin: 3ErDdF5JeG9RMx2DXwXEeunrsc5dVHjmeq
Dogecoin: DH4WZPTjwKh2TarQhkpQrKjHZ9kNTkiMNL
Ethereum: 0x491f0b4bAd15FF178257D9Fa81ce87baa8b6E242

git master: built on 20230313

- ffmpeg-git-amd64-static.tar.xz - md5
- ffmpeg-git-i686-static.tar.xz - md5
- ffmpeg-git-arm64-static.tar.xz - md5
- ffmpeg-git-armhf-static.tar.xz - md5
- ffmpeg-git-armel-static.tar.xz - md5
- build info
- source

release: 6.0

- ffmpeg-release-amd64-static.tar.xz - md5
- ffmpeg-release-i686-static.tar.xz - md5
- ffmpeg-release-arm64-static.tar.xz - md5
- ffmpeg-release-armhf-static.tar.xz - md5
- ffmpeg-release-armel-static.tar.xz - md5
- build info
- source
- old releases

Methodology

- FFmpeg version: Downloaded CPU specific versions from johnvansickle.com/ffmpeg/
  - Compared with Ubuntu native version (4.4)
  - Used faster version
  - Used version from Multicoreware for Graviton (x265 performance on Graviton was very poor until this version)

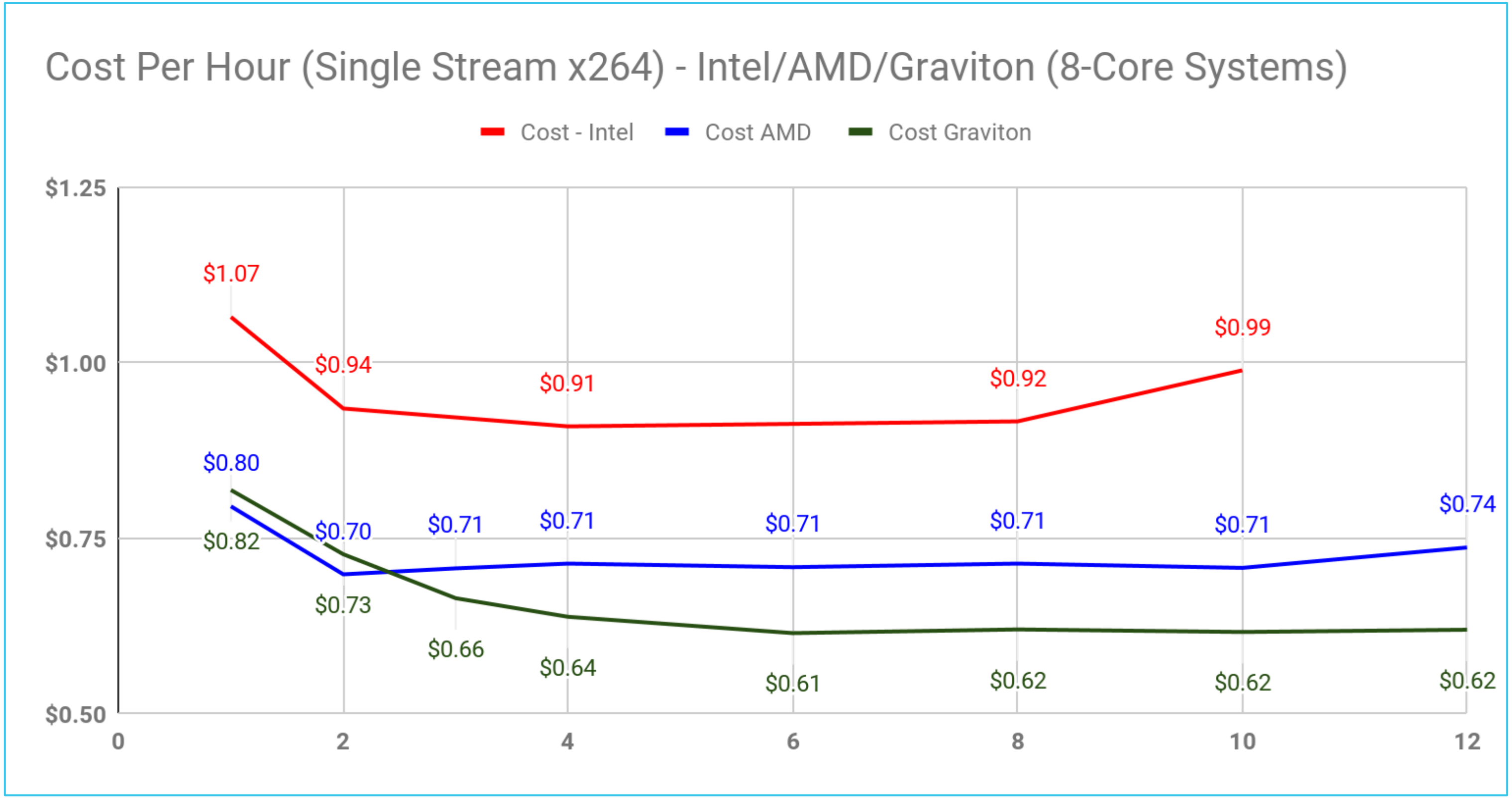
- Encoded test file using multiple instances to find most efficient encoding cost
- Produce single 1080p output stream (not encoding ladder)
- Track seconds, compute cost per hour
- Rinse and repeat with different CPUs

Instances	Encode time	Frames	Frames/ second	Frames/ hour	seconds of video	Minutes of video	Cost/Hour
1	110	900	8.18	29,455	982	16.36	$1.24667
2	171	1800	10.53	37,895	1,263	21.05	$0.96900
4	325	3600	11.08	39,877	1,329	22.15	$0.92083
8	649	7200	11.09	39,938	1,331	22.19	$0.91942
10	873	9000	10.31	37,113	1,237	20.62	$0.98940

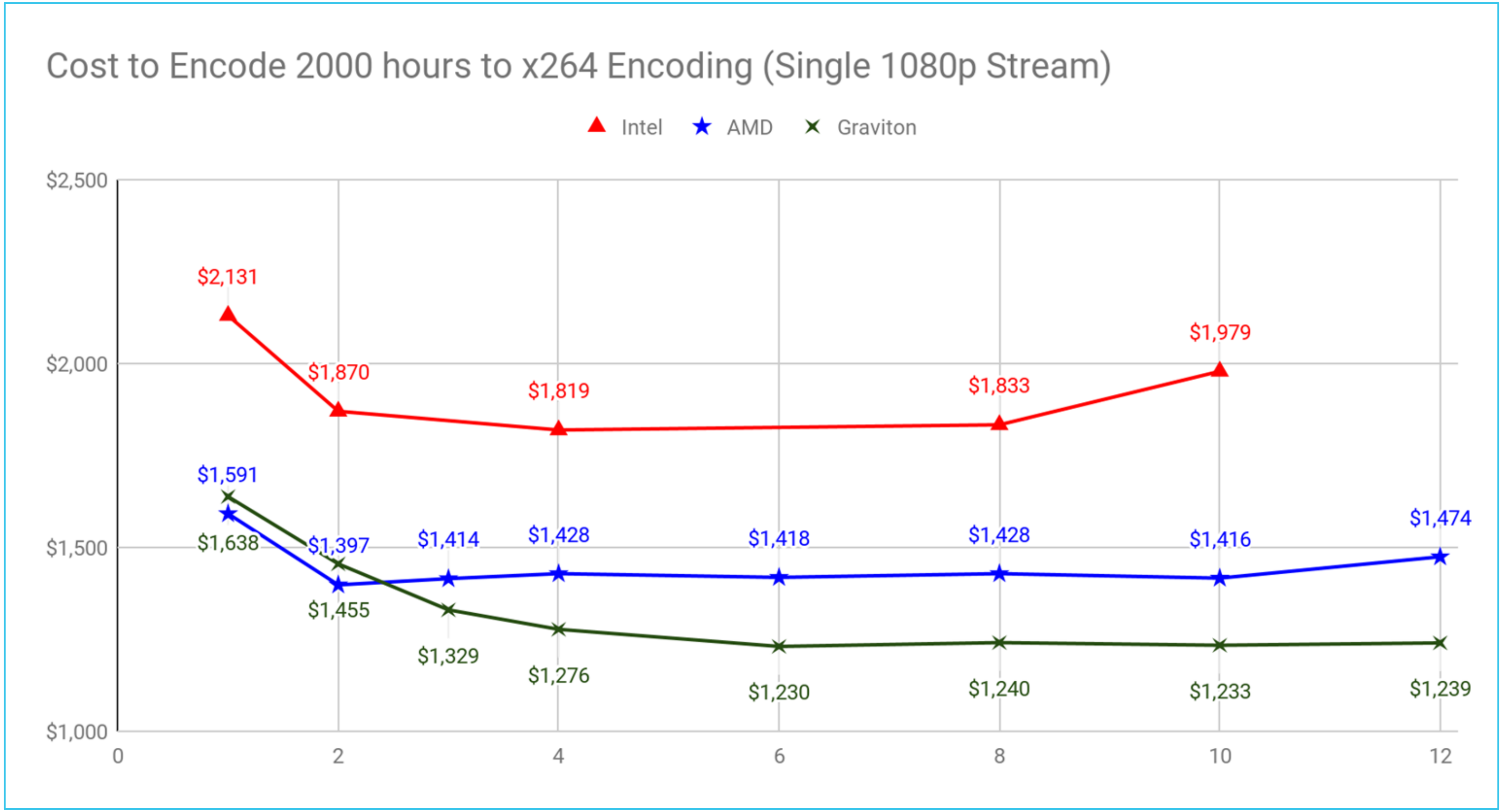
8-Core AWS Instance - On Demand Pricing

- AMD - c6a.2xlarge - $0.306
- Graviton - c7g.2xlarge - $0.289
- Intel - c6i.2xlarge - $0.34

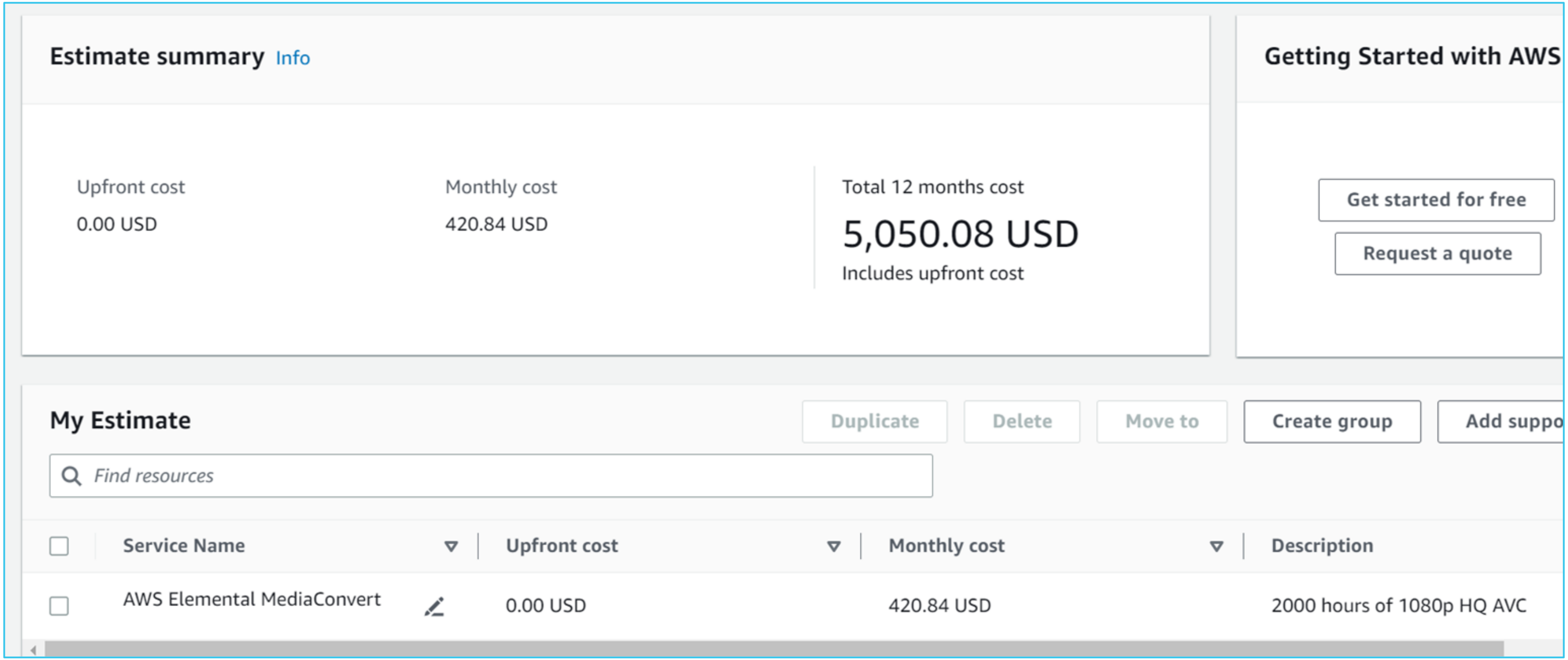
The Winner Is - For x264 – Graviton (per-hour)



The Winner Is - For x264 – Graviton (2000 hours)



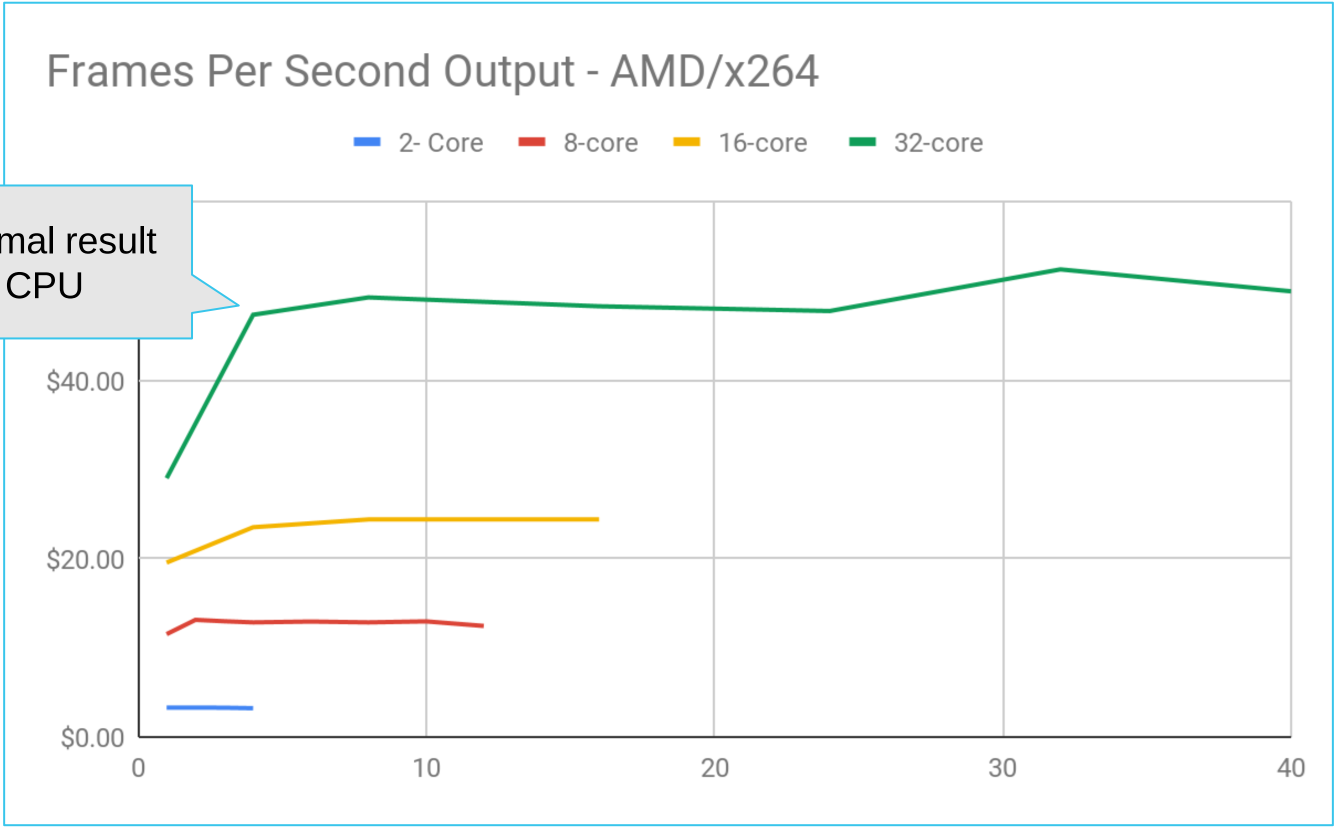
Reality Check: MediaConvert Pricing - 2000 hours AVC HQ



Encoding String

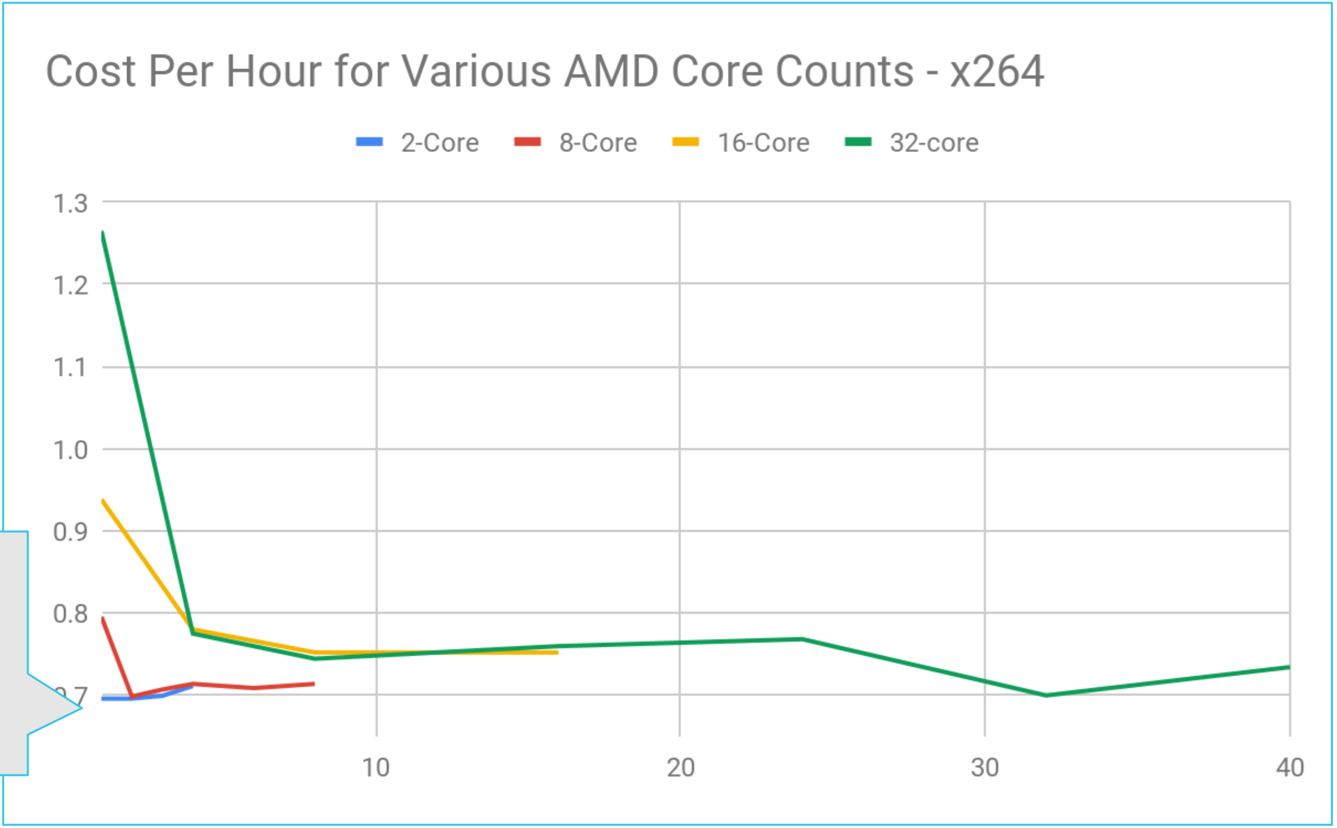
```
ffmpeg -y -i Orchestra.mp4 -c:v libx264 -profile:v high  -preset veryslow -g 60
-keyint_min 60 -sc_threshold 0  -b:v 4200k -pass 1  -f mp4 /dev/null

ffmpeg -y -i Orchestra.mp4 -c:v libx264 -profile:v high  -preset veryslow -g 60
-keyint_min 60 -sc_threshold 0  -b:v 4200k -maxrate 8400k -bufsize 8400k -pass
2  orchestra_x264_output.mp4
```

Most Efficient CPU Core Count?



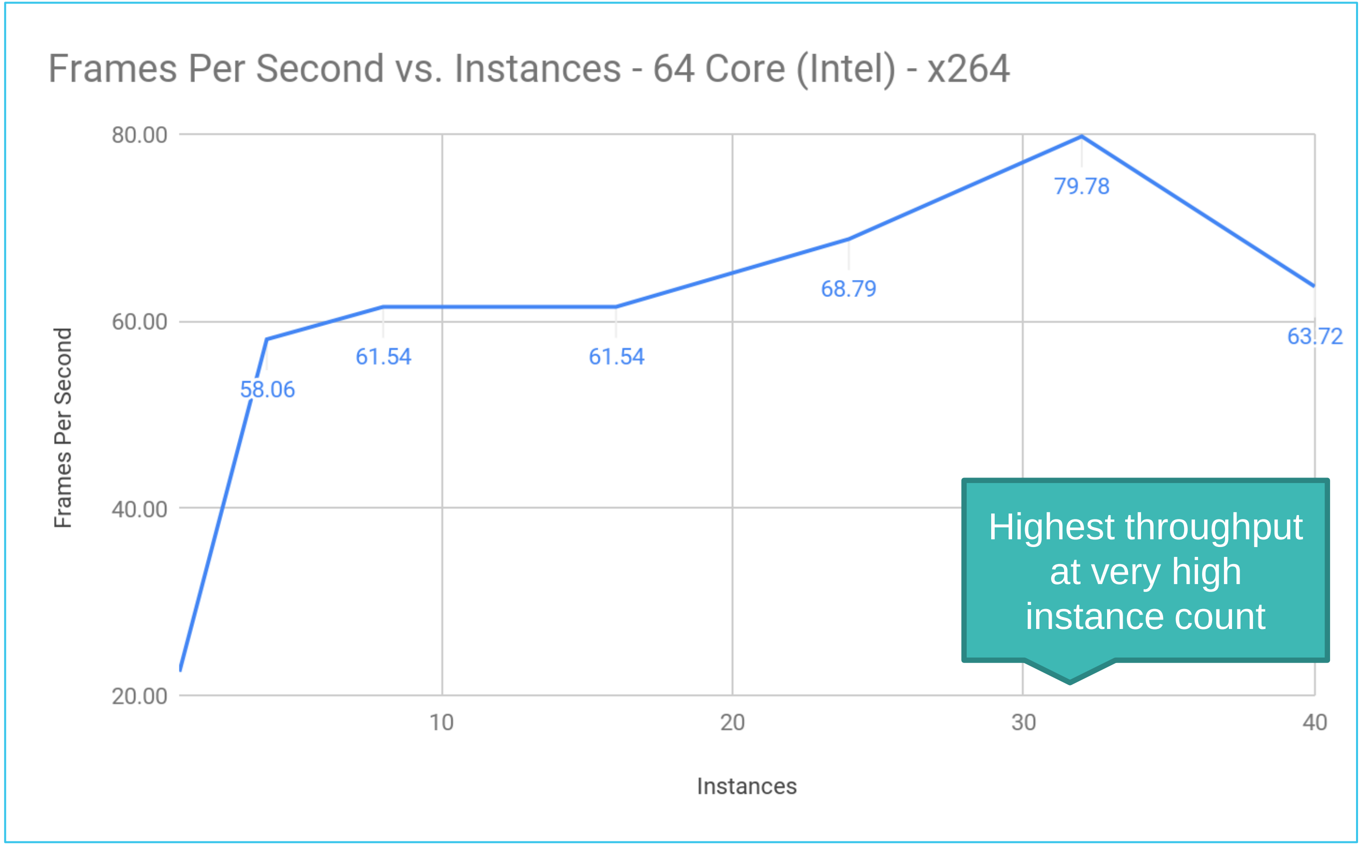
Seldom get optimal result at 1 encode per CPU

Most Efficient CPU Core Count?

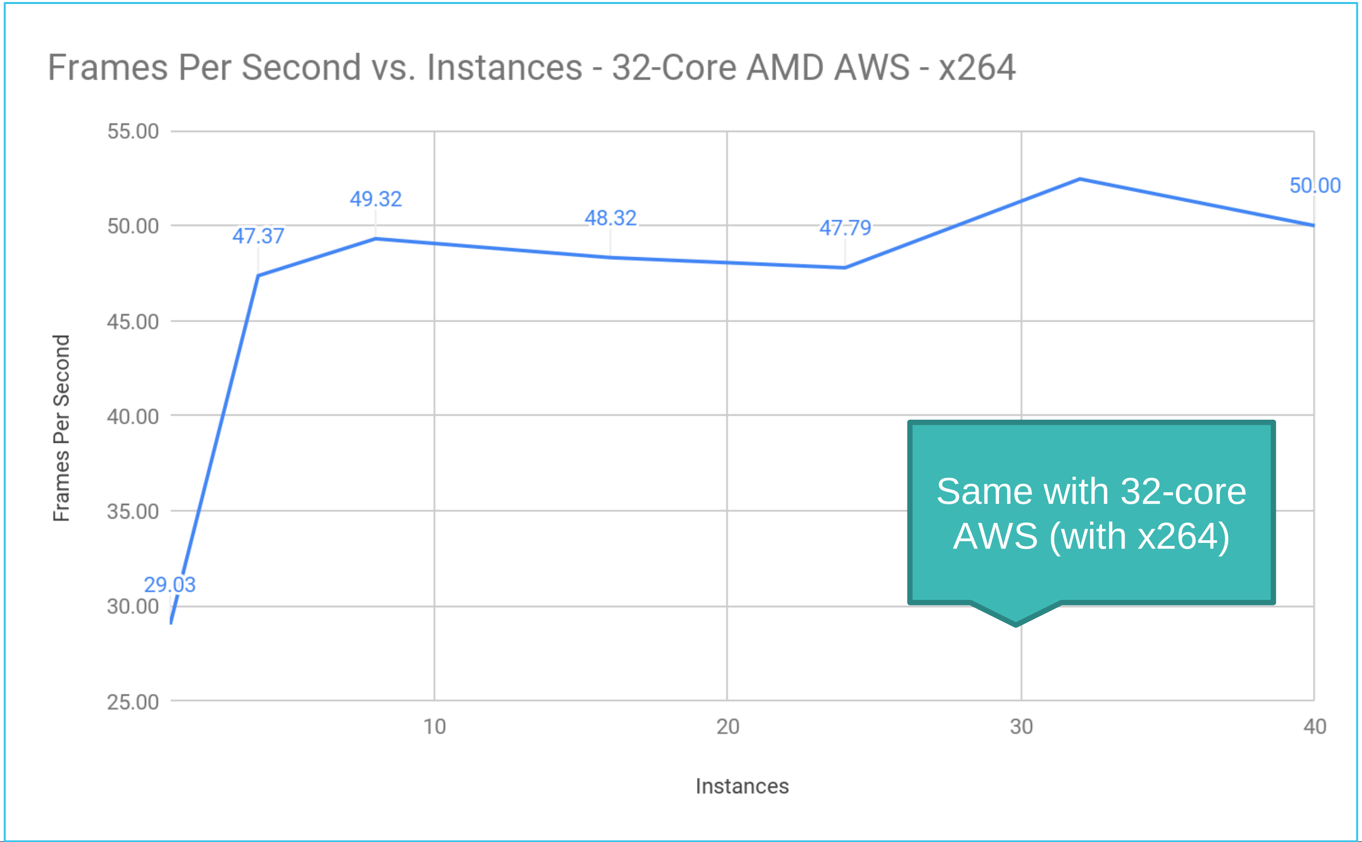


May be slight advantage using 8-cores for 1080p30 x264

Frames Per Second - 64-core Intel Workstation



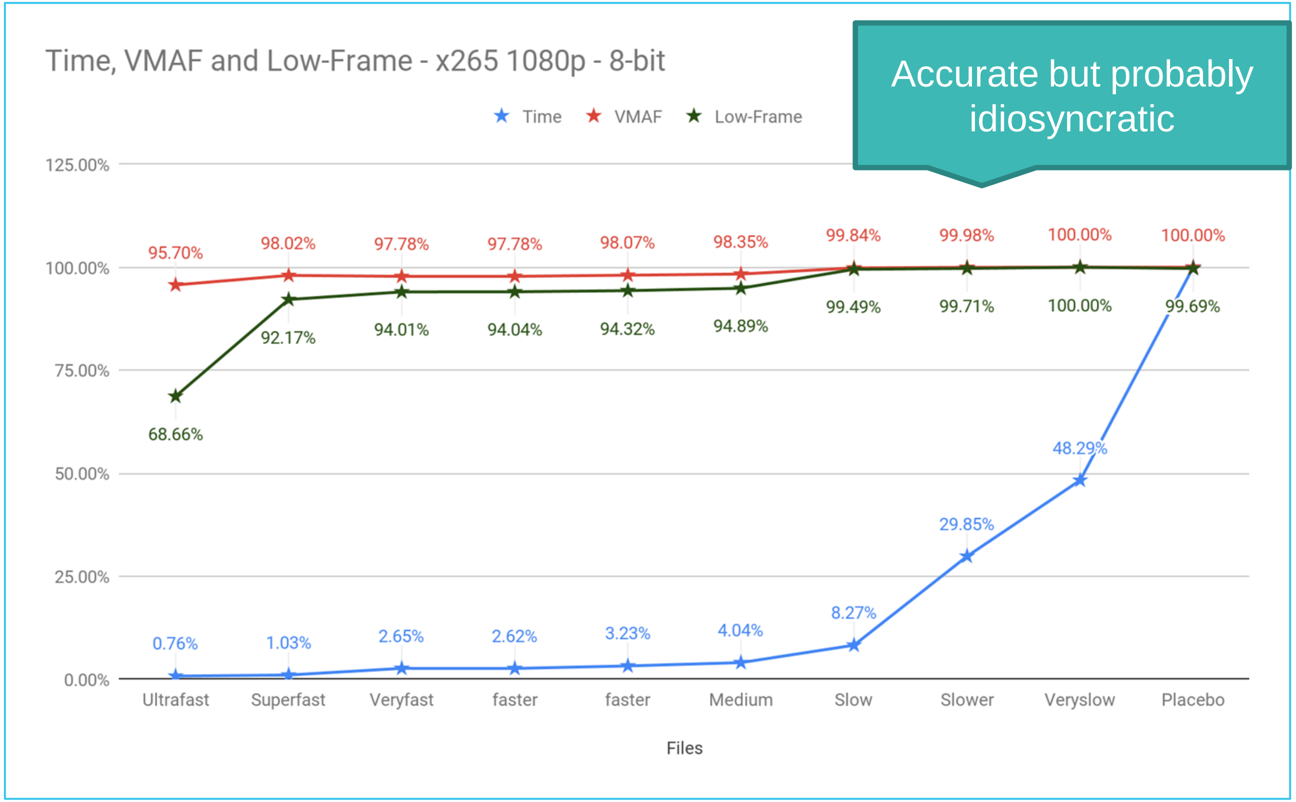
Frames Per Second - 32-core AWS AMD Instance



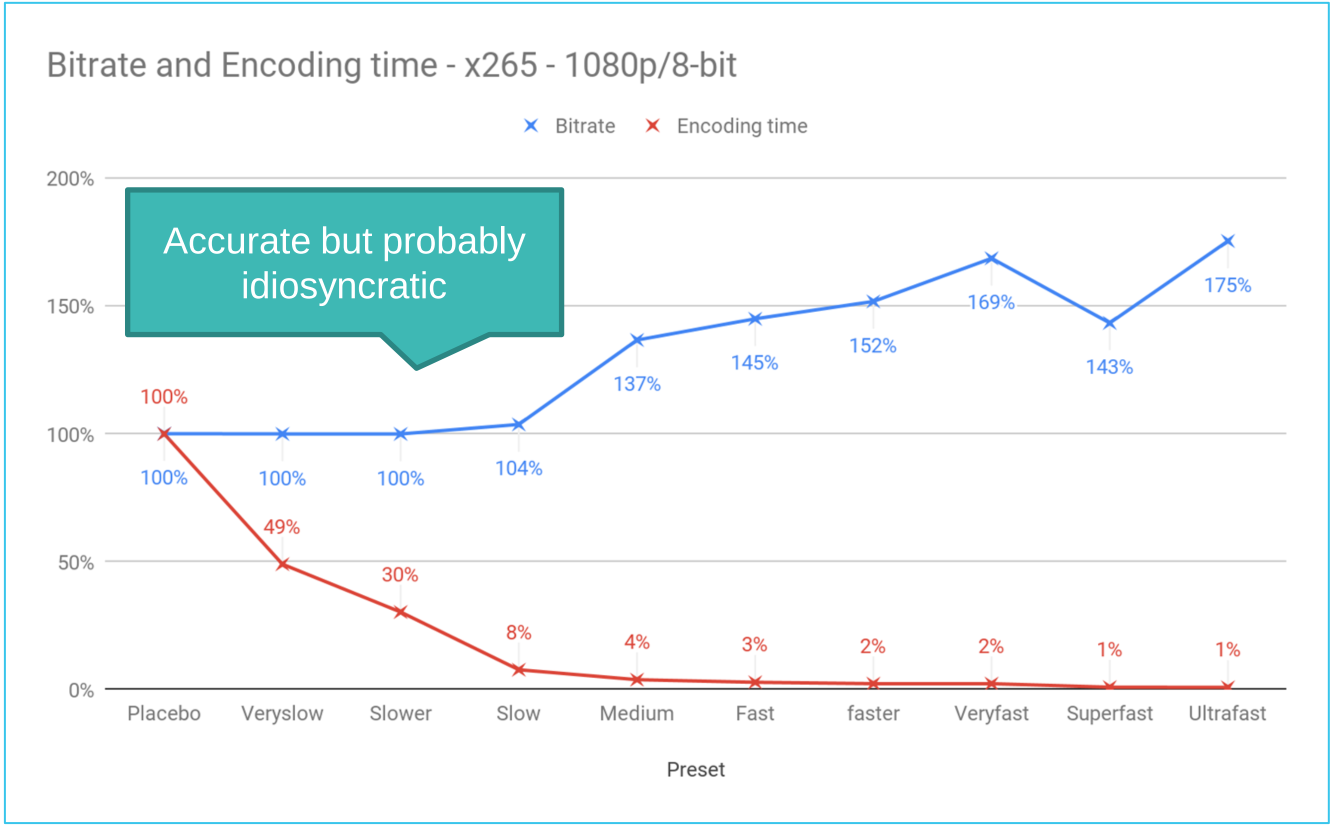
x265 - 8-Bit 1080p (Rinse and Repeat)

- Preset
- Reference frames
- Bitrate control
- 10-bit vs 8-bit output
- Best AWS CPU
- Best core count

HEVC - 8-bit 1080p Preset



HEVC - 8-bit 1080p Preset



Preset	**Bitrate**	**Encoding time**
Ultrafast	175%	1%
Superfast	143%	1%
Veryfast	169%	2%
faster	152%	2%
Fast	145%	3%
Medium	137%	4%
Slow	104%	8%
Slower	100%	30%
Veryslow	100%	49%
Placebo	100%	100%

x265 - 1080p - Viewer Count Breakeven - $0.08/GB

10x encoding cost increase

At higher bandwidth costs, saving bandwidth matters more than encoding costs.

		Bitrate		2500						
		MBytes per hour		1125		Cost per GB		0.08		
		Encode/hr		5.5						
Preset	**Encode**	**Bandwidth**			**50**	**100**	**250**	**500**	**1000**	**5000**
Ultrafast	$0.53	2.19	$0.18		$9	$18	$44	$88	$176	$876
Superfast	$0.59	1.92	$0.15		$8	$16	$39	$77	$154	$767
Veryfast	$0.73	1.69	$0.13		$7	$14	$34	$68	$136	$675
faster	$0.99	1.41	$0.11		$7	$12	$29	$57	$114	$564
fast	$1.25	1.40	$0.11		$7	$12	$29	$57	$114	$563
Medium	$1.44	1.25	$0.10		$6	$11	$27	$52	$102	$503
Slow	$2.08	1.20	$0.10		$7	$12	$26	$50	$98	$483
Slower	$2.95	1.17	$0.09		$8	$12	$26	$50	$97	$471
Veryslow	$5.50	1.13	$0.09		$10	$15	$28	$51	$96	$456
Placebo	$21.89	1.13	$0.09		$26	$31	$44	$67	$112	$473

x265 - 1080p - Viewer Count Breakeven - $0.04/GB

		Bitrate		**2500**						
		MBytes per hour		**1125**		**Cost per GB**		**0.04**		
		Encode/hr		**5.5**						
Preset	**Encode**	**Bandwidth**			**50**	**100**	**250**	**500**	**1000**	**5000**
Ultrafast	$0.53	2.19	$0.09		$5	$9	$22	$44	$88	$438
Superfast	$0.59	1.92	$0.08		$4	$8	$20	$39	$77	$384
Veryfast	$0.73	1.69	$0.07		$4	$7	$18	$34	$68	$338
faster	$0.99	1.41	$0.06		$4	$7	$15	$29	$57	$282
fast	$1.25	1.40	$0.06		$4	$7	$15	$29	$57	$282
Medium	$1.44	1.25	$0.05		$4	$6	$14	$27	$52	$252
Slow	$2.08	1.20	$0.05		$4	$7	$14	$26	$50	$243
Slower	$2.95	1.17	$0.05		$5	$8	$15	$26	$50	$237
Veryslow	$5.50	1.13	$0.05		$8	$10	$17	$28	$51	$231
Placebo	$21.89	1.13	$0.05		$24	$26	$33	$44	$67	$247

x265 - Viewer Count Breakeven - $0.02/GB

As bandwidth costs drop, encoding cost matters longer

		Bitrate		2500						
		MBytes per hour		1125		Cost per GB		0.02		
		Encode/hr		5.5						
Preset	Encode	Bandwidth			50	100	250	500	1000	5000
Ultrafast	$0.53	2.19	$0.04		$3	$5	$11	$22	$44	$219
Superfast	$0.59	1.92	$0.04		$3	$4	$10	$20	$39	$192
Veryfast	$0.73	1.69	$0.03		$2	$4	$9	$18	$34	$169
faster	$0.99	1.41	$0.03		$2	$4	$8	$15	$29	$142
fast	$1.25	1.40	$0.03		$3	$4	$8	$15	$29	$142
Medium	$1.44	1.25	$0.03		$3	$4	$8	$14	$27	$127
Slow	$2.08	1.20	$0.02		$3	$4	$8	$14	$26	$122
Slower	$2.95	1.17	$0.02		$4	$5	$9	$15	$26	$120
Veryslow	$5.50	1.13	$0.02		$7	$8	$11	$17	$28	$118
Placebo	$21.89	1.13	$0.02		$23	$24	$28	$33	$44	$135

x264 - Viewer Count Breakeven - $0.02/GB

As bandwidth costs drop, encoding cost matters longer (but still not that long)

		Bitrate		**2500**						
		MBytes per hour		**1125**		**Cost per GB**		**0.02**		
		Encode/hr		**5.5**						
Preset	**Encode**	**Bandwidth**			**50**	**100**	**250**	**500**	**1000**	**5000**
Ultrafast	$0.53	2.19	$0.04		$3	$5	$11	$22	$44	$219
Superfast	$0.59	1.92	$0.04		$3	$4	$10	$20	$39	$192
Veryfast	$0.73	1.69	$0.03		$2	$4	$9	$18	$34	$169
faster	$0.99	1.41	$0.03		$2	$4	$8	$15	$29	$142
fast	$1.25	1.40	$0.03		$3	$4	$8	$15	$29	$142
Medium	$1.44	1.25	$0.03		$3	$4	$8	$14	$27	$127
Slow	$2.08	1.20	$0.02		$3	$4	$8	$14	$26	$122
Slower	$2.95	1.17	$0.02		$4	$5	$9	$15	$26	$120
Veryslow	$5.50	1.13	$0.02		$7	$8	$11	$17	$28	$118
Placebo	$21.89	1.13	$0.02		$23	$24	$28	$33	$44	$135

Default

Reference Frames

Writing library : x265 3.5+96-9c9ab68fc:[Windows][GCC 12.2.0][64 bit] 8bit+10bit+12bit
cpuid=1111039 / frame-threads=5 / numa-pools=32,32 / wpp / no-pmode / no-pme / no-psnr / no-ssim / log-level=2 / input-csp=1 / input-res=1920x1080 / interlace=0 / total-frames=0 / level-idc=0 / high-tier=1 / uhd-bd=0 / ref=5 / no-allow-non-conformance / no-repeat-headers / annexb / no-aud / no-eob / no-eos / no-hrd / info / hash=0 / temporal-layers=0 / open-gop / min-keyint=60 / keyint=60 / gop-lookahead=0 / bframes=8 / b-adapt=2 / b-pyramid / bframe-bias=0 / rc-lookahead=40 / lookahead-slices=0 / scenecut=0 / no-hist-scenecut / radl=0 / no-splice / no-intra-

- Slower preset use 5 reference frames
  - How much encoding time does this take?
  - How much quality do they add?
  - Is it worth it/

Reference Frames - x265

	Baseline	Ref=1	Ref=2	Ref=3	Ref=4	Delta
Freedom	0:04:56	0:03:36	0:04:04	0:04:24	0:04:34	27.03%
Football	0:08:34	0:06:22	0:07:04	0:07:42	0:07:58	25.68%
Average	0:06:45	0:04:59	0:05:34	0:06:03	0:06:16	26.17%
Bitrate	**Baseline**	**Ref=1**	**Ref=2**	**Ref=3**	**Ref=4**	**Delta**
Freedom	2,159	2,156	2155	2,161	2,156	0.28%
Football	2,349	2,354	2354	2,354	2,350	0.21%
Average	**2,254**	**2,255**	**2,255**	**2,258**	**2,253**	**0.20%**
VMAF	**Baseline**	**Ref=1**	**Ref=2**	**Ref=3**	**Ref=4**	**Delta**
Freedom	93.45	93.19	93.33	93.40	93.42	0.27%
Football	93.74	93.53	93.59	93.66	93.70	0.22%
Average	**93.59**	**93.36**	**93.46**	**93.53**	**93.56**	**0.25%**
Low Frame	**Baseline**	**Ref=1**	**Ref=2**	**Ref=3**	**Ref=4**	**Delta**
Freedom	88.47	88.03	88.20	88.23	88.50	0.54%
Football	83.54	83.28	83.39	83.28	83.47	0.31%
Average	**86.00**	**85.65**	**85.79**	**85.75**	**85.99**	**0.41%**
Standard Deviation	**Baseline**	**Ref=1**	**Ref=2**	**Ref=3**	**Ref=4**	**Delta**
Freedom	2.51	2.57	2.55	2.52	2.52	2.54%
Football	4.18	4.21	4.20	4.19	4.18	0.76%
Average	**3.34**	**3.39**	**3.38**	**3.35**	**3.35**	**1.44%**

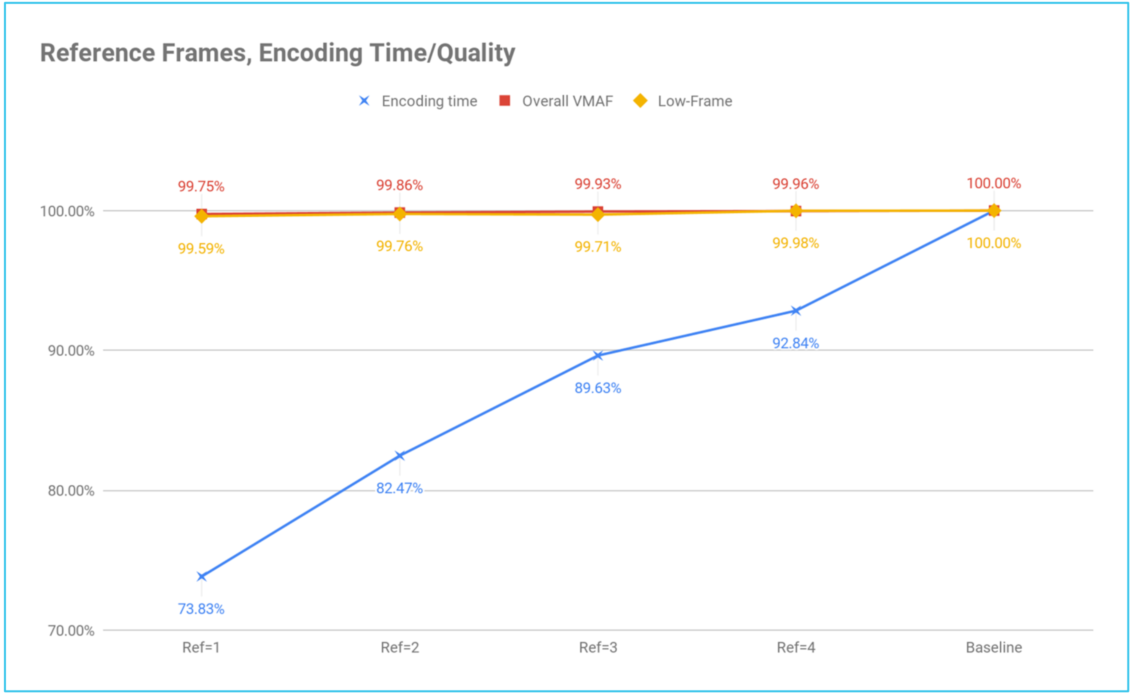
26% encoding time

Overall VMAF

Low Frame VMAF

VMAF Std. Dev.

Reference Frames - x265



- Reduce encoding time by 26%; minimal quality delta.
- May work differently with different source clips
  - I tested football and a concert video
- Test - may be able to shave 25%+ from encoding time with minimal quality impact

Constrained VBR Levels

- Again, 100% (CBR) is worst result all round
- Minimal quality deltas for average and low frame
- Meaningful difference with standard deviation
- Avoid 100%; experiment with higher values in high bandwidth rungs/ environments
    - Top rung
    - IPTV

Constrained VBR % - x265					
Encoding time	**100%**	**150%**	**200%**	**300%**	**Delta**
Freedom	0:04:58	0:04:54	0:04:54	0:04:54	1.34%
Football	0:08:30	0:08:26	0:08:28	0:08:28	0.78%
Average	**0:06:44**	**0:06:40**	**0:06:41**	**0:06:41**	**0.99%**
Bitrate	**100%**	**150%**	**200%**	**300%**	**Delta**
Freedom	2,162	2,159	2,349	2,349	8.09%
Football	2,368	2,350	2,158	2,158	8.87%
Average	**2,265**	**2,255**	**2,254**	**2,254**	**0.51%**
VMAF	**100%**	**150%**	**200%**	**300%**	**Delta**
Freedom	93.47	93.46	93.43	93.46	0.04%
Football	93.50	93.74	93.77	93.75	0.29%
Average	93.49	93.60	93.60	93.60	**0.13%**
Low Frame	**100%**	**150%**	**200%**	**300%**	**Delta**
Freedom	88.56	88.46	88.27	88.53	0.32%
Football	80.86	83.85	83.62	83.47	3.56%
Average	84.71	86.15	85.95	86.00	1.68%
Standard Deviation	**100%**	**150%**	**200%**	**300%**	**Delta**
Freedom	2.47	2.51	2.52	2.51	1.85%
Football	5.44	4.16	4.18	4.19	23.40%
Average	**3.95**	**3.34**	**3.35**	**3.35**	**15.64%**

10-bit vs. 8-bit Output

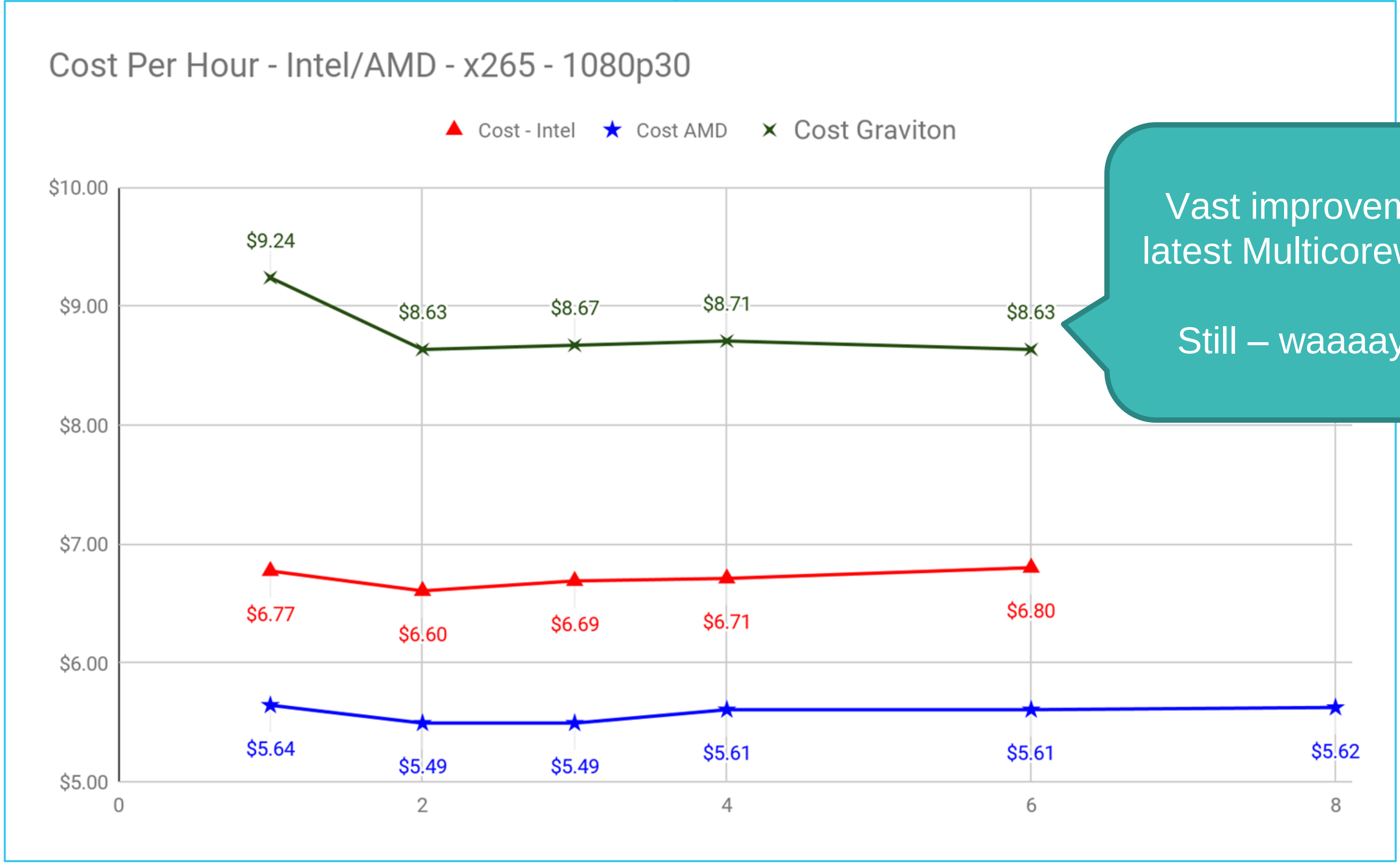
- Question: Does encoding 8-bit source as 10-bit HEVC output improve quality?
- No, but difference is minor

	8-bit	10-bit	Delta
El Ultimo	96.10	96.08	0.02%
Football	94.50	94.33	0.18%
Freedom	92.07	92.06	0.01%
Meridian	95.94	95.94	0.00%
Soccer	96.93	96.76	0.18%
TOS	95.57	95.49	0.09%
Zoo	97.51	97.23	0.28%

Best Instance for x265

- Three types of instances on AWS
  - Intel (c6i.xlarge) - Compute/Intel
  - AMD (c6a.xlarge) - Compute/AMD
  - AWS - Graviton - (c7g.xlarge) - Compute Graviton
- Which encodes most efficiently?
- Test methodology

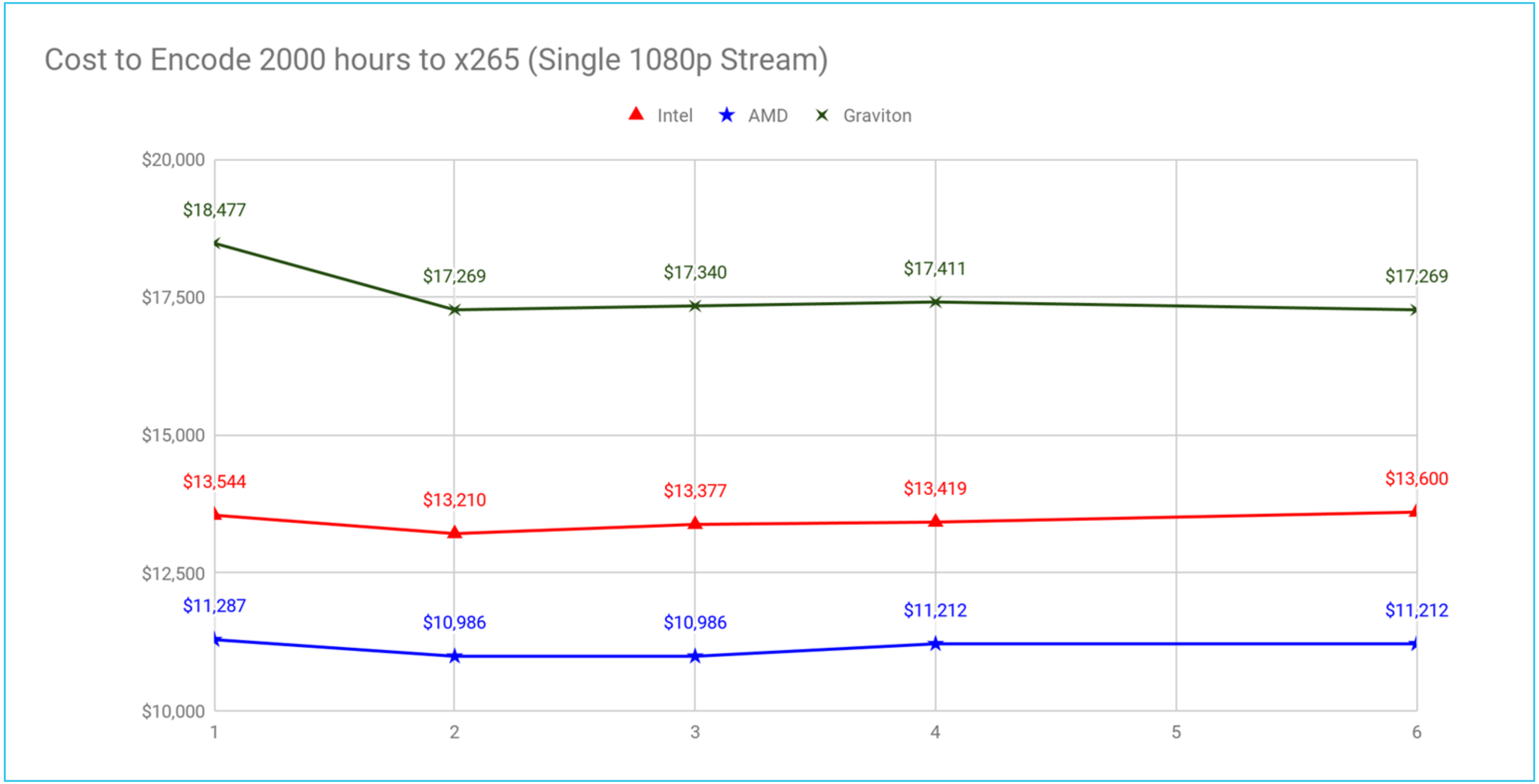
The Winner Is - For x265 1080p - AMD



Vast improvement with latest Multicoreware code

Still – waaaay behind

The Winner Is - For x265 1080p – AMD ($2,000 hours)



Reality Check: MediaConvert Pricing - 2000 hours HEVC HQ

My Estimate Edit

Export ▼ Share

Estimate summary Info

Upfront cost
0.00 USD

Monthly cost
3,366.72 USD

Total 12 months cost
40,400.64 USD
Includes upfront cost

Getting Started with AWS

Get started for free

Request a quote

My Estimate

Duplicate Delete Move to Create group Add support Add service

Find resources

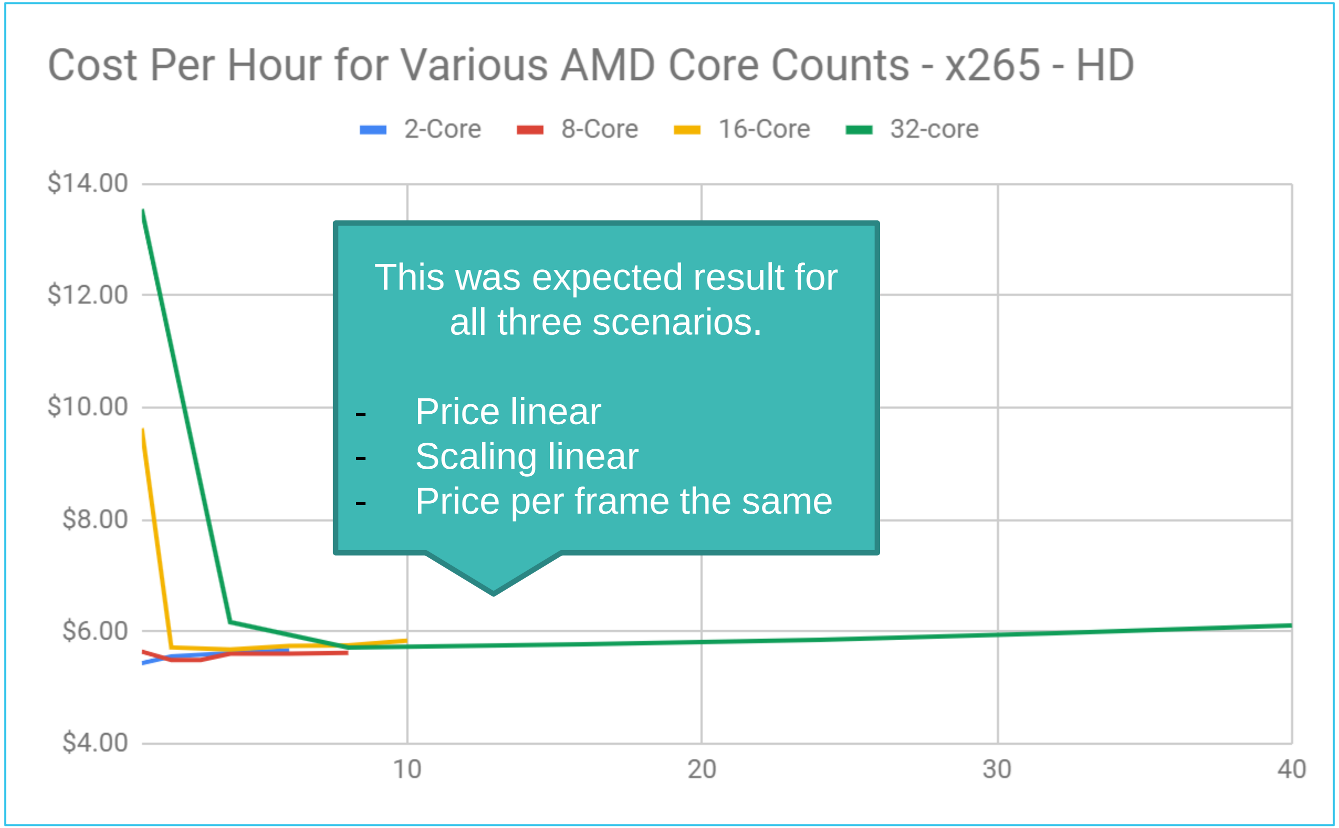
1

Service Name	Upfront cost	Monthly cost	Description	Region	Config Summary
AWS Elemental MediaConvert	0.00 USD	3,366.72 USD	HEVC - 1080p - 2000 hours	US East (Ohio)	Output usage (167 Hours per…

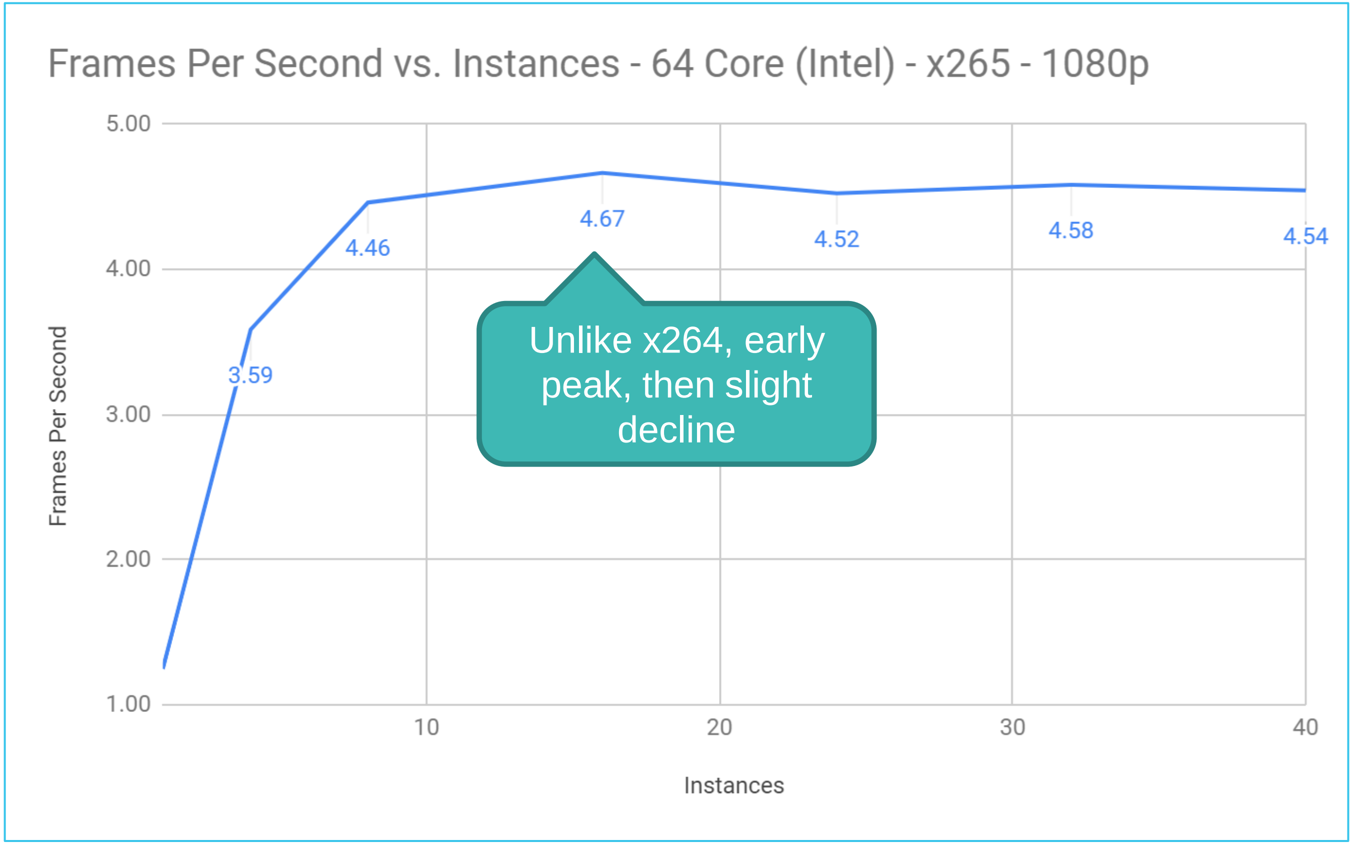
Encoding String

```
ffmpeg  -y -i Football_short.mp4 -c:v libx265 -preset slower -x265-params
keyint=60:min-keyint=60:scenecut=0:bitrate=3500:pass=1  -f mp4 /dev/null

ffmpeg  -y -i Football_short.mp4 -c:v libx265 -preset slower -x265-params
keyint=60:min-keyint=60:scenecut=0:bitrate=3500:vbv-maxrate=7000:vbv-
bufsize=7000:pass=2  Football_x265_HD_output.mp4
```

Most Efficient CPU Core Count?



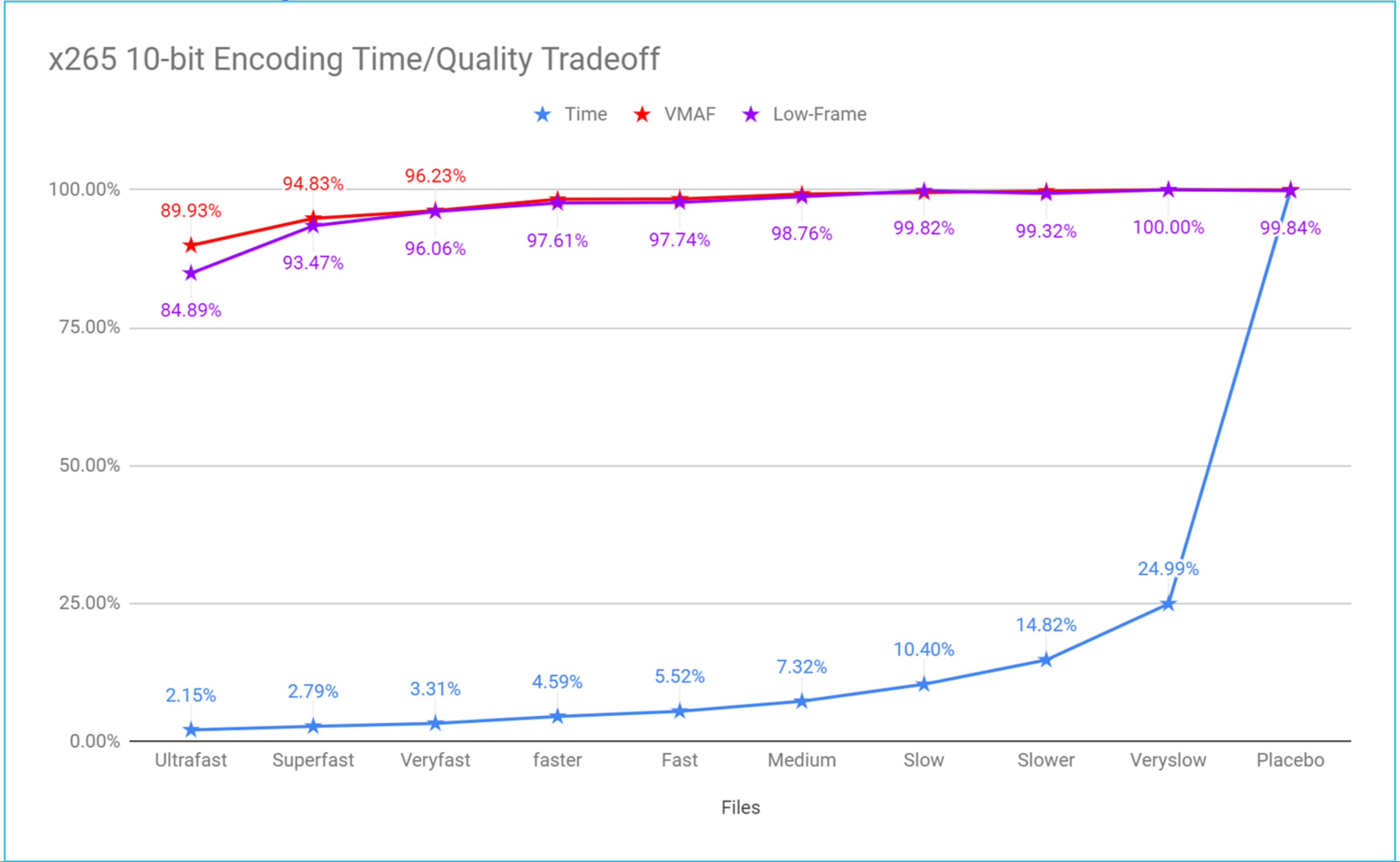
On 64-core Intel Workstation



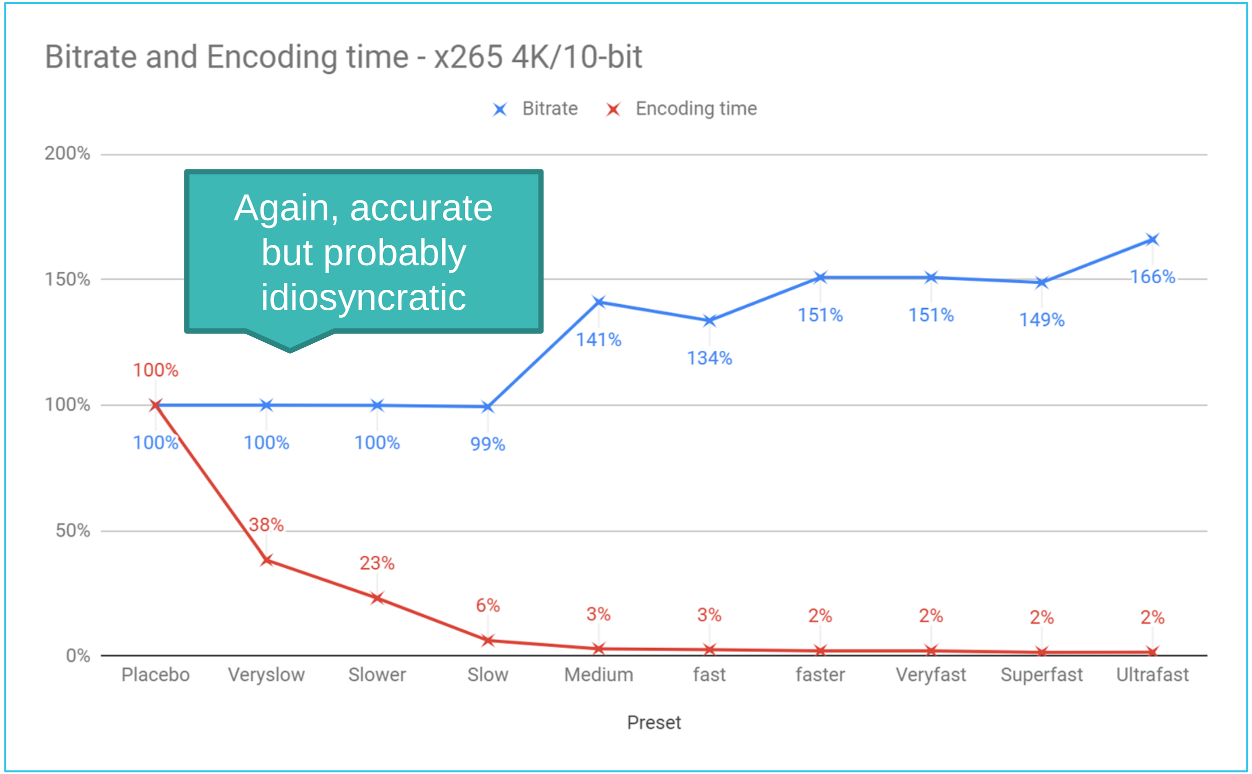
x265 - 10-Bit 4K

- Preset
- Bitrate control
- Scaling
- Best AWS CPU
- Best core count

HEVC -10-bit 1080p Preset



HEVC - 8-bit 1080p Preset



Preset	**Bitrate**	**Encoding time**
Ultrafast	166%	2%
Superfast	149%	2%
Veryfast	151%	2%
faster	151%	2%
fast	134%	3%
Medium	141%	3%
Slow	99%	6%
Slower	100%	23%
Veryslow	100%	38%
Placebo	100%	100%

x265 - 1080p - Viewer Count Breakeven - $0.08/GB

Big changes

At higher bandwidth costs, saving bandwidth matters more than encoding costs.

		Bitrate		15000						
		MBytes per hour		6750		Cost per GB		0.08		
		Encode/hr		15						
Preset	**Encode**	**Bandwidth**			**50**	**100**	**250**	**500**	**1000**	**5000**
Ultrafast	$1.45	13.13	$1.05		$54	$107	$264	$527	$1,052	$5,255
Superfast	$1.61	11.49	$0.92		$48	$94	$231	$461	$921	$4,598
Veryfast	$2.00	10.12	$0.81		$42	$83	$204	$407	$812	$4,050
faster	$2.70	8.44	$0.68		$36	$70	$172	$340	$678	$3,380
fast	$3.42	8.43	$0.67		$37	$71	$172	$340	$678	$3,374
Medium	$3.94	7.52	$0.60		$34	$64	$154	$305	$606	$3,013
Slow	$5.68	7.22	$0.58		$35	$63	$150	$294	$583	$2,893
Slower	$8.04	7.03	$0.56		$36	$64	$149	$289	$570	$2,818
Veryslow	$15.00	6.75	$0.54		$42	$69	$150	$285	$555	$2,715
Placebo	$59.69	6.77	$0.54		$87	$114	$195	$330	$601	$2,766

x265 - 1080p - Viewer Count Breakeven - $0.04/GB

		Bitrate		**15000**						
		MBytes per hour		**6750**		**Cost per GB**		**0.04**		
		Encode/hr		**15**						
Preset	**Encode**	**Bandwidth**			**50**	**100**	**250**	**500**	**1000**	**5000**
Ultrafast	$1.45	13.13	$0.53		$28	$54	$133	$264	$527	$2,628
Superfast	$1.61	11.49	$0.46		$25	$48	$117	$231	$461	$2,300
Veryfast	$2.00	10.12	$0.40		$22	$42	$103	$204	$407	$2,026
faster	$2.70	8.44	$0.34		$20	$36	$87	$172	$340	$1,692
fast	$3.42	8.43	$0.34		$20	$37	$88	$172	$340	$1,689
Medium	$3.94	7.52	$0.30		$19	$34	$79	$154	$305	$1,509
Slow	$5.68	7.22	$0.29		$20	$35	$78	$150	$294	$1,449
Slower	$8.04	7.03	$0.28		$22	$36	$78	$149	$289	$1,413
Veryslow	$15.00	6.75	$0.27		$29	$42	$83	$150	$285	$1,365
Placebo	$59.69	6.77	$0.27		$73	$87	$127	$195	$330	$1,413

x264 - Viewer Count Breakeven - $0.02/GB

As bandwidth costs drop, encoding cost matters longer (but still not that long)

		Bitrate		15000						
		MBytes per hour		6750		Cost per GB		0.02		
		Encode/hr		15						
Preset	Encode	Bandwidth			50	100	250	500	1000	5000
Ultrafast	$1.45	13.13	$0.26		$15	$28	$67	$133	$264	$1,315
Superfast	$1.61	11.49	$0.23		$13	$25	$59	$117	$231	$1,151
Veryfast	$2.00	10.12	$0.20		$12	$22	$53	$103	$204	$1,014
faster	$2.70	8.44	$0.17		$11	$20	$45	$87	$172	$847
fast	$3.42	8.43	$0.17		$12	$20	$46	$88	$172	$846
Medium	$3.94	7.52	$0.15		$11	$19	$42	$79	$154	$756
Slow	$5.68	7.22	$0.14		$13	$20	$42	$78	$150	$728
Slower	$8.04	7.03	$0.14		$15	$22	$43	$78	$149	$711
Veryslow	$15.00	6.75	$0.14		$22	$29	$49	$83	$150	$690
Placebo	$59.69	6.77	$0.14		$66	$73	$94	$127	$195	$736

x264 - Viewer Count Breakeven - $0.02/GB

As bandwidth costs drop, encoding cost matters longer (but still not that long)

		Bitrate		15000						
		MBytes per hour		6750		Cost per GB		0.02		
		Encode/hr		15						
Preset	Encode	Bandwidth			50	100	250	500	1000	5000
Ultrafast	$1.45	13.13	$0.26		$15	$28	$67	$133	$264	$1,315
Superfast	$1.61	11.49	$0.23		$13	$25	$59	$117	$231	$1,151
Veryfast	$2.00	10.12	$0.20		$12	$22	$53	$103	$204	$1,014
faster	$2.70	8.44	$0.17		$11	$20	$45	$87	$172	$847
fast	$3.42	8.43	$0.17		$12	$20	$46	$88	$172	$846
Medium	$3.94	7.52	$0.15		$11	$19	$42	$79	$154	$756
Slow	$5.68	7.22	$0.14		$13	$20	$42	$78	$150	$728
Slower	$8.04	7.03	$0.14		$15	$22	$43	$78	$149	$711
Veryslow	$15.00	6.75	$0.14		$22	$29	$49	$83	$150	$690
Placebo	$59.69	6.77	$0.14		$66	$73	$94	$127	$195	$736

Default

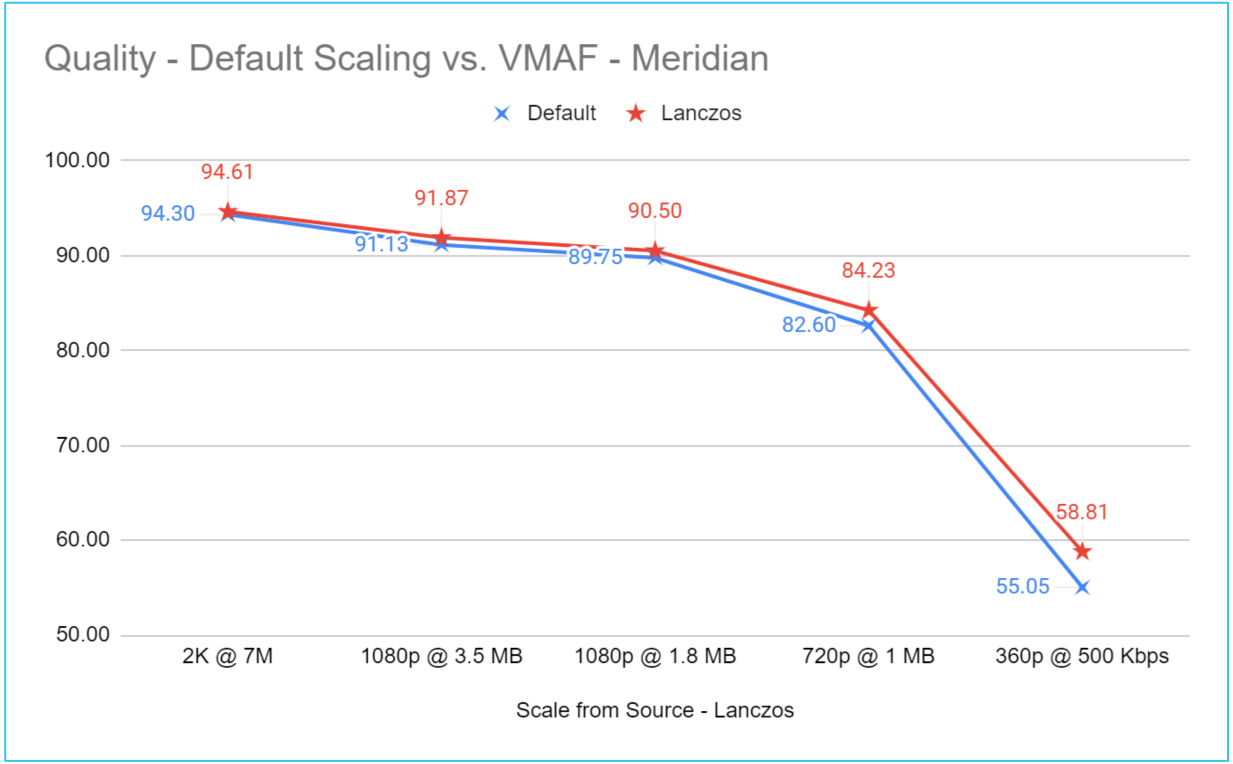
Scaling with Lanczos for Lower Rungs

https://bit.ly/ffmpeg_scaling

- FFmpeg default scaling is bilinear
- Lanczos gives slightly higher quality in lower rungs
  - **(-vf scale=1280×720 -sws_flags lanczos)**
- No impact on throughput

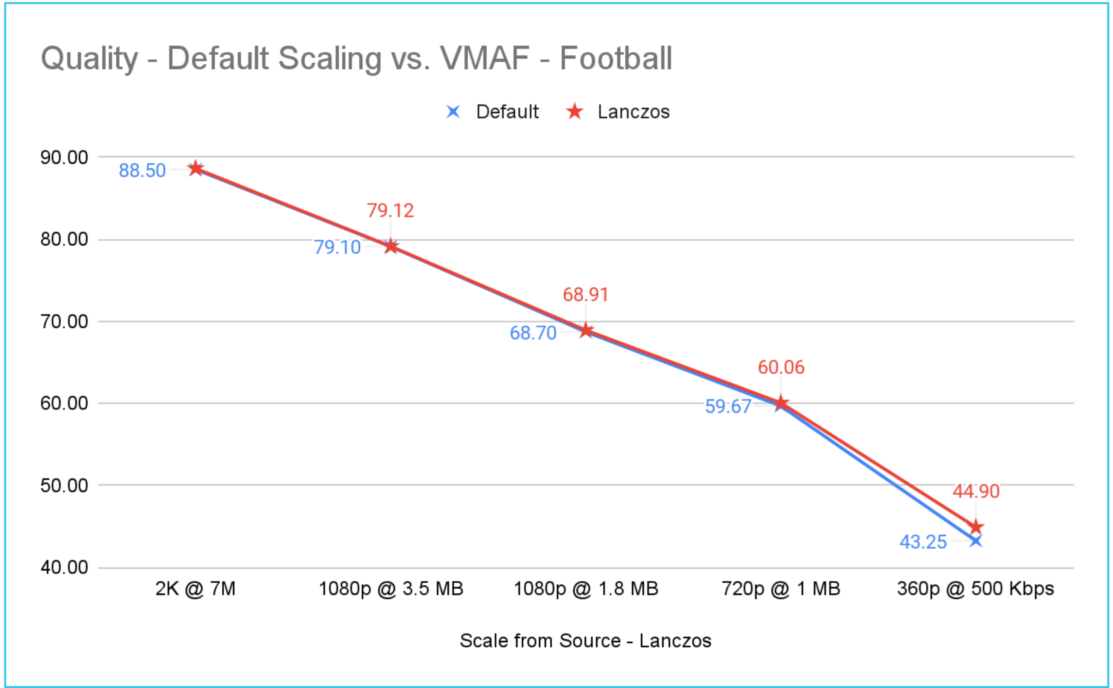
Scaling - Meridian

	Default	Lanczos
2K @ 7M	94.30	94.61
1080p @ 3.5 MB	91.13	91.87
1080p @ 1.8 MB	89.75	90.50
720p @ 1 MB	82.60	84.23
360p @ 500 Kbps	55.05	58.81



Scaling - Football

VMAF	Default	Lanczos
2K @ 7M	88.50	88.62
1080p @ 3.5 MB	79.10	79.12
1080p @ 1.8 MB	68.70	68.91
720p @ 1 MB	59.67	60.06
360p @ 500 Kbps	43.25	44.90



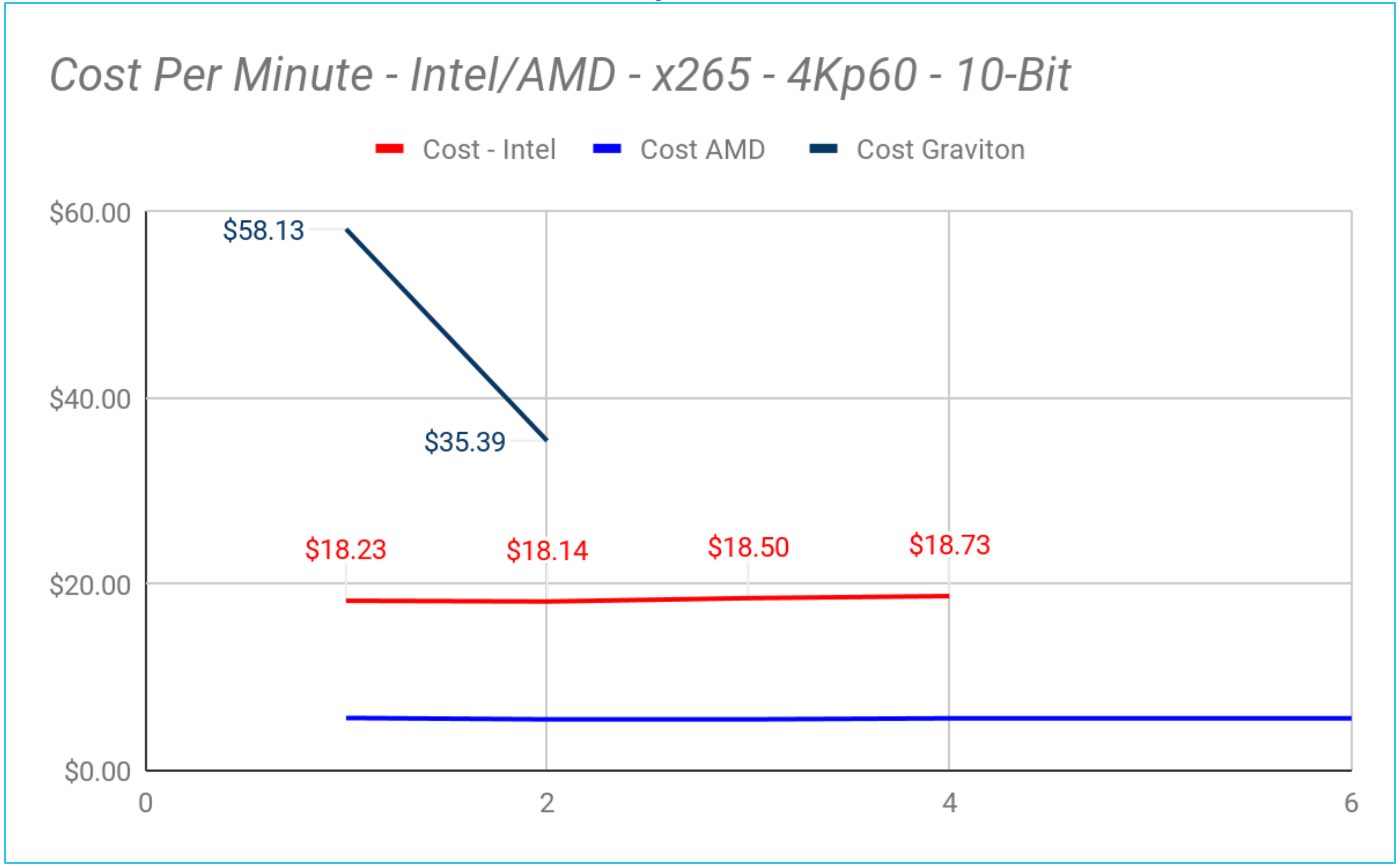
Scaling

- Not a major difference, but no downside

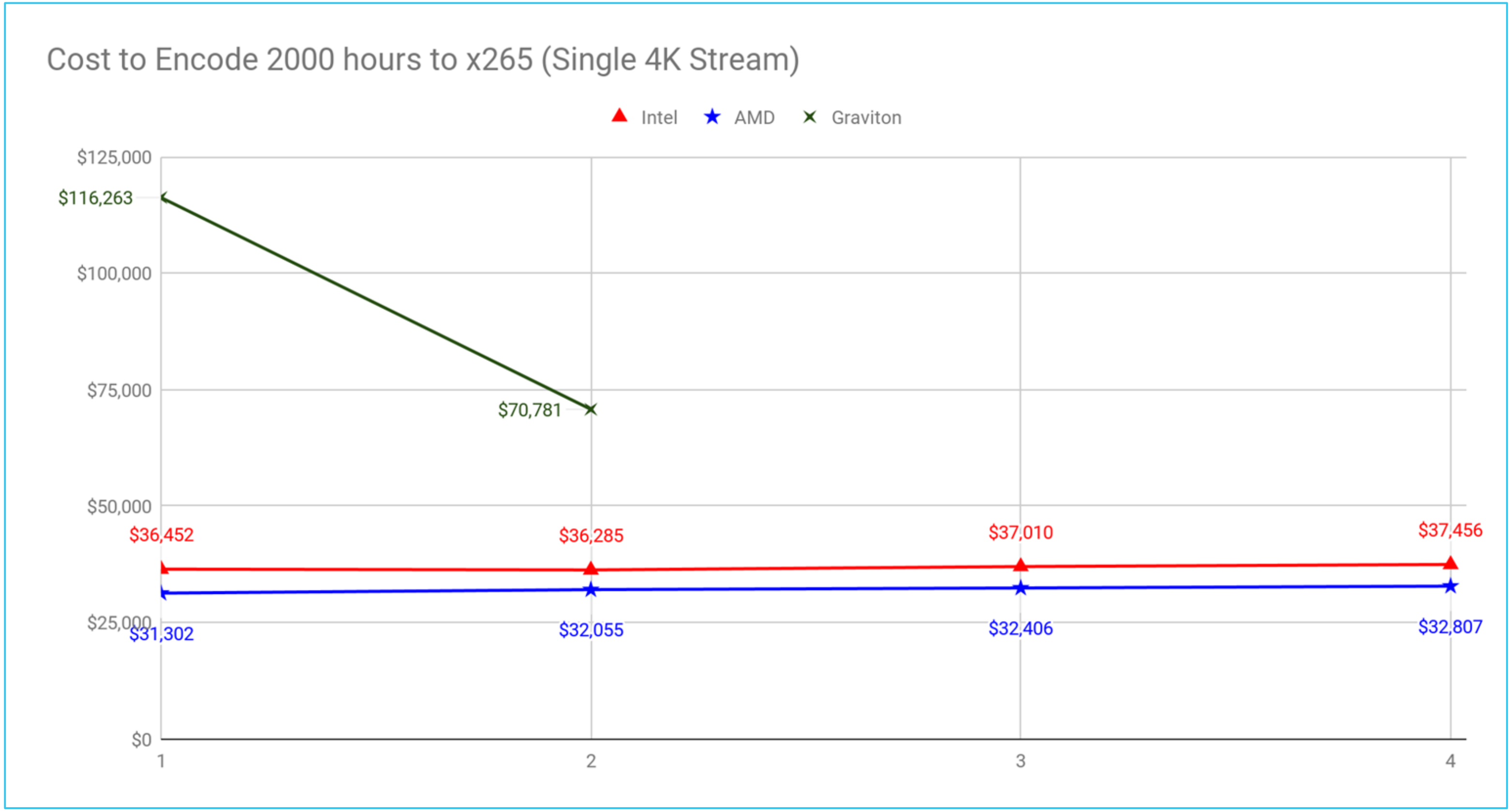
Best Instance for x265

- Three types of instances on AWS
  - Intel (c6i.xlarge) - Compute/Intel
  - AMD (c6a.xlarge) - Compute/AMD
  - AWS - Graviton - (c7g.xlarge) - Compute Graviton
- Which encodes most efficiently?
- Test methodology

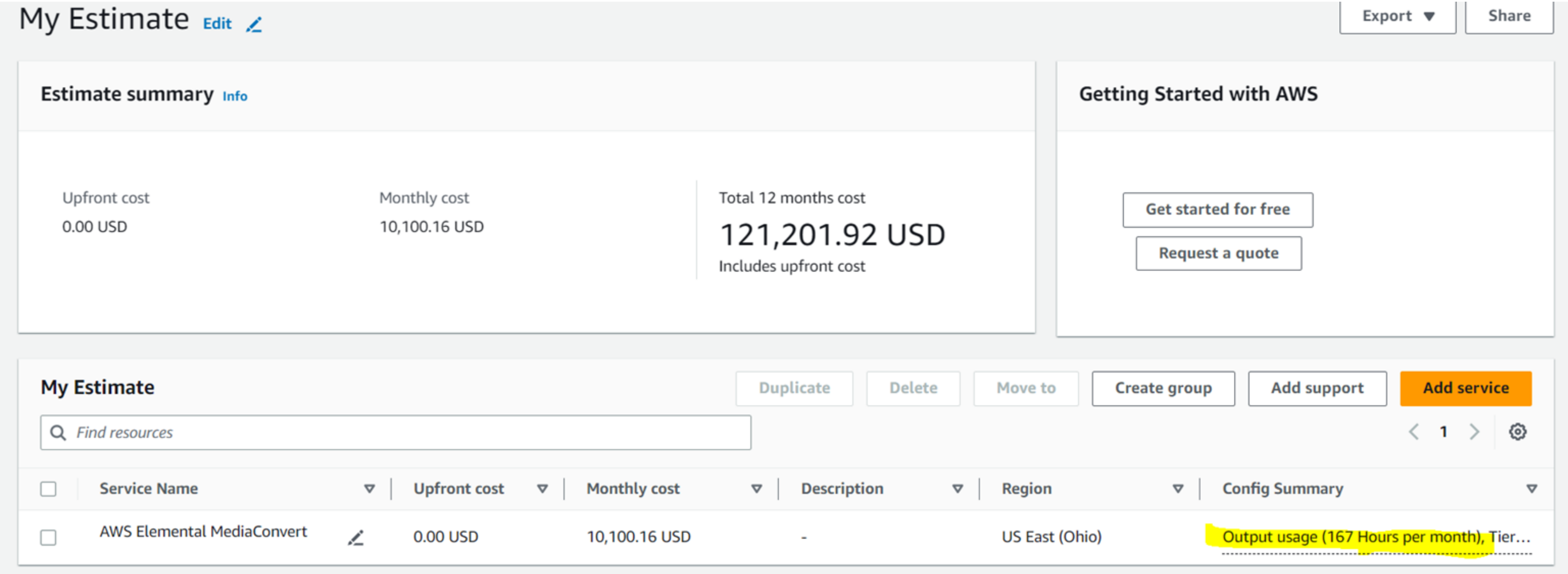
The Winner Is - For x265 1080p - AMD



The Winner Is - For x265 4K - 10bit



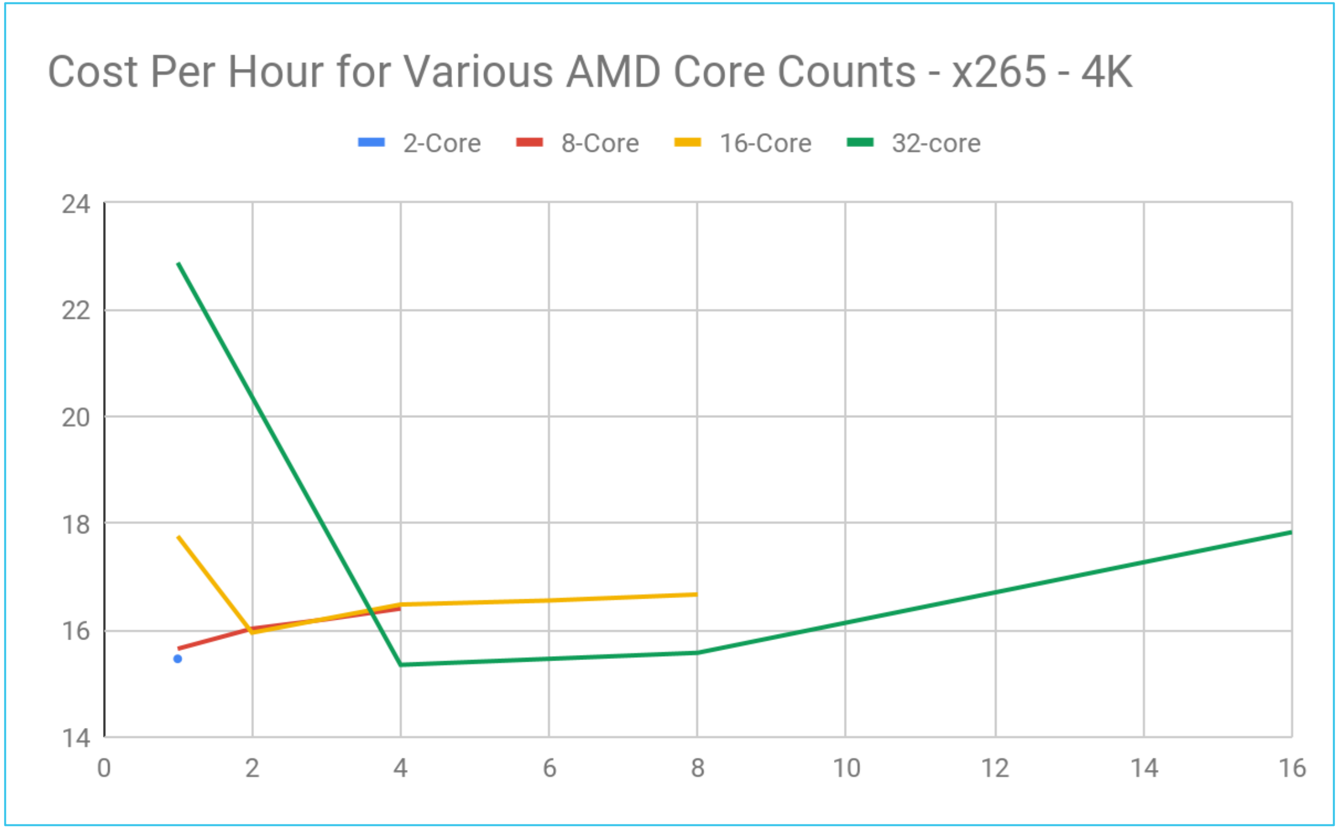
Reality Check: MediaConvert Pricing - 2000 hours AVC HQ



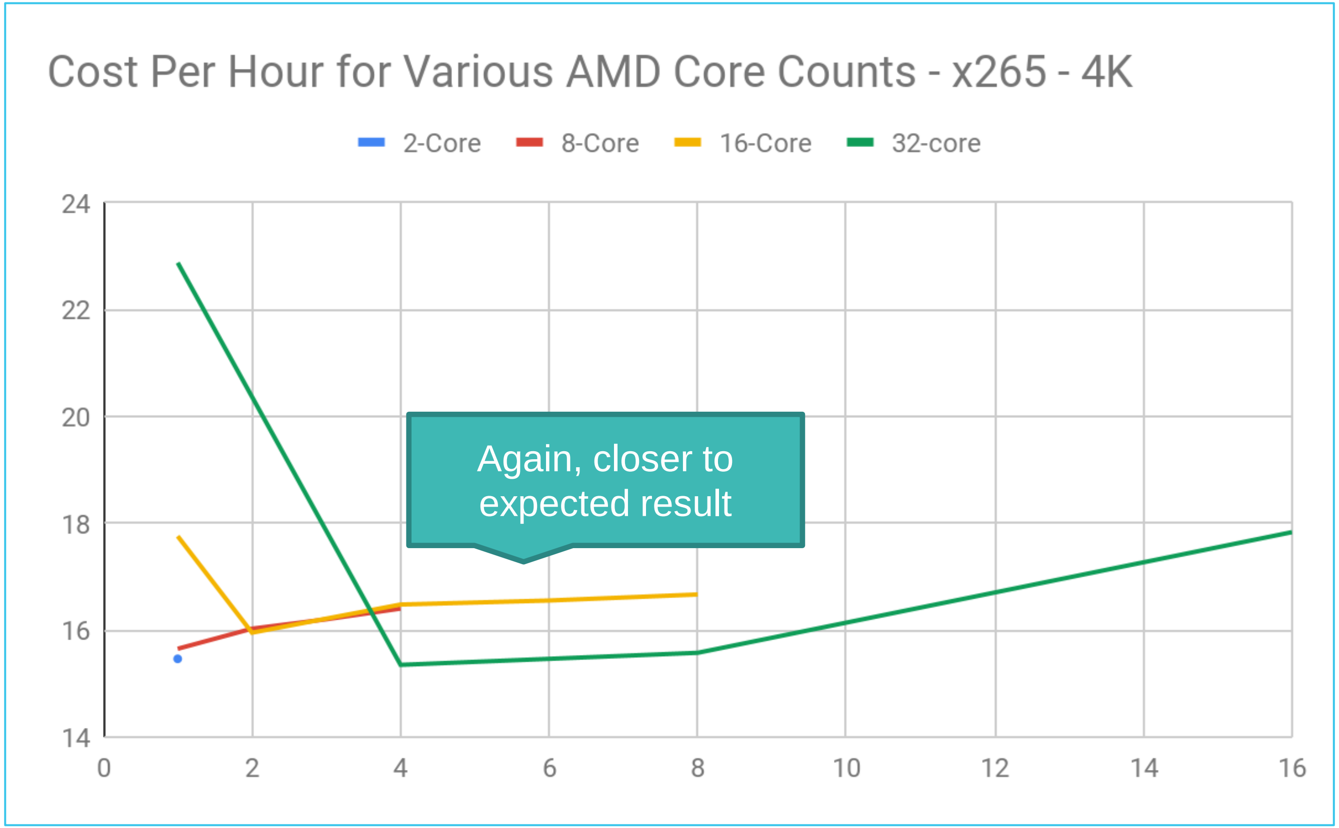
Encoding String

```
ffmpeg -y -i Football_4K60.mp4 -c:v libx265 -preset slow -x265-params keyint=120:min-keyint=120:scenecut=0:bitrate=12500K:pass=1  -f mp4 /dev/null

ffmpeg -y -i Football_4K60.mp4 -c:v libx265 -preset slow -x265-params keyint=120:min-keyint=120:scenecut=0:bitrate=12500K:vbv-maxrate=25000K:vbv-bufsize=25000K:pass=2 Football_4K_output.mp4
```

Most Efficient CPU Core Count?



Desktop – 64-core Intel



Bonus Content - What I learned about AWS

- Caveat:
  - I am not an expert in running a cloud encoding facility
  - Sharing random data points you might find useful
- One instance almost never delivers best performance
- Different instances for different jobs
- Best performance varies by codec
  - x264 - gets slightly more efficient with more jobs
  - x265 - reaches peak and then drops slowly

One Instance Never Best Performance 32-core

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3198	ubuntu	20	0	3698044	1.2g	16264	S	1767	2.0	1:41.07	ffmpeg

Single instance = 1767%/3200%

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3435	ubuntu	20	0	4224508	1.6g	16124	S	541.7	2.5	1:25.65	ffmpeg
3436	ubuntu	20	0	4224536	1.6g	16172	S	522.0	2.5	1:25.59	ffmpeg
3433	ubuntu	20	0	4224512	1.6g	16380	S	507.0	2.5	1:24.36	ffmpeg
3434	ubuntu	20	0	4224512	1.6g	16372	S	493.7	2.5	1:22.31	ffmpeg

Four instances = 2063%/3200%

One Instance Never Best Performance 32-core

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
11425	ubuntu	20	0	4224504	1.6g	16392	S	316.7	2.5	1:16.34	ffmpeg
11429	ubuntu	20	0	4224536	1.6g	15820	S	302.7	2.5	1:16.64	ffmpeg
11430	ubuntu	20	0	4224576	1.6g	16392	S	302.0	2.5	1:16.96	ffmpeg
11422	ubuntu	20	0	4224512	1.6g	16392	S	299.3	2.5	1:16.25	ffmpeg
11420	ubuntu	20	0	4224548	1.6g	16392	S	296.3	2.5	1:16.54	ffmpeg
11431	ubuntu	20	0	4224524	1.6g	16392	S	295.3	2.5	1:15.93	ffmpeg
11428	ubuntu	20	0	4224512	1.6g	15896	S	287.7	2.5	1:15.83	ffmpeg
11427	ubuntu	20	0	4224616	1.6g	16376	S	286.0	2.5	1:16.33	ffmpeg

Eight instances = 2400%/3200%

One Instance Never Best Performance 32-core

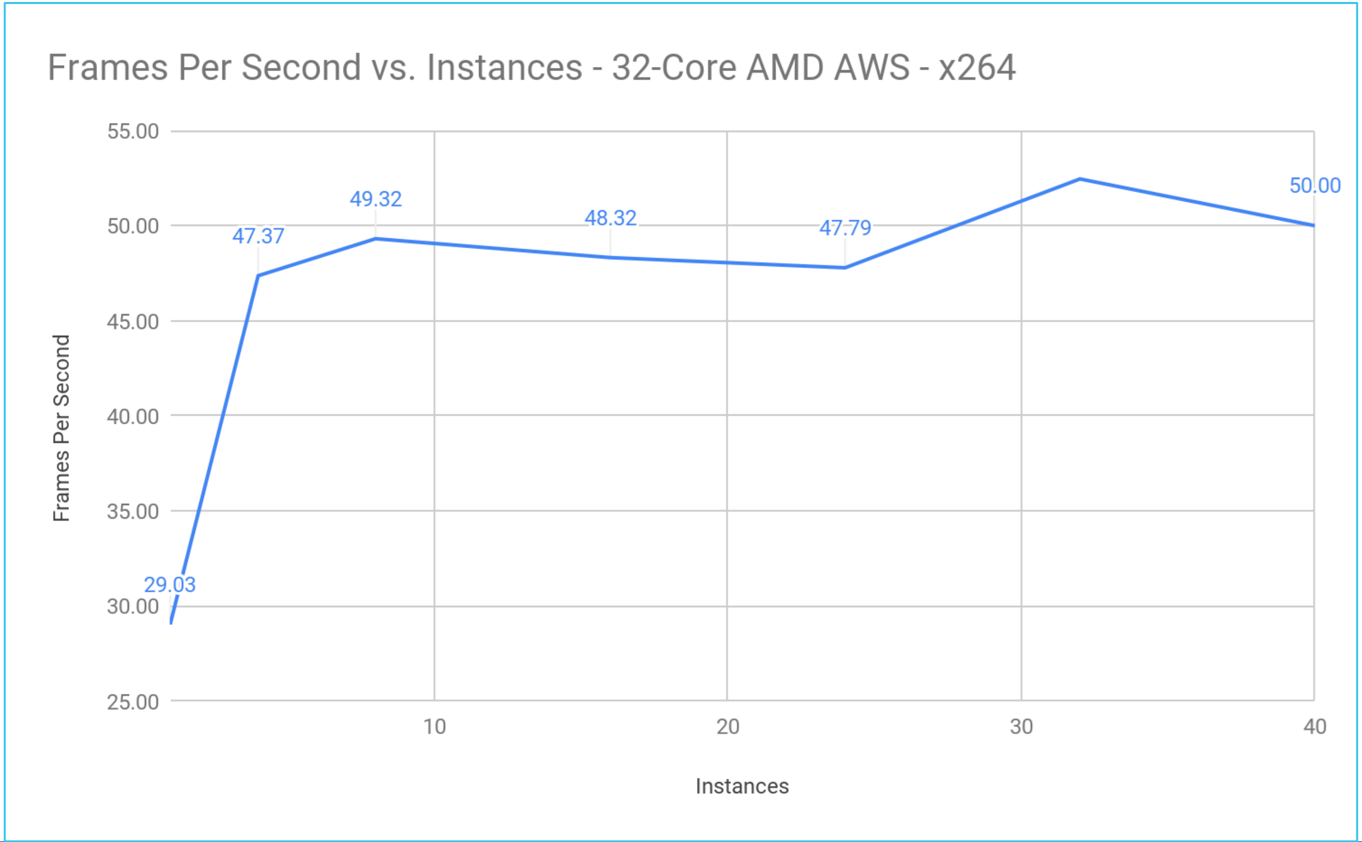
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
12654	ubuntu	20	0	4224376	1.5g	16328	S	199.3	2.5	0:37.61	ffmpeg
12644	ubuntu	20	0	4224372	1.5g	16328	S	198.7	2.5	0:39.35	ffmpeg
12640	ubuntu	20	0	4224372	1.5g	16320	S	196.7	2.5	0:38.29	ffmpeg
12643	ubuntu	20	0	4224532	1.5g	16328	S	192.0	2.5	0:37.34	ffmpeg
12657	ubuntu	20	0	4224372	1.5g	16328	S	190.7	2.5	0:36.99	ffmpeg
12650	ubuntu	20	0	4224372	1.5g	16328	S	188.3	2.5	0:37.39	ffmpeg
12652	ubuntu	20	0	4224376	1.5g	16328	S	187.3	2.5	0:37.49	ffmpeg
12636	ubuntu	20	0	4224376	1.5g	16312	S	187.0	2.5	0:36.74	ffmpeg
12656	ubuntu	20	0	4224376	1.5g	16328	S	184.3	2.5	0:37.81	ffmpeg
12648	ubuntu	20	0	4224532	1.5g	16328	S	184.0	2.5	0:37.83	ffmpeg
12659	ubuntu	20	0	4224372	1.5g	16328	S	180.3	2.5	0:36.73	ffmpeg
12634	ubuntu	20	0	4224376	1.5g	16328	S	180.0	2.5	0:37.24	ffmpeg
12661	ubuntu	20	0	4224372	1.5g	16328	S	179.0	2.5	0:37.57	ffmpeg
12642	ubuntu	20	0	4224376	1.5g	16308	S	178.3	2.5	0:37.09	ffmpeg
12660	ubuntu	20	0	4224376	1.5g	16328	S	178.0	2.5	0:36.79	ffmpeg
12647	ubuntu	20	0	4224376	1.5g	16328	S	175.7	2.5	0:36.63	ffmpeg

16 instances = 2960%/3200%

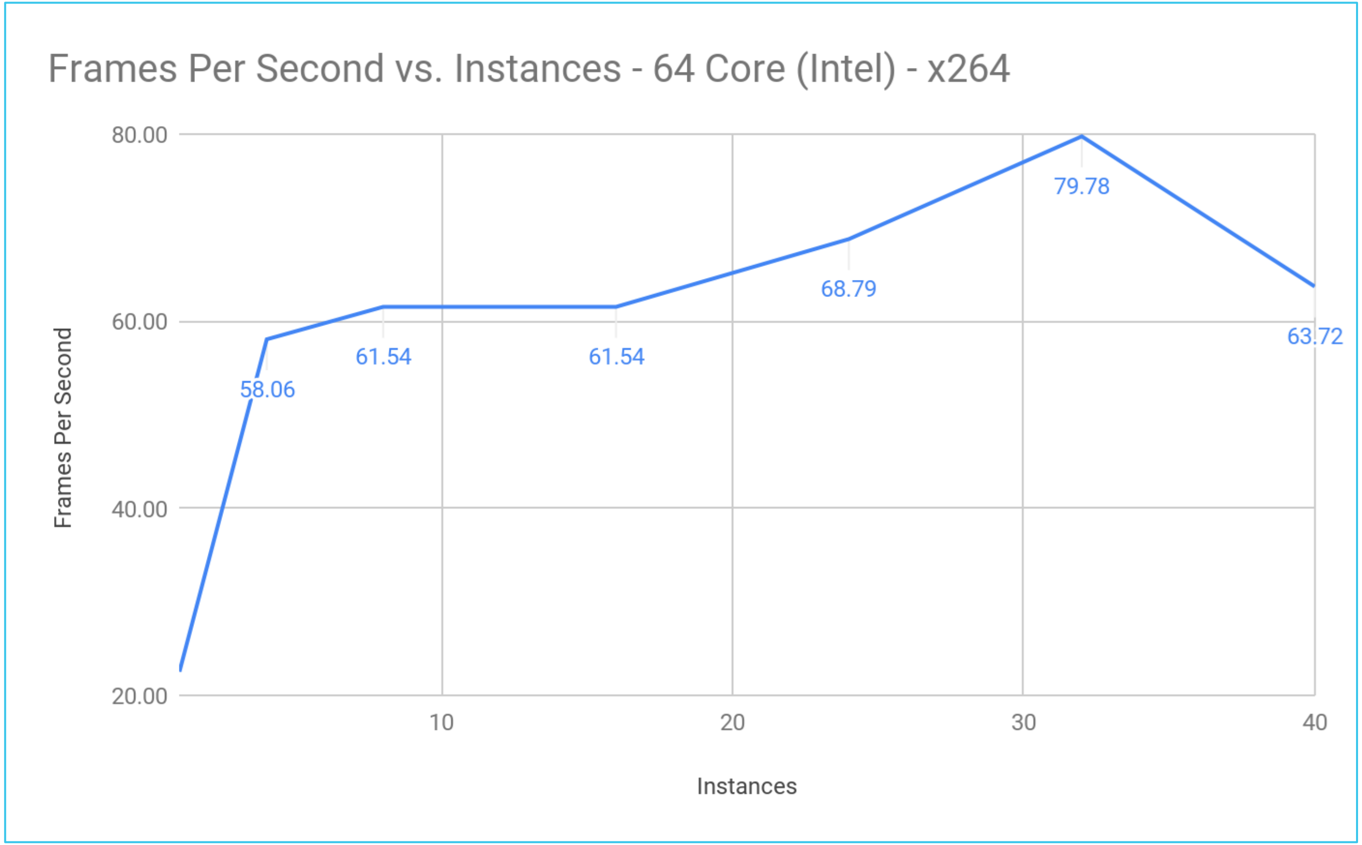
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
15076	ubuntu	20	0	4224376	1.5g	16128	S	144.6	2.5	0:24.71	ffmpeg
15097	ubuntu	20	0	4224372	1.5g	16328	S	143.9	2.5	0:25.97	ffmpeg
15103	ubuntu	20	0	4224376	1.5g	16328	S	140.9	2.5	0:24.38	ffmpeg
15063	ubuntu	20	0	4224372	1.5g	16268	R	140.6	2.5	0:24.35	ffmpeg
15068	ubuntu	20	0	4224376	1.5g	16044	S	139.9	2.5	0:24.72	ffmpeg
15082	ubuntu	20	0	4224372	1.5g	16328	R	137.0	2.5	0:24.24	ffmpeg
15081	ubuntu	20	0	4224372	1.5g	16328	S	135.6	2.5	0:24.50	ffmpeg
15101	ubuntu	20	0	4224376	1.5g	16328	S	134.7	2.5	0:24.83	ffmpeg
15073	ubuntu	20	0	4224372	1.5g	16328	S	134.0	2.5	0:24.31	ffmpeg
15074	ubuntu	20	0	4224376	1.5g	16084	S	134.0	2.5	0:25.21	ffmpeg
15102	ubuntu	20	0	4224376	1.5g	16328	S	134.0	2.5	0:25.75	ffmpeg
15104	ubuntu	20	0	4224372	1.5g	16328	S	130.7	2.5	0:24.11	ffmpeg
15069	ubuntu	20	0	4224376	1.5g	16120	S	130.4	2.5	0:24.07	ffmpeg
15078	ubuntu	20	0	4224376	1.5g	16328	S	129.0	2.5	0:24.15	ffmpeg
15094	ubuntu	20	0	4224376	1.5g	16328	S	128.1	2.5	0:24.95	ffmpeg
15100	ubuntu	20	0	4224376	1.5g	16328	S	128.1	2.5	0:26.10	ffmpeg
15086	ubuntu	20	0	4224376	1.5g	16328	S	127.7	2.5	0:25.58	ffmpeg
15089	ubuntu	20	0	4224372	1.5g	16328	S	126.1	2.5	0:23.84	ffmpeg
15061	ubuntu	20	0	4224372	1.5g	16280	S	125.4	2.5	0:23.55	ffmpeg
15067	ubuntu	20	0	4224372	1.5g	16228	S	123.8	2.5	0:24.08	ffmpeg
15092	ubuntu	20	0	4224376	1.5g	16328	S	121.8	2.5	0:24.61	ffmpeg
15071	ubuntu	20	0	4224372	1.5g	16328	S	119.8	2.5	0:23.94	ffmpeg
15090	ubuntu	20	0	4224376	1.5g	16328	S	119.8	2.5	0:24.31	ffmpeg
15084	ubuntu	20	0	4224372	1.5g	16328	S	119.5	2.5	0:25.58	ffmpeg

24 instances = ~3120%/3200%

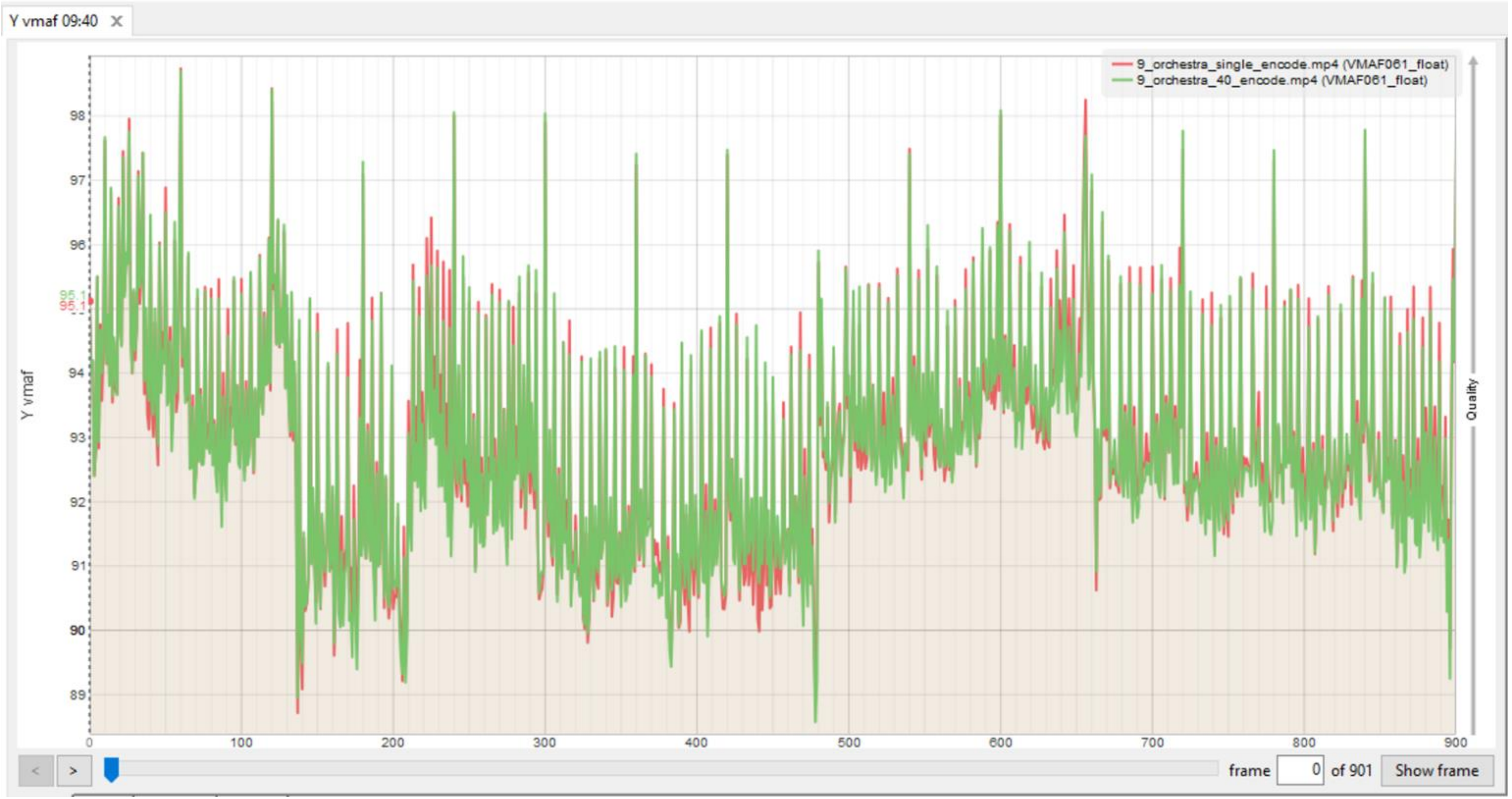
Frames Per Second - 32-core AWS AMD Instance



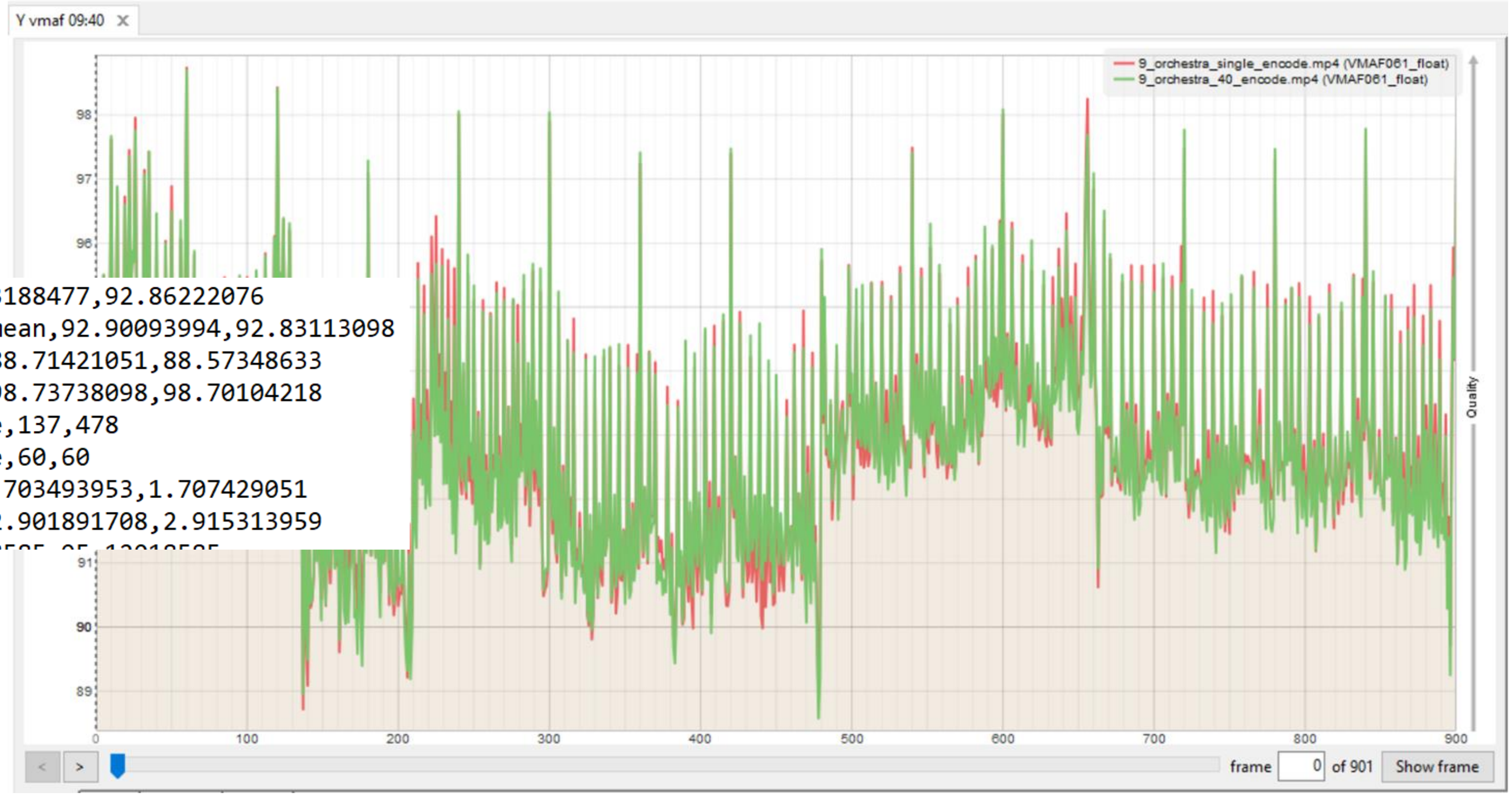
Frames Per Second - 64-core Intel Workstation



Quality Delta - Single vs. 40 encode



Quality Delta - Single vs. 40 encode




```
mean,92.93188477,92.86222076
harmonic mean,92.90093994,92.83113098
min. val,88.71421051,88.57348633
max. val,98.73738098,98.70104218
min. frame,137,478
max. frame,60,60
std dev,1.703493953,1.707429051
variance,2.901891708,2.915313959
```

Varies by Encode - 4K/60 10-bit x265

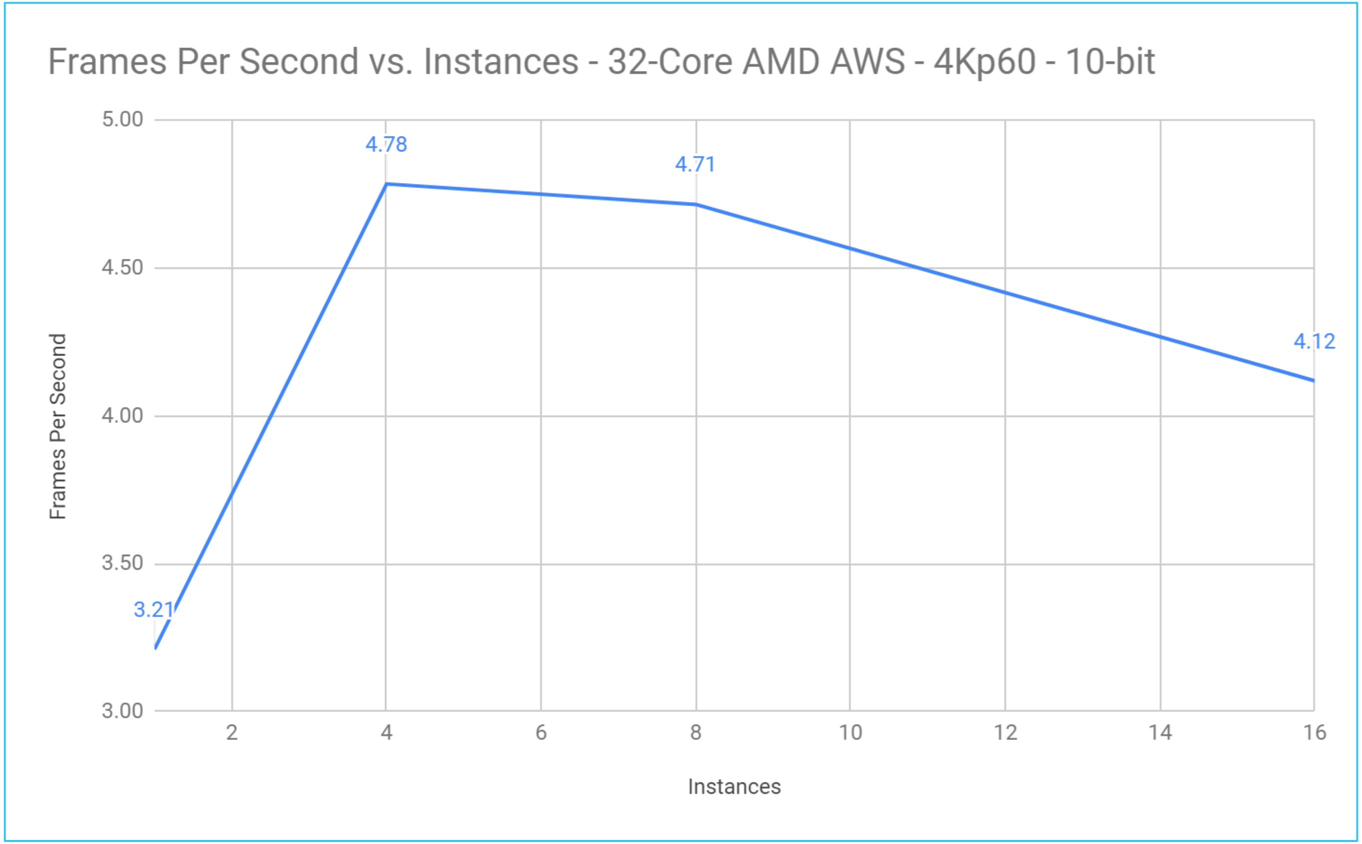
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
4761	ubuntu	20	0	8534308	4.1g	18184	S	1592	6.6	4:21.82	ffmpeg

Single instance = 1592%/3200%

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
4912	ubuntu	20	0	8527964	4.1g	18160	S	950.2	6.6	2:14.65	ffmpeg
4913	ubuntu	20	0	8527700	4.1g	17796	S	755.1	6.6	2:01.54	ffmpeg
4914	ubuntu	20	0	8507232	4.1g	17820	S	740.5	6.6	2:04.54	ffmpeg
4911	ubuntu	20	0	8527696	4.1g	18156	S	733.9	6.6	2:04.40	ffmpeg

Four instances = 3178%/3200%

FPS - 32-core AWS AMD - x265 4Kp60 10-bit



FPS - 32-core AWS AMD - x265 - 1080p 8-bit