BEST PRACTICES FOR ENCODING H.264 AND HEVC

Jan Ozer, Streaming Learning Center
Jan.ozer@streaminglearningcenter.com

Best Practices for H.264 and HEVC Encoding

- H.264
  - Choosing the optimal GOP size
  - Benefits of a variable GOP
  - Bitrate control
  - Choosing a preset
  - Choosing the optimal thread count
  - Best AWS CPU
- HEVC
  - Choosing the optimal GOP size
  - Benefits of a variable GOP
  - Bitrate control
  - Choosing a preset
  - Choosing the optimal thread count
  - Working with Wavefront Parallel Processing
- Both
  - Optimizing scaling for lower rung production

Fundamentals

- Top rung target quality
  - Premium content – 93 – 95 VMAF
  - UGC – 85 – 92 VMAF
  - Getting there:
    - Choose configuration options
    - Adjust bitrate to hit the target
- VMAF
  - Measure harmonic mean
  - And low-frame (potential for transient quality problems)
- Just noticeable difference (how much does a difference matter?)
  - Greater than 50% of viewer notice
  - ~ 3 VMAF point
- Most differences discussed here will be much less
  - Still, .4 VMAF here, .6 there, pretty soon you’re close to a JND
  - Plus – the target is 95 (or whatever)
  - After a few adjustments, you will have to increase the bitrate to achieve the target (boosting your bandwidth costs)

H.264 Agenda

- Choosing the optimal GOP size
  - Benefits of a variable GOP
- Bitrate control
- Choosing a preset
- Choosing the optimal thread count
- Best AWS CPU

Uber Best Practice

- Content Adaptive Encoding is the ultimate best practice
  - None of the techniques discussed herein can touch CAE as an optimization technique

Best Practice 1 – Choose Longest Possible GOP Size

- What: GOP size (I-frame interval) is a key config option in all encodes
- Historical
  - Very small (like .5 second) for MPEG-2
  - Very long (10-20 seconds) for downloaded video
  - Typically, 2-5 seconds for adaptive bitrate video
    - Must divide evenly into segment size
- Question
  - How does GOP size impact quality
- Test – 13 files in 4 categories
  - Entertainment
  - Sports
  - Animation
  - Office

Best Practice 1: Longer is Better

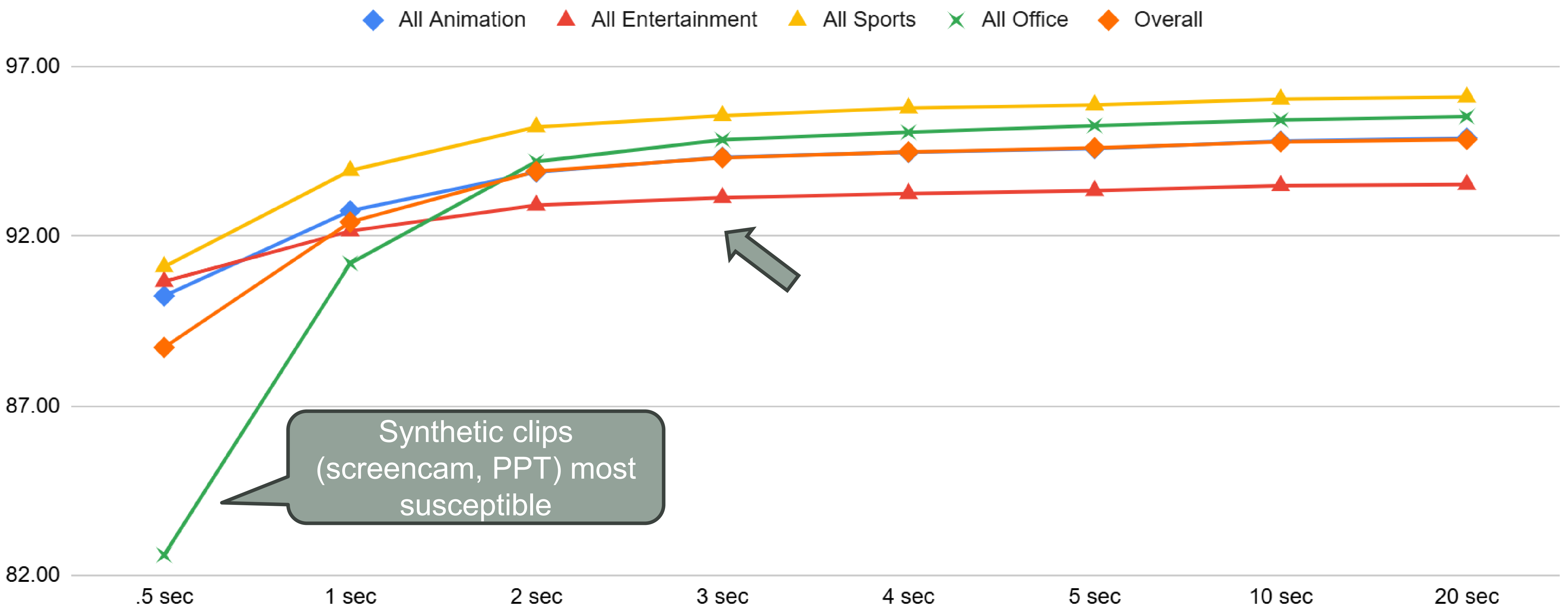
Overall - H.264	.5 sec	1 sec	2 sec	3 sec	4 sec	5 sec	10 sec	20 sec
All Animation	90.25	92.75	93.90	94.33	94.48	94.59	94.81	94.90
All Entertainment	90.67	92.16	92.92	93.14	93.26	93.35	93.50	93.53
All Sports	91.11	93.94	95.23	95.56	95.78	95.88	96.05	96.11
All Office	82.61	91.21	94.21	94.85	95.07	95.26	95.43	95.53
Overall	**88.73**	**92.42**	**93.92**	**94.32**	**94.49**	**94.61**	**94.79**	**94.86**
Delta from Max	**6.13**	**2.43**	**0.94**	**0.54**	**0.37**	**0.25**	**0.07**	**0.00**

Diminishing returns

- Benefit significant at lower range
- Then diminishing returns

- Key limit: must divide evenly into segment size
  - 10 second copy – 1/2/5/10
  - Why not try 10? Check for playability

VMAF Score by GOP Size - H.264



BP2: Consider Variable GOP Sizes

***Meta's David Ronca**, "The optimal GOP size is aligned to the encoder's placement of intra frames with a max spacing between 5-10 seconds. That is, let the encoder decide as much as possible.*

- So, longer GOP + GOPs at scene changes
- Need packager/player compatible with variable segment sizes

Meridian	2-sec GOP	5-sec GOP	10-sec GOP	Max Delta
VMAF	95.38	95.56	95.61	0.23
Low-Frame	79.52	82.66	82.98	3.46
TOS	**2-sec GOP**	**5-sec GOP**	**10-sec GOP**	**Delta**
VMAF	94.44	94.89	95.03	0.59
Low-Frame	69.48	74.58	76.59	7.10

I-frames at scene changes boosts low-frame score

Cost/ GB/ hours	500k Hours	1M hours	10M hours	100M hours	1B hours
$0.08	$6,866	$13,733	$137,329	NA	NA
$0.02	NA	$3,433	$34,332	$343,323	NA
$0.005	NA	NA	NA	$85,831	$858,307

https://bit.ly/variable_GOP

Best Practice 1: GOP Size

- Use the longest possible GOP size (segment size)
- Use variable GOPs/segment sizes if supported

Best Practice 2: Optimize Bitrate Control

- Data rate:
  - Assigned to file during encoding
  - Bitrate control - how encoder allocates the data rate
- Question: What's the best bitrate control technique (and how much difference in quality and throughput?)
  - CBR (constant bitrate encoding)
  - Two-pass VBR (variable bitrate encoding)
    - 150%, 200%, 400% constrained
  - Capped CRF (constant rate factor)

2-Pass VBR

```
ffmpeg -y -i input.mp4 -c:v libx264 -b:v 2M -maxrate 4M -bufsize 4M -
preset veryslow -g 60 -threads -threads 8 -pass 1 -f mp4 NUL`

ffmpeg -y -i input.mp4 -c:v libx264 -b:v 2M -maxrate 4M -bufsize 4M -
preset veryslow -g 60 -threads -threads 8 output.mp4
```

- Constrained VBR
  - Target = 1x
  - Max/VBV = 2x
    - Typically ranges from 1.1x to 4x
    - Tested 1.5x, 2x, and 4x
- Bitrates and GOP size customized for each file
  - Target ~94 VMAF
  - 2 seconds for 24, 25, 30, 60 fps

- Pros
  - Overall and low-frame quality
- Cons
  - Encoding time increase
  - Bitrate variability
  - Max frame values (deliverability)
- Use case
  - VOD

1-Pass CBR

ffmpeg -y -i input.mp4 -c:v libx264 **-b:v 2M -maxrate 2M -bufsize 2M** -preset veryslow -g 60 -threads -threads 8 output.mp4

- CBR
  - Target = 1x
  - Max/VBV = 1x
- Bitrates and GOP size customized for each file

- Pros
  - Shorter encoding time
  - Bitrate consistency
- Cons
  - Overall/low-frame quality
- Use case
  - Live

Capped CRF

```
ffmpeg -y -i input.mp4 -c:v libx264 -crf 27 -maxrate 2M -bufsize 4M -preset veryslow -g 60 -threads -threads 8 output.mp4
```

- Capped CRF
  - Target = crf value = ~ VMAF 94
  - Max = VBR/CBR target bitrate
  - VBV = 2x target
- CRF/Caps and GOP size customized for each file

- Pros
  - Reduced encoding time (single pass)
  - Bitrate reduction (form of per-title)
- Cons
  - Overall/low-frame quality
- Use case
  - Live/VOD

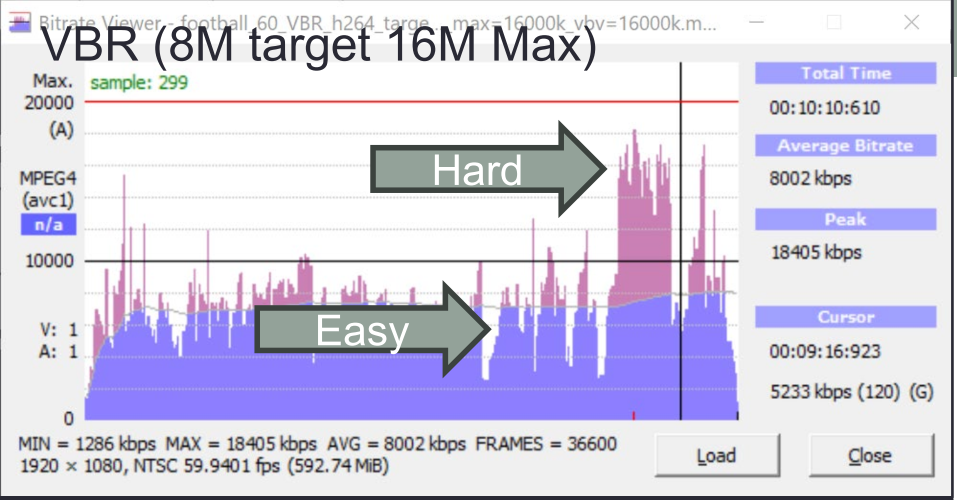
About Capped CRF

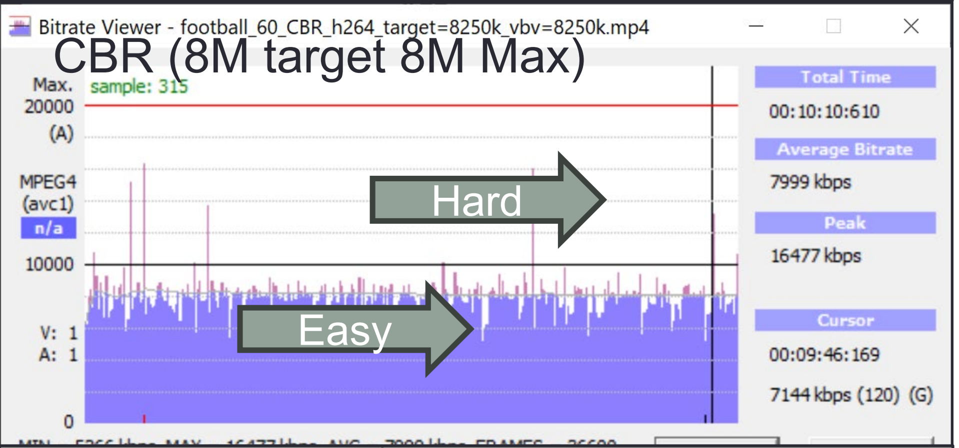
OTT VERSE | Home | Encoding | Streaming | DRM | AdTech | FFmpeg | Industry | Events | OTTV

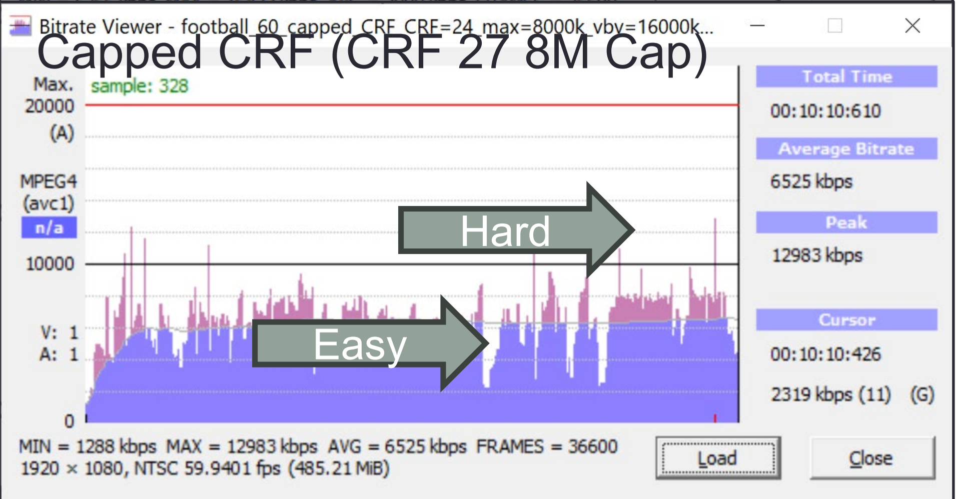
What is CBR, VBR, CRF, Capped-CRF? Rate Control Modes Explained

Jan's Corner, Compression / By Jan Ozer / July 22, 2021

https://ottverse.com/what-is-cbr-vbr-crf-capped-crf-rate-control-explained/



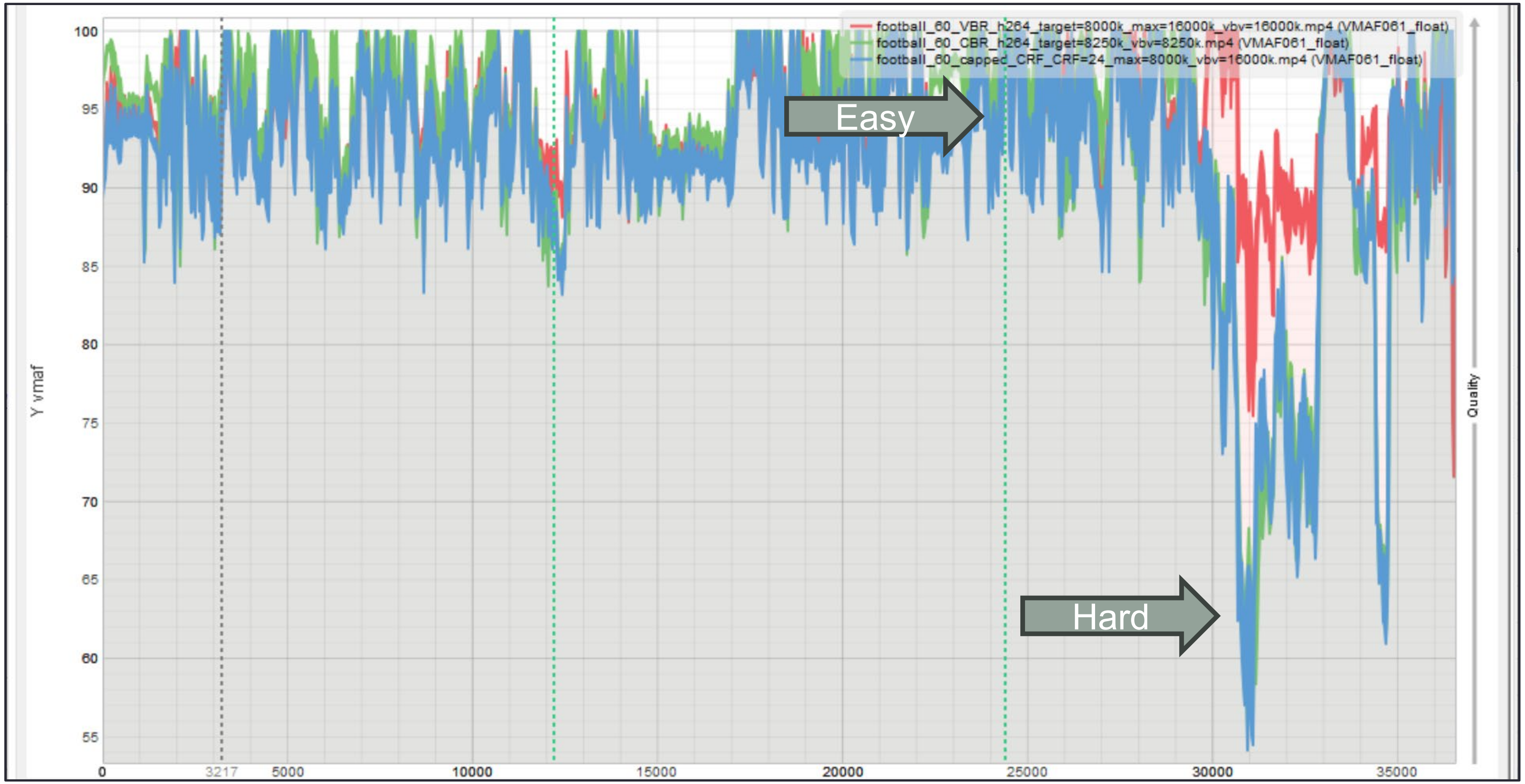


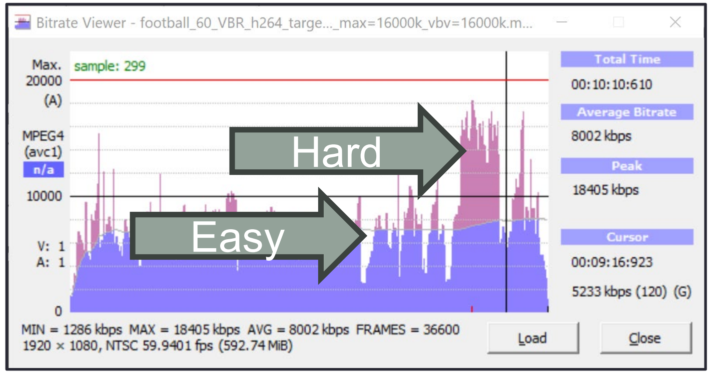


	Time	Bitrate	Max BR	VMAF	Low-Frame
VBR	64:40	8,002K	18,405K	94.10	71.57
CBR	52:52	7,999K	16,477K	92.61	55.29
Capped CRF	52:42	6,525K	12,983K	91.14	54.14

Here's What VBR's Flexibility Gives You

Red = VBR
Green = CBR
Blue = Capped CRF





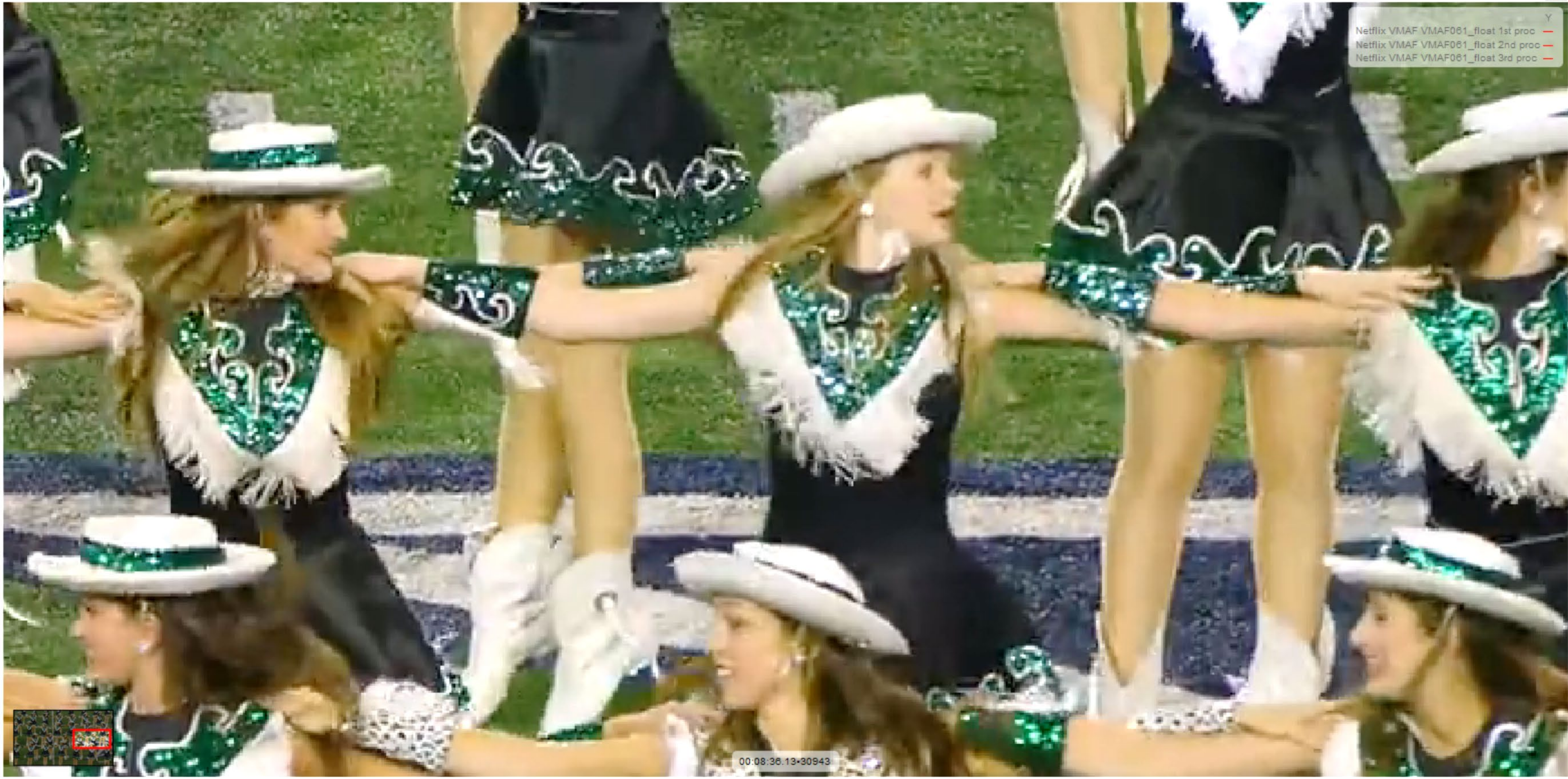
Source

D:\Numbers_2024\H.264\1080p_bitrate\60 fps sports\football_60\football_60.mp4

1920x1080

2-Pass VBR

D:\Numbers_2024\H.264\1080p_bitrate\60 fps sports\football_60\football_60_VBR_h264_target=8000k_max=16000k_vbv=16000k.mp4 1920x1080



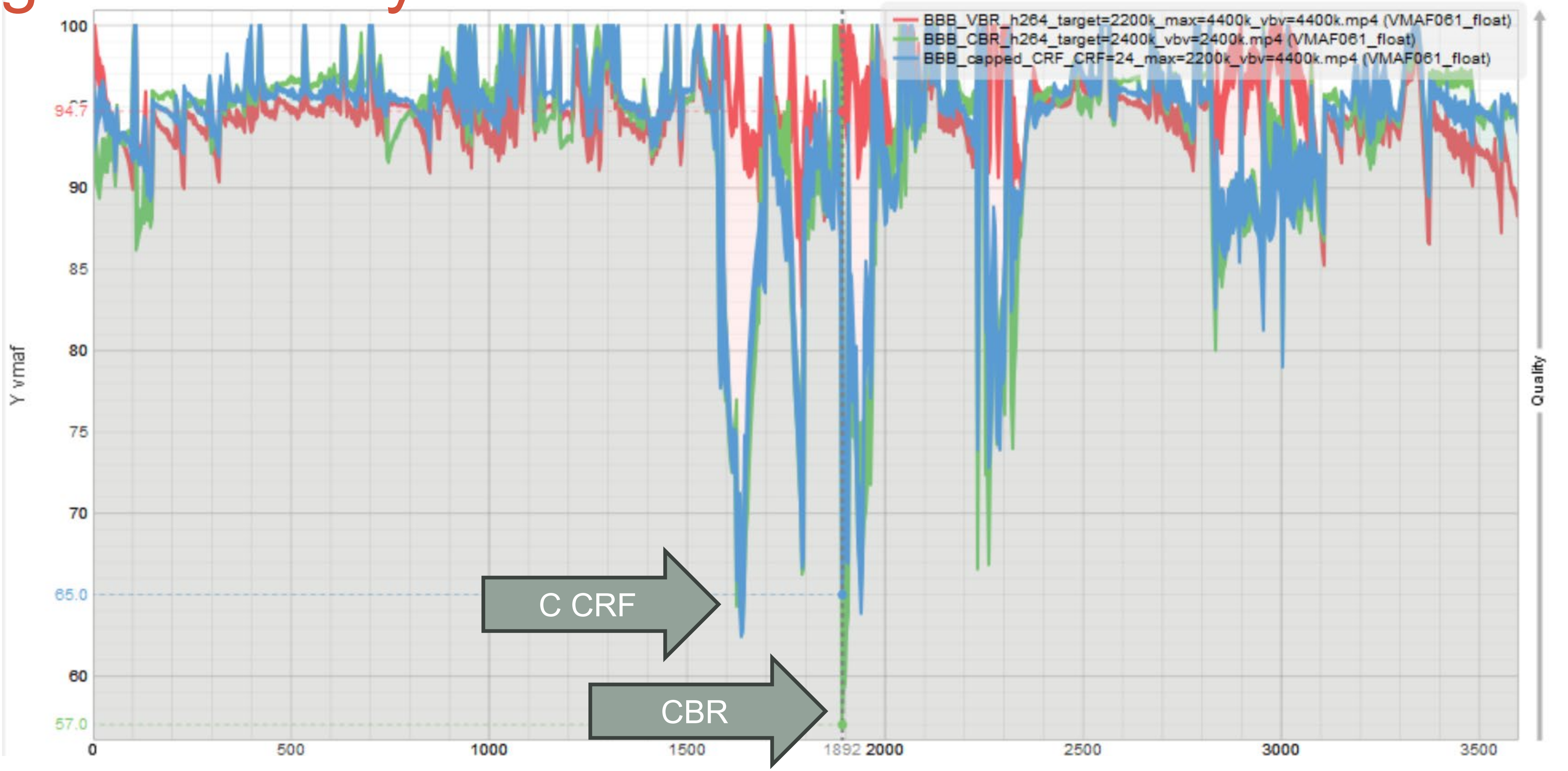
CBR

D:\Numbers_2024\H.264\1080p_bitrate\60 fps sports\football_60\football_60_CBR_h264_target=8250k_vbv=8250k.mp4 1920x1080

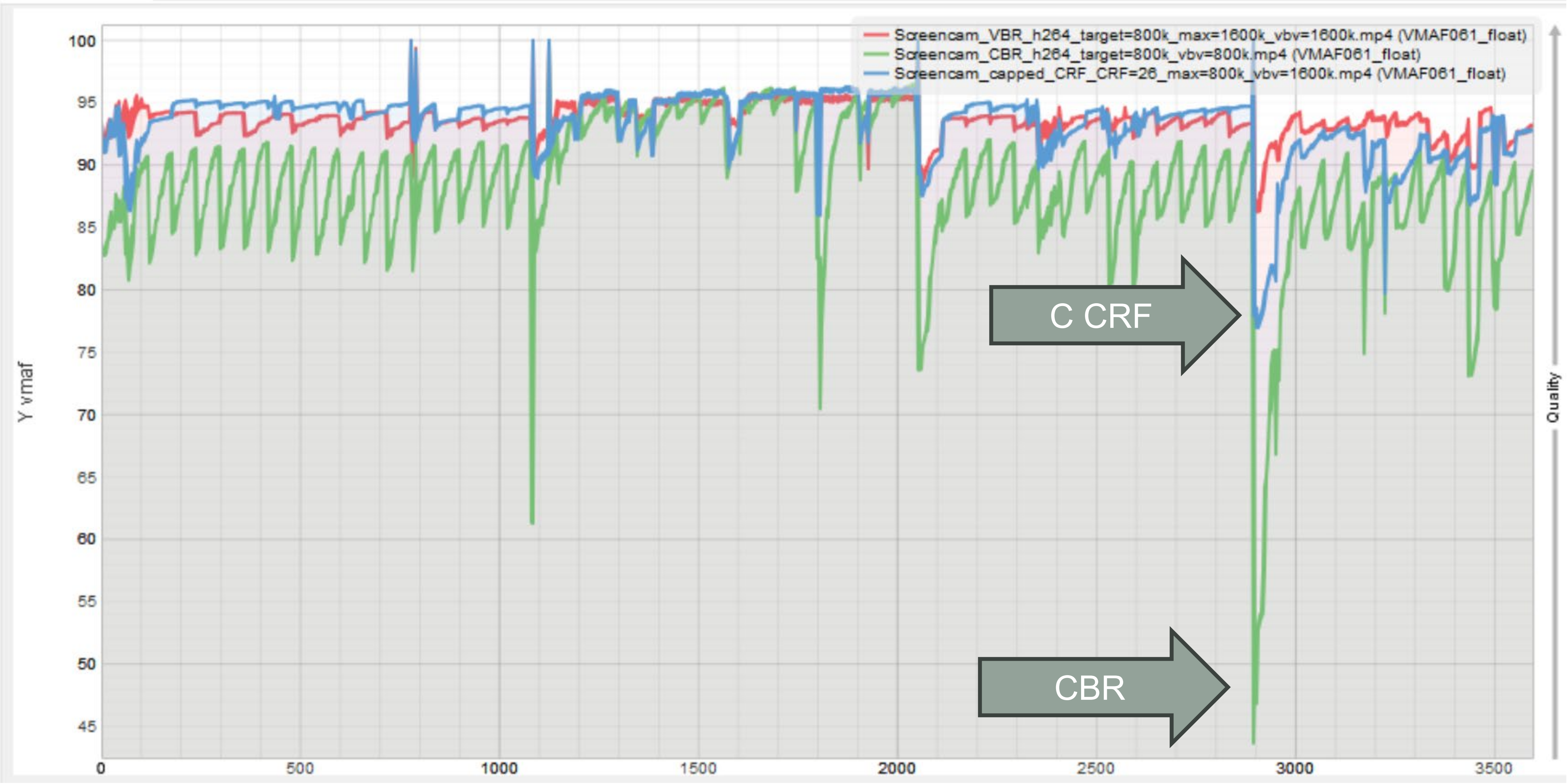
Capped CRF

D:\Numbers_2024\H.264\1080p_bitrate\60 fps sports\football_60\football_60_capped_CRF_CRF=24_max=8000k_vbv=16000k.mp4 1920x1080

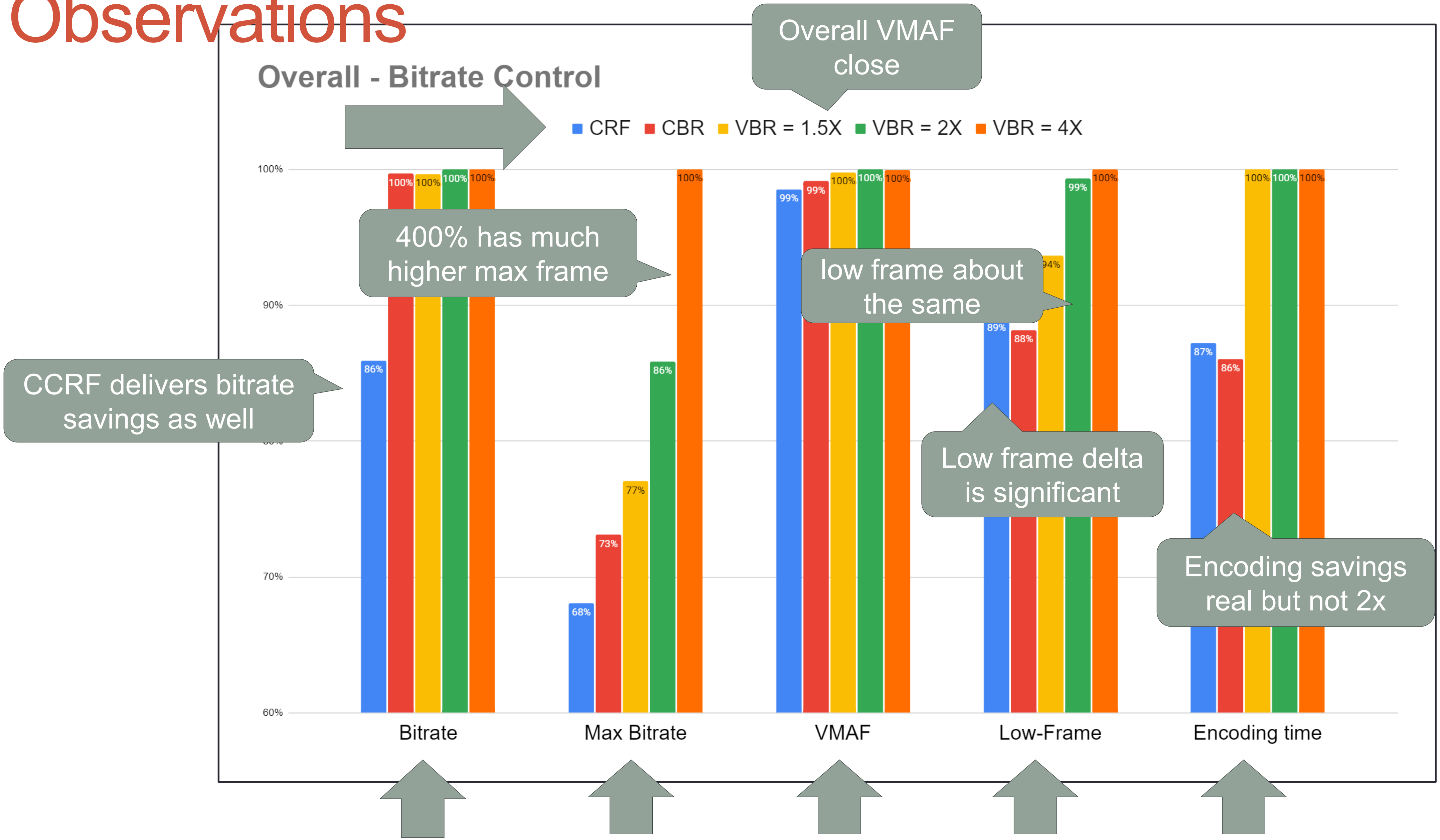
Big Buck Bunny



Office - Screencam



Observations



Capped CRF Disclaimer

- Typically used instead of fixed ladder (like Apple's)
- So "cap" is typically much higher, like 7800 kbps
  - Lots of potential bitrate reduction baked in
- In these tests, cap was same as CBR/VBR (~95 VMAF)
  - So, very little room to generate savings
  - Mostly controlled by the cap, not CRF
  - Cap very stringently applied, which degrades both overall and low-frame scores
  - Useful for comparison purposes, but not a fair look

16:9 aspect ratio	H.264/AVC	Frame rate
416 x 234	145	≤ 30 fps
640 x 360	365	≤ 30 fps
768 x 432	730	≤ 30 fps
768 x 432	1100	≤ 30 fps
960 x 540	2000	Same as source
1280 x 720	3000	Same as source
1280 x 720	4500	Same as source
1920 x 1080	6000	Same as source
1920 x 1080	7800	Same as source

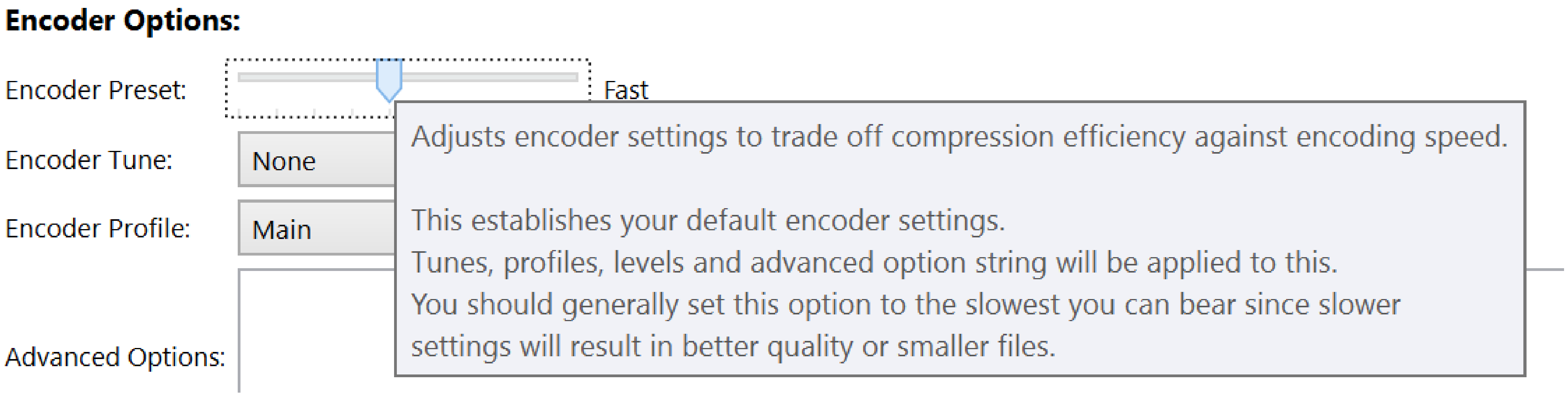
Bottom Line

- CBR
  - Only when essential
  - Live/tight connection bandwidths
- Capped CRF
  - Alluring technology - bandwidth savings are understated
  - But
    - Saves only 13% encoding time
- 2-Pass VBR
  - Slight increase in encoding cost and bandwidth
  - Best overall and low-frame quality
  - 200% seems the best option
  - How I tested all future encodes

Best Practice 2: Bitrate Control

- Nobody ever got fired for using 200% 2-Pass VBR
- Two-pass x264 is very fast (so not 2x one-pass)
- CBR – low frame issues, no bitrate benefit
- Capped CRF
  - Saves some encoding time
  - Can shave significant bitrate
  - Low-frame issues – legit concern, even with fair comparison

Best Practice 3: Match Preset to View Count



- Preset functions and differences
  - AWS MediaConvert - Elemental codec
  - HandBrake - x264 codec (ultrafast > placebo)
- Fundamental tradeoff
- Preset selection math

Exploring Presets

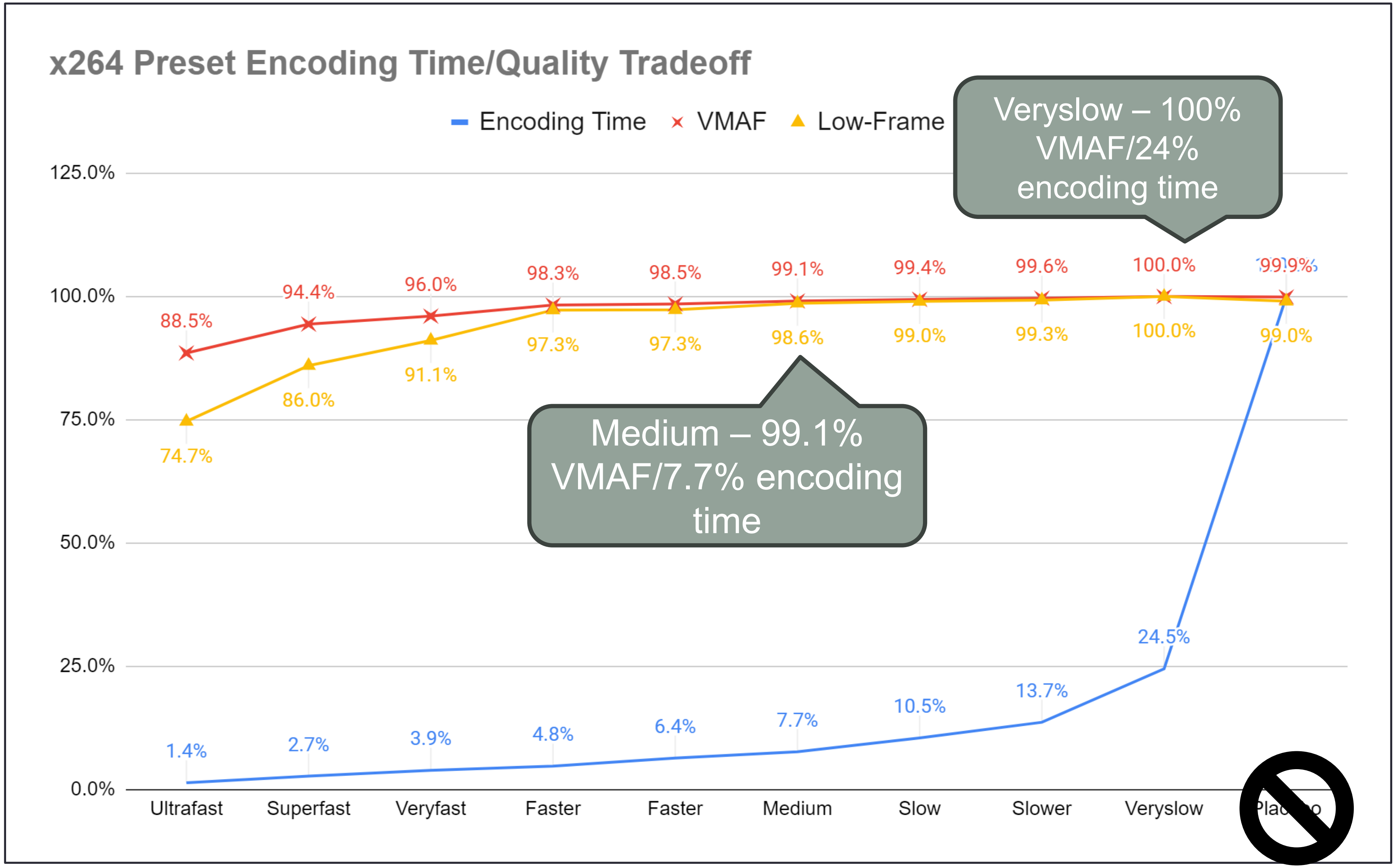
- What does the preset do?
  - Adjusts parameters to producers can choose desired quality/encoding time tradeoff
    - x264 - 10 presets - ultrafast to placebo
- Big Question: Does the preset control distribution quality?
    - Yes?
    - No?

Preset Role

- Controls ***encoding time/cost***, not ***quality***
- 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*** and impacts ***bandwidth cost***
  - Choosing a preset is ***always*** a tradeoff between encoding cost and bandwidth cost

Presets: Quality vs. Encoding Time Tradeoff

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

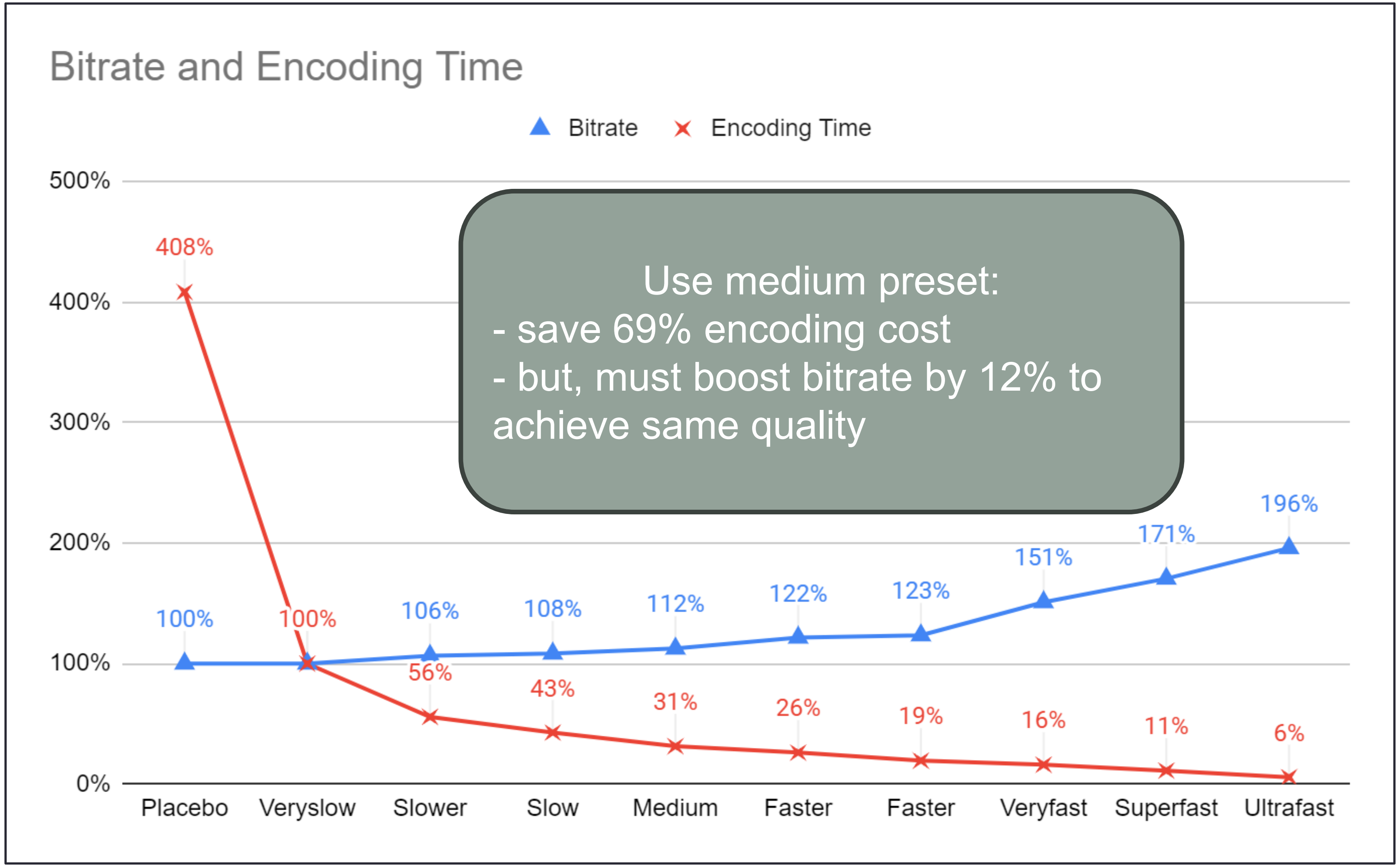


Next Question

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

So, if your target is 95, and you use the medium preset, what's the required bitrate boost

H.264 Preset



Would never use placebo, so adjust comparisons to veryslow

Preset	Bitrate	Encoding time
Ultrafast	196%	6%
Superfast	171%	11%
Veryfast	151%	16%
faster	123%	19%
fast	122%	26%
Medium	112%	31%
Slow	108%	43%
Slower	106%	56%
Veryslow	100%	100%
Placebo	100%	408%

x264 - 1080p30 file Viewer Count Breakeven - $0.08/GB

Encoding cost = $.12
2.22 GB/hr @ .08 = $.18/hour
250 @ $0.18 = $45
Total = $45.12

Encoding cost = $.35
1.9168 GB/hr @ .08 = $.1533/hour
5000 @ $0.1536 = $766.72
Total = $767

		Bitrate		4000						
		GB/hour		1.8		Cost per GB		0.08		
		Encode/hr		0.62						
Preset	Encode	Bandwidth GB	Bandwidth Cost		50	100	250	500	1000	5000
Ultrafast	$0.04	3.53	$0.28		$14.14	$28.24	$71	$141	$282	$1,410
Superfast	$0.07	3.07	$0.25		$12.35	$24.62	$61	$123	$246	$1,228
Veryfast	$0.10	2.72	$0.22		$10.97	$21.84	$54	$109	$217	$1,087
faster	$0.12	2.22	$0.18		$9.01	$17.90	$45	$89	$178	$889
fast	$0.16	2.19	$0.18		$8.92	$17.67	$44	$88	$175	$875
Medium	$0.19	2.02	$0.16		$8.29	$16.39	$41	$81	$162	$810
Slow	$0.27	1.95	$0.16		$8.06	$15.86	$39	$78	$156	$780
Slower	$0.35	1.92	$0.15		$8.01	$15.68	$39	$77	$154	$767
Veryslow	$0.62	1.80	$0.14		$7.82	$15.02	$37	$73	$145	$721
Placebo	$2.53	1.80	$0.14		$10	$17	$39	$75	$147	$723

x264 - 1080p30 file Viewer Count Breakeven - $0.08/GB

Encoding cost = $.12
2.22 GB/hr @ .08 = $.18/hour
250 @ $0.18 = $45
Total = $45.12

Encoding cost = $.35
1.9168 GB/hr @ .08 = $.1533/hour
5000 @ $0.1536 = $766.72
Total = $767

		Bitrate	4000							
		GB/hour	1.8			Cost per GB		0.08		
		Encode/hr	0.62							
Pre									1000	5000
Ultrafast									$282	$1,410
Superfa									$246	$1,228
Veryfast									$217	$1,087
faster									$178	$889
fast									$175	$875
Medium									$162	$810
Slow								$78	$156	$780
Slower	$0.35	1.92	$0.15		$8.01	$15.68	$39	$77	$154	$767
Veryslow	$0.62	1.80	$0.14		$7.82	$15.02	$37	$73	$145	$721
Placebo	$2.53	1.80	$0.14		$10	$17	$39	$75	$147	$723

What's the Point?

Encoding is a fraction of the overall cost of distribution.

Even at modest distributions, it makes sense to encode at the highest possible quality

x264 - Viewer Count Breakeven - $0.04/GB

		Bitrate		4000						
		GB/hour		1.8		Cost per GB		0.04		
		Encode/hr		0.62						
Preset	Encode	Bandwidth			50	100	250	500	1000	5000
Ultrafast	$0.04	3.53	$0.14		$7	$14	$35	$71	$141	$705
Superfast	$0.07	3.07	$0.12		$6	$12	$31	$61	$123	$614
Veryfast	$0.10	2.72	$0.11		$6	$11	$27	$54	$109	$544
faster	$0.12	2.22	$0.09		$5	$9	$22	$45	$89	$445
fast	$0.16	2.19	$0.09		$5	$9	$22	$44	$88	$438
Medium	$0.19	2.02	$0.08		$4	$8	$20	$41	$81	$405
Slow	$0.27	1.95	$0.08		$4	$8	$20	$39	$78	$390
Slower	$0.35	1.92	$0.08		$4	$8	$20	$39	$77	$384
Veryslow	$0.62	1.80	$0.07		$4	$8	$19	$37	$73	$361
Placebo	$2.53	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							
		GB/hour		1.8		Cost per GB		0.02			
		Encode/hr		0.62							
Preset	Encode	Bandwidth			50	100	250	500	1000	5000
Ultrafast	$0.04	3.53	$0.07		$4	$7	$18	$35	$71	$353
Superfast	$0.07	3.07	$0.06		$3	$6	$15	$31	$61	$307
Veryfast	$0.10	2.72	$0.05		$3	$6	$14	$27	$54	$272
faster	$0.12	2.22	$0.04		$2	$5	$11	$22	$45	$222
fast	$0.16	2.19	$0.04		$2	$5	$11	$22	$44	$219
Medium	$0.19	2.02	$0.04		$2	$4	$10	$20	$41	$203
Slow	$0.27	1.95	$0.04		$2	$4	$10	$20	$39	$195
Slower	$0.35	1.92	$0.04		$2	$4	$10	$20	$39	$192
Veryslow	$0.62	1.80	$0.04		$2	$4	$10	$19	$37	$181
Placebo	$2.53	1.80	$0.04		$4	$6	$12	$21	$39	$183

Best Practice 3: Preset

Best practice: Balance encoding/delivery cost

- Low distribution volumes – minimize encoding cost; boost bandwidth to achieve target quality
- High distribution volumes (hundreds of hours) – maximize encoding efficient for the lowest possible bitrate

Best Practice 4: Optimize Thread Count for Quality

- What are threads
- Impact on quality
- Impact on throughput
- Recommended for production
- Recommended for testing

What Are Threads

CPU

Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz

Logical processors

3% (3%)	1% (1%)	0% (12%)	1% (1%)	6% (5%)	1% (1%)	4% (4%)	1% (1%)
2% (2%)	1% (1%)	3% (3%)	1% (1%)	4% (4%)	1% (1%)	5% (3%)	1% (1%)
6% (4%)	1% (1%)	2% (2%)	1% (1%)	2% (2%)	1% (1%)	2% (2%)	1% (1%)
1% (1%)	2% (2%)	1% (1%)	2% (2%)	2% (2%)	2% (2%)	2% (2%)	6% (5%)
8% (4%)	54% (7%)	8% (11%)	4% (11%)	8% (11%)	4% (3%)	23% (8%)	3% (3%)
23% (8%)	3% (3%)	4% (3%)	35% (9%)	13% (4%)	3% (3%)	9% (12%)	3% (1%)
19% (5%)	1% (1%)	18% (6%)	2% (1%)	1% (12%)	3% (3%)	0% (12%)	4% (2%)
5% (16%)	2% (2%)	6% (13%)	3% (1%)	5% (12%)	6% (5%)	7% (15%)	7% (4%)

- Cores - physical hardware components in CPU that execute instructions
- Threads - virtual components that divide tasks to be handled by the cores
  - This computer has 2 CPUs with 16 cores
  - Each core has two threads
  - 64 total threads
- FFmpeg – can assign threads to command line. Impacts
  - Transcoding speed
  - Overall throughput
  - To lesser degree, single file quality

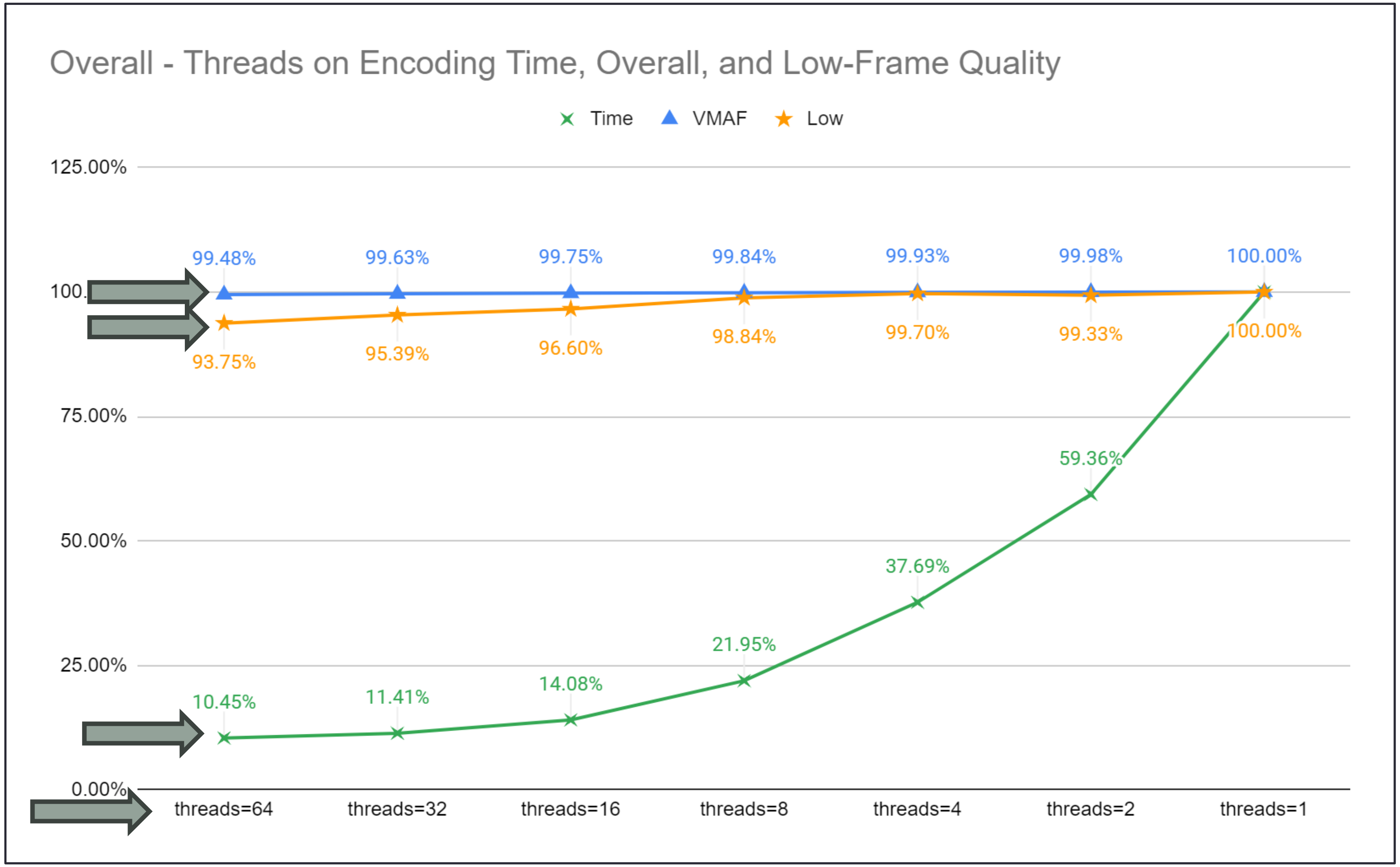
What's Default?

Stream size :	7.08 MiB (99%)
Writing library :	x264 core 164 r3191 4613ac3
Encoding settings :	cabac=0 / ref=1 / deblock=0:0:0 / analyse=0:0 / me=dia / subme=0 / psy=1 / psy_rd=1.00:0.00 / mixed_ref=0 / me_range=16 / chroma_me=1 / trellis=0 / 8x8dct=0 / cqm=0 / deadzone=21,11 / fast_pskip=1 / chroma_qp_offset=0 / threads=34 / lookahead_threads=5 / sliced_threads=0 / nr=0 / decimate=1 / interlaced=0 / bluray_compat=0 / constrained_intra=0 / bframes=0 / weightp=0 / keyint=250 / keyint_min=24 / scenecut=0 / intra_refresh=0 / rc=crf / mbtree=0 / crf=23.0 / qcomp=0.60 / qpmin=0 / qpmax=69 / qpstep=4 / ip_ratio=1.40 / aq=0
Language :	English

- Not sure - here's recent encode on 64-core workstation
  - Encoding only this file
- 34 threads - let's see impact on quality/throughput

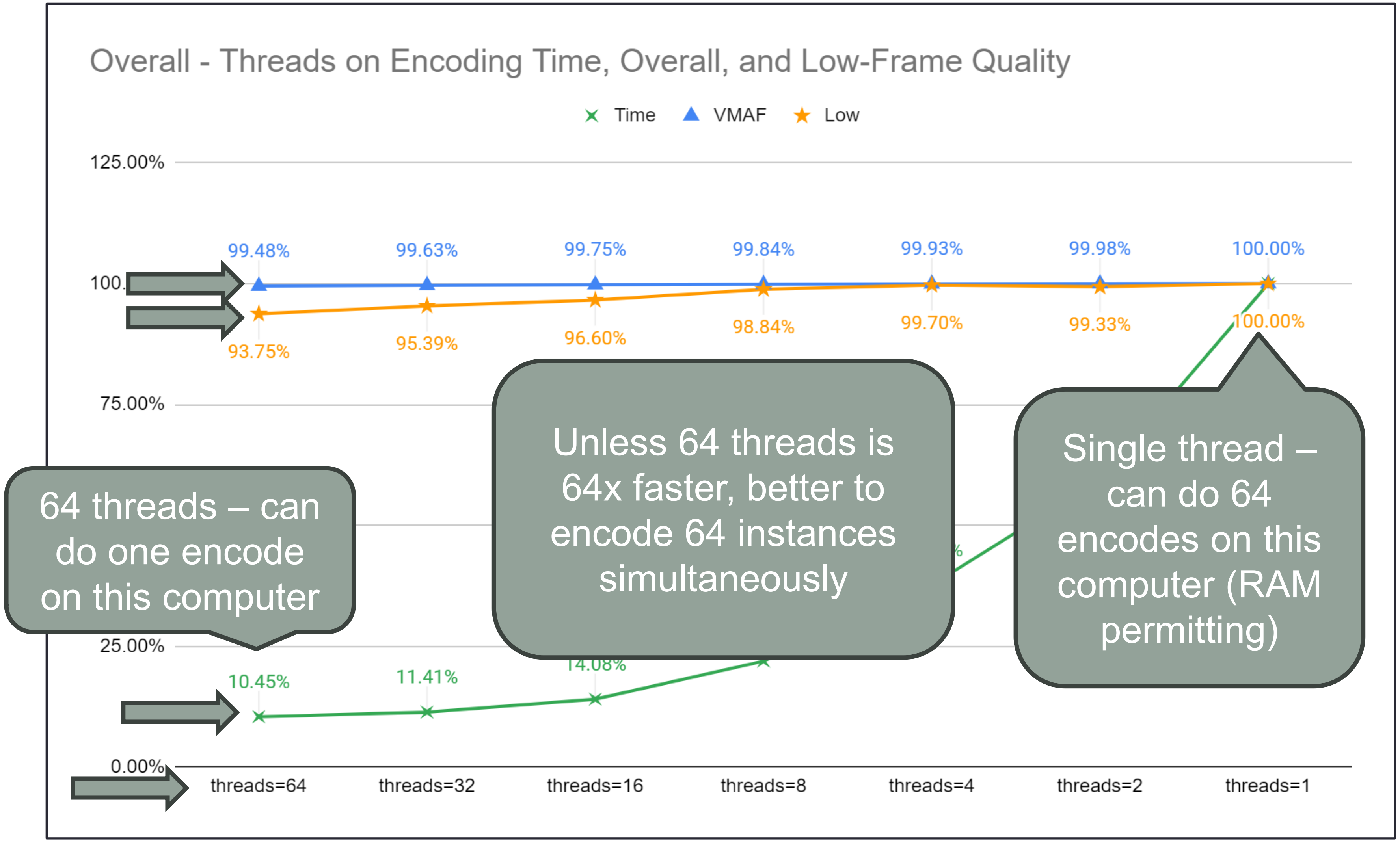
Impact on Quality - Overall

- Overall
  - Max .52 VMAF delta - Harmonic
  - Max 6.25 VMAF - low frame



Impact on Quality - Overall

- Overall
  - Max .52 VMAF delta - Harmonic
  - Max 6.25 VMAF - low frame



Soccer - 1 - 64

1-thread delivers best quality (no surprise)

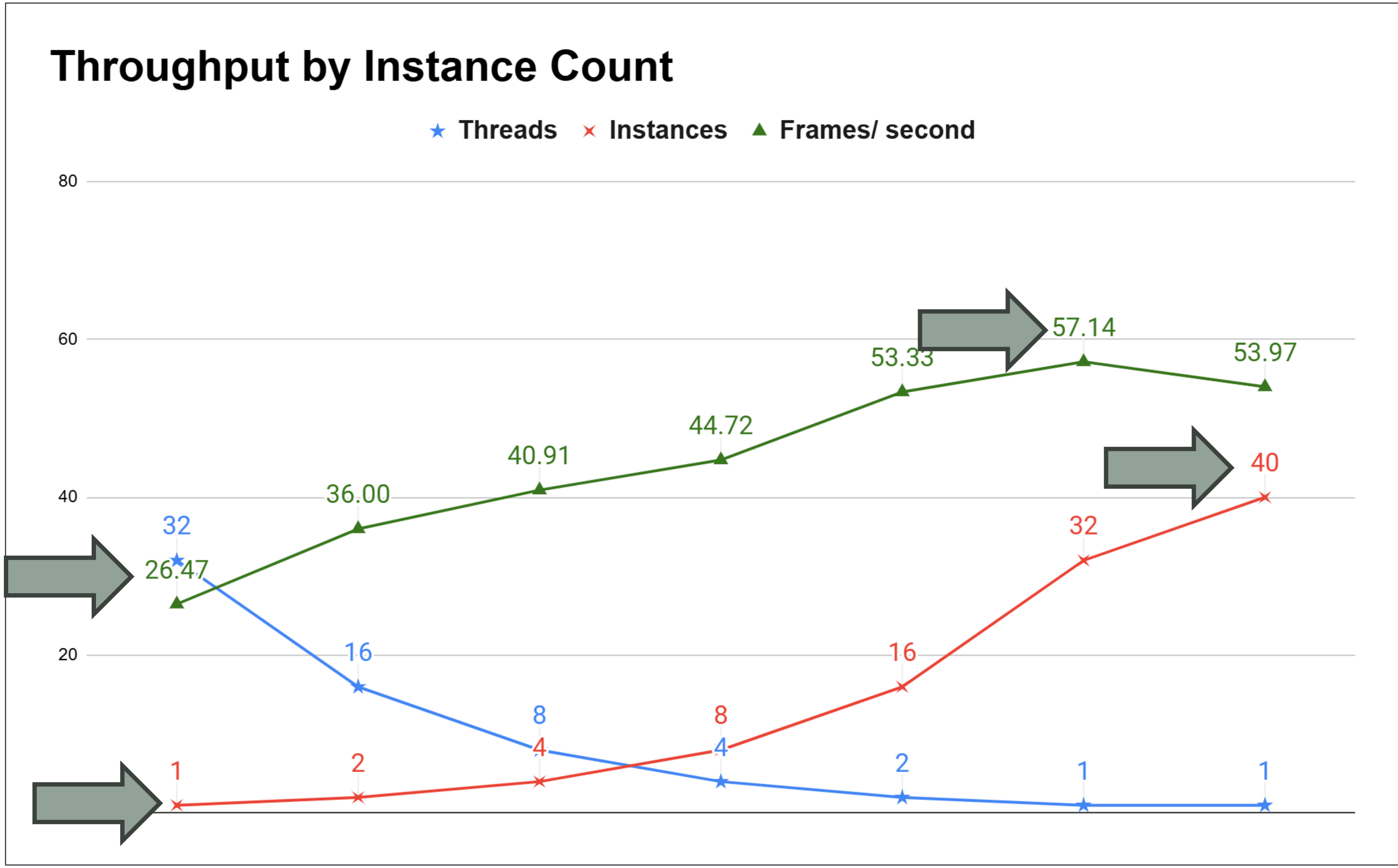
64-thread dramatically worse (big surprise)

From a Quality Perspective

- Limit threads when encoding on multicore machine
  - For production with x264, a single thread is always highest quality option
- What about performance?

Cost Per Stream

- As instances increase
- And threads decrease
- FPS increases
- Until you oversaturate threads (> 32)
  - Crashing
- Quality increases as well



Best Practice – Threads – H.264

- Low thread count with high instances seems to deliver
  - Best throughput
  - Best quality
- Awful configuration for testing (files encode so slowly)
  - I tested with eight threads

Best Practice 4: Thread Count

Best practice: Balance encoding/delivery cost

- Low distribution volumes – minimize encoding cost; boost bandwidth to achieve target quality
- High distribution volumes (hundreds of hours) – maximize encoding efficient for the lowest possible bitrate

Bonus Best Practice for AWS

- Choose the best CPU for H.264 processing

Three Contestants

https://www.johnvansickle.com/ffmpeg/

	Amazon	AMD	Intel
Instance	c7g.8xlarge	c7a.8xlarge	c7i.8xlarge
On Demand	$1.1562	$1.64224	$1.428

30% cheaper than AMD

13% cheaper than AMD

release: 7.0.1

ffmpeg-release-amd64-static.tar.xz - md5

ffmpeg-release-i686-static.tar.xz - md5

ffmpeg-release-arm64-static.tar.xz - md5

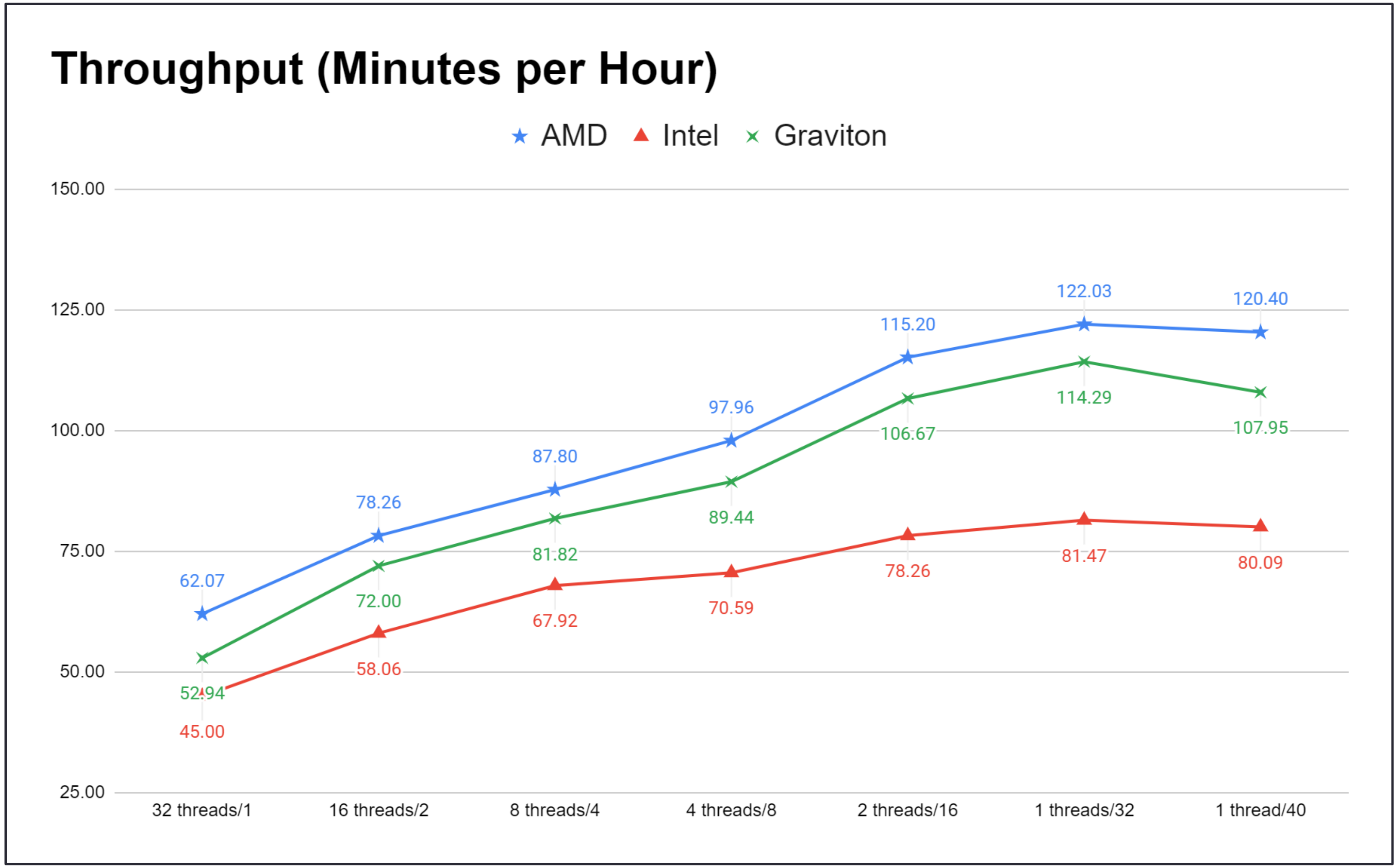
ffmpeg-release-armhf-static.tar.xz - md5

ffmpeg-release-armel-static.tar.xz - md5

- Three 32 - vCPU CPUs
  - Test from 1 instance/32-cores to 40 instances/1-core (1080p veryslow transcode)
  - Computer cost per-hour to encode
- Goals
  - ID best configuration (you've seen)
  - ID whether going beyond CPU count is advised (to 40)
  - ID fastest CPU
  - ID Least expensive CPU

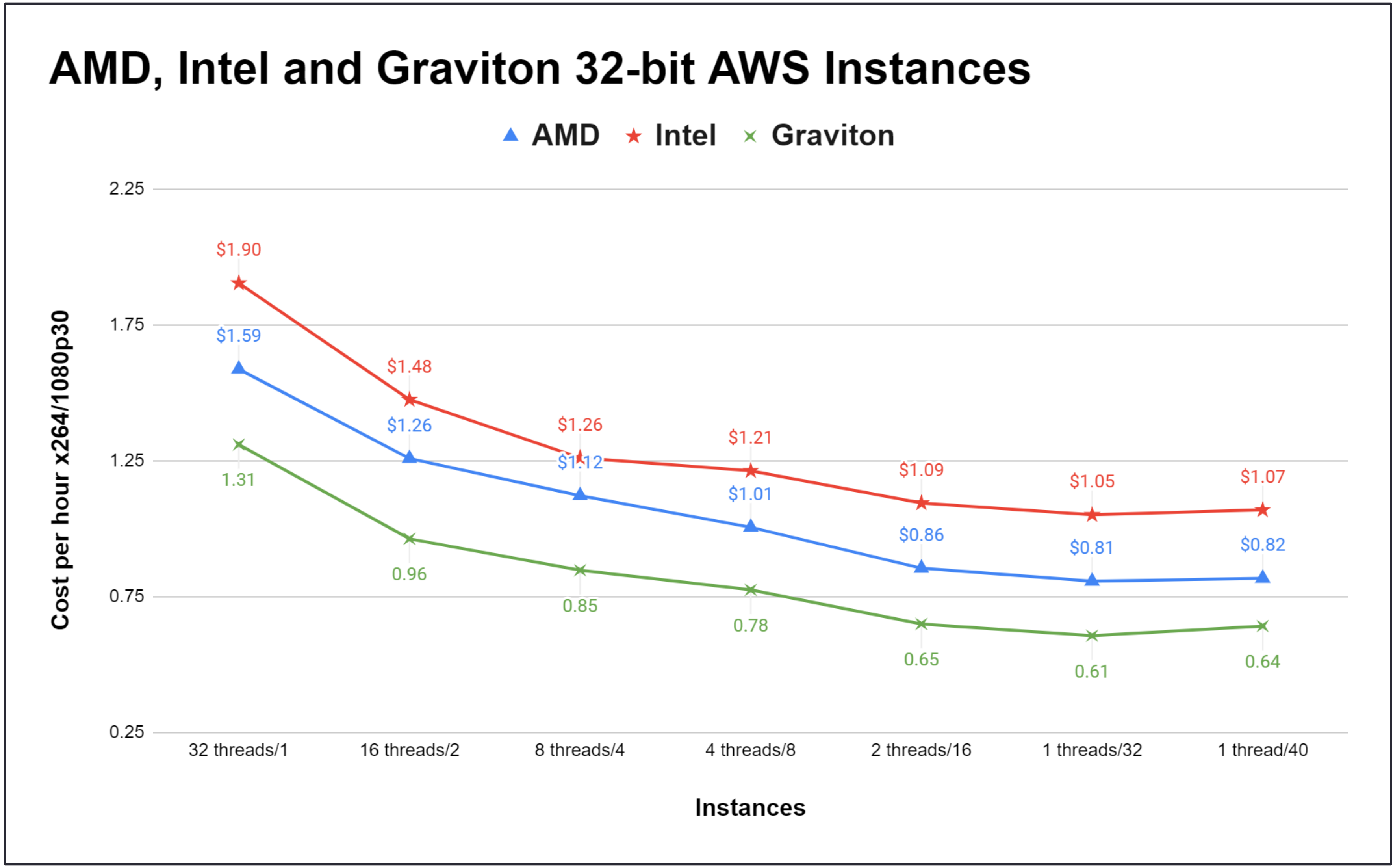
AMD Was the Fastest

- AMD delivered fastest throughput (minutes of video processed per hour)
- This increased with the number of instances
- If you're in a hurry, use AMD



Graviton was Lowest Cost Per Hour

- Graviton output less, but cost a lot less as well
- If you're on a budget, use Graviton
- And threads decrease
- FPS increases
- Until you oversaturate threads (> 32)
  - Crashing
- Quality increases as well



As Stated Previously

- Low threads/high instances delivers:
  - Best quality
  - Best throughput
- Don’t go beyond cores on workstation
  - Throughput drops - all
  - Intel - crashed

HEVC Agenda

- Choosing the optimal GOP size
  - Benefits of a variable GOP
- Bitrate control
- Choosing a preset
- Choosing the optimal thread count
- Working with Wavefront Parallel Processing

Best Practice 1 – HEVC – Best GOP Size

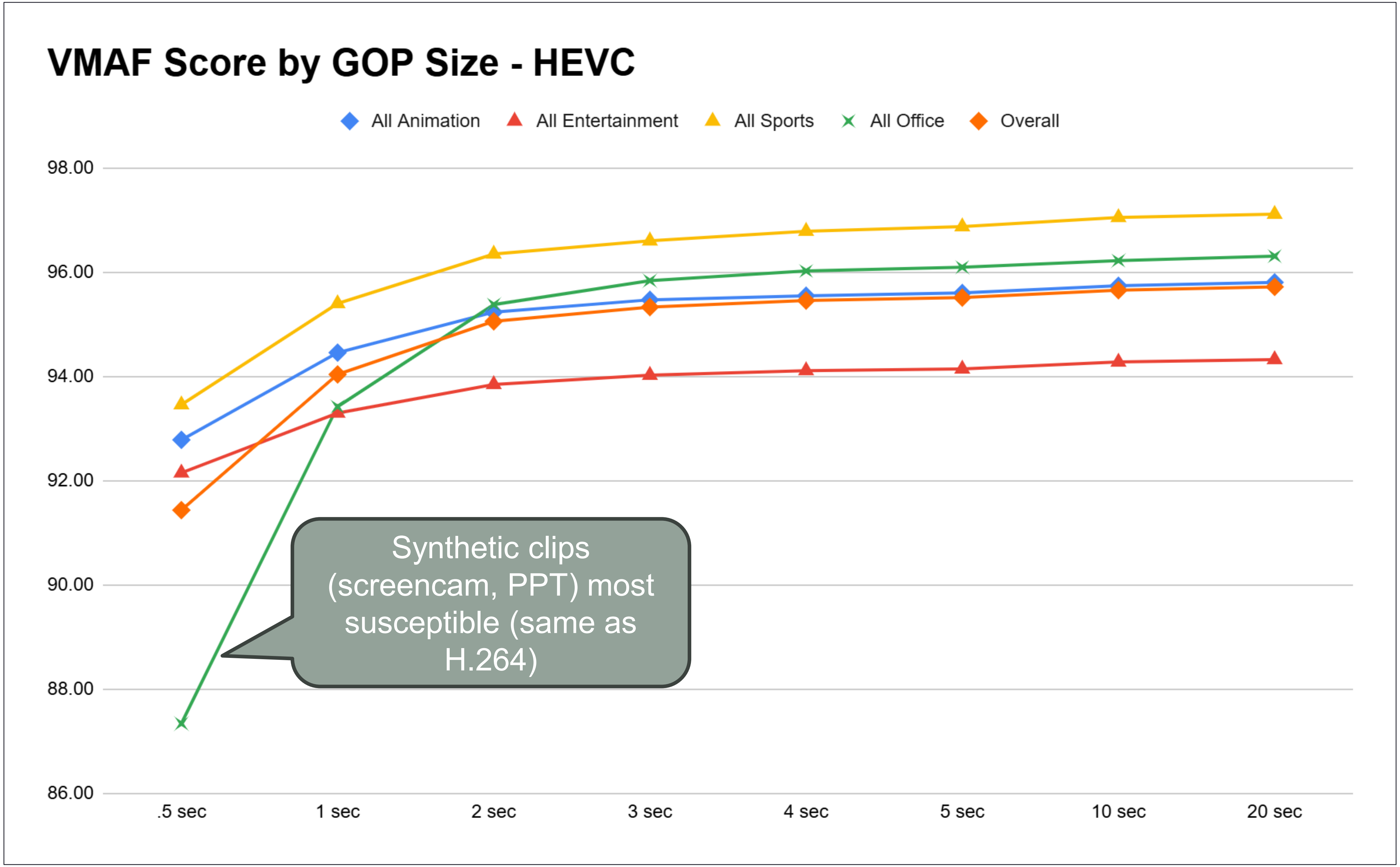
- What: GOP size (I-frame interval) is a key config option in all encodes
- Historical
  - Very small (like .5 second) for MPEG-2
  - Very long (10-20 seconds) for downloaded video
  - Typically, 2-5 seconds for adaptive bitrate video
    - Must divide evenly into segment size
- Question
  - How does GOP size impact quality
- Test – 13 files in 4 categories
  - Entertainment
  - Sports
  - Animation
  - Office

Best Practice 1: Longer is Better

Overall - HEVC	.5 sec	1 sec	2 sec	3 sec	4 sec	5 sec	10 sec	20 sec
All Animation	92.79	94.46	95.24	95.48	95.56	95.61	95.75	95.81
All Entertainment	92.15	93.30	93.85	94.03	94.12	94.15	94.28	94.33
All Sports	93.46	95.41	96.36	96.61	96.80	96.88	97.06	97.12
All Office	87.34	93.43	95.39	95.85	96.03	96.10	96.23	96.32
Overall	**91.44**	**94.04**	**95.06**	**95.34**	**95.46**	**95.52**	**95.66**	**95.73**
Delta from Max	**4.29**	**1.68**	**0.66**	**0.39**	**0.26**	**0.21**	**0.06**	**0.00**

Diminishing returns

- Benefit significant at lower range
  - About 2/3 of H.264
- Then diminishing returns
- Key limit: must divide evenly into segment size
  - 10 second copy – 1/2/5/10
  - Why not try 10? Check for playability


VMAF Score by GOP Size - HEVC
All Animation
All Entertainment
All Sports
All Office
Overall
98.00
96.00
94.00
92.00
90.00
88.00
86.00
.5 sec
1 sec
2 sec
3 sec
4 sec
5 sec
10 sec
20 sec
Synthetic clips (screencam, PPT) most susceptible (same as H.264)

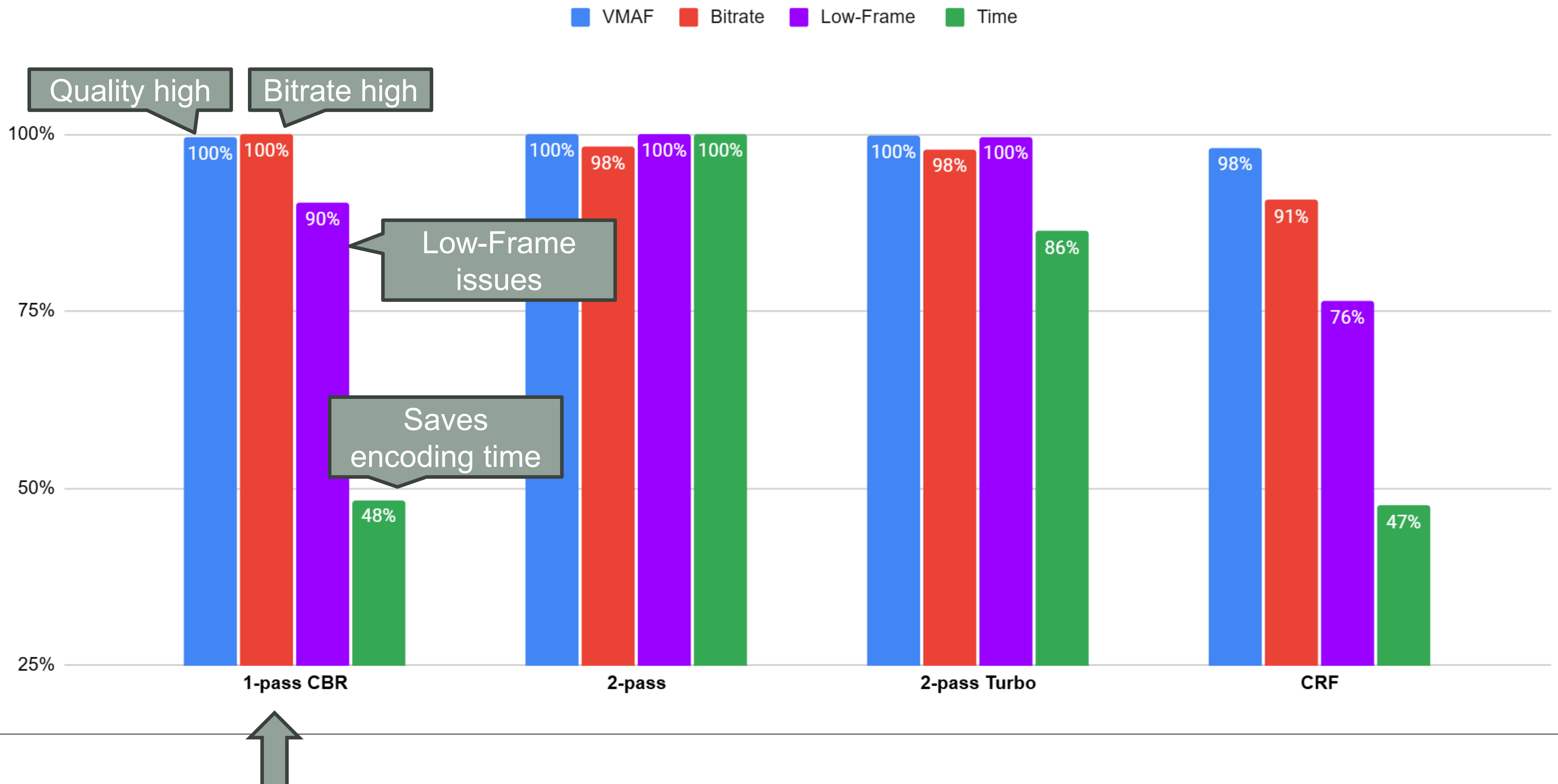
Best Practice 1: GOP Size

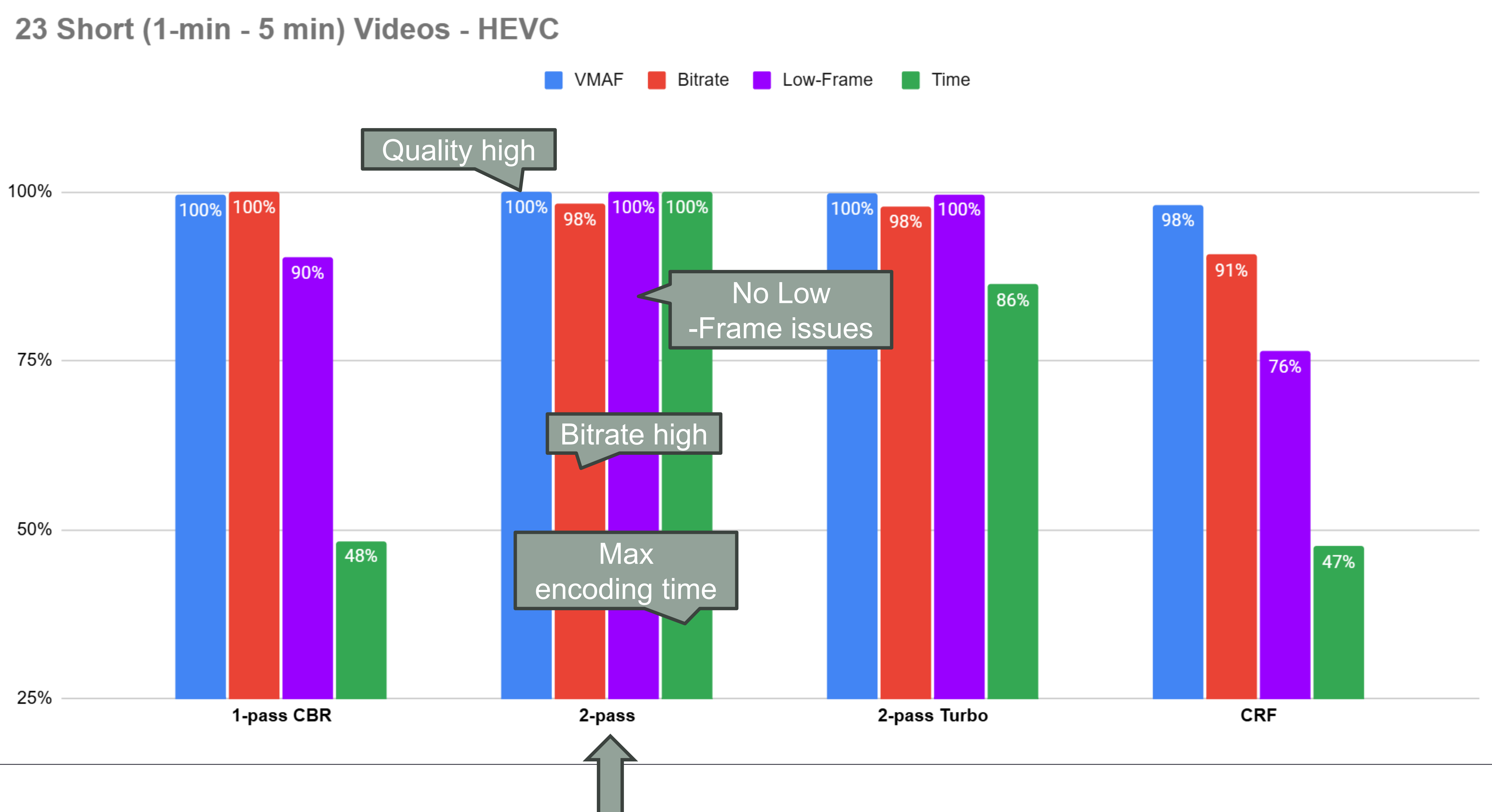
Longer is better

Best Practice 2: Bitrate Control

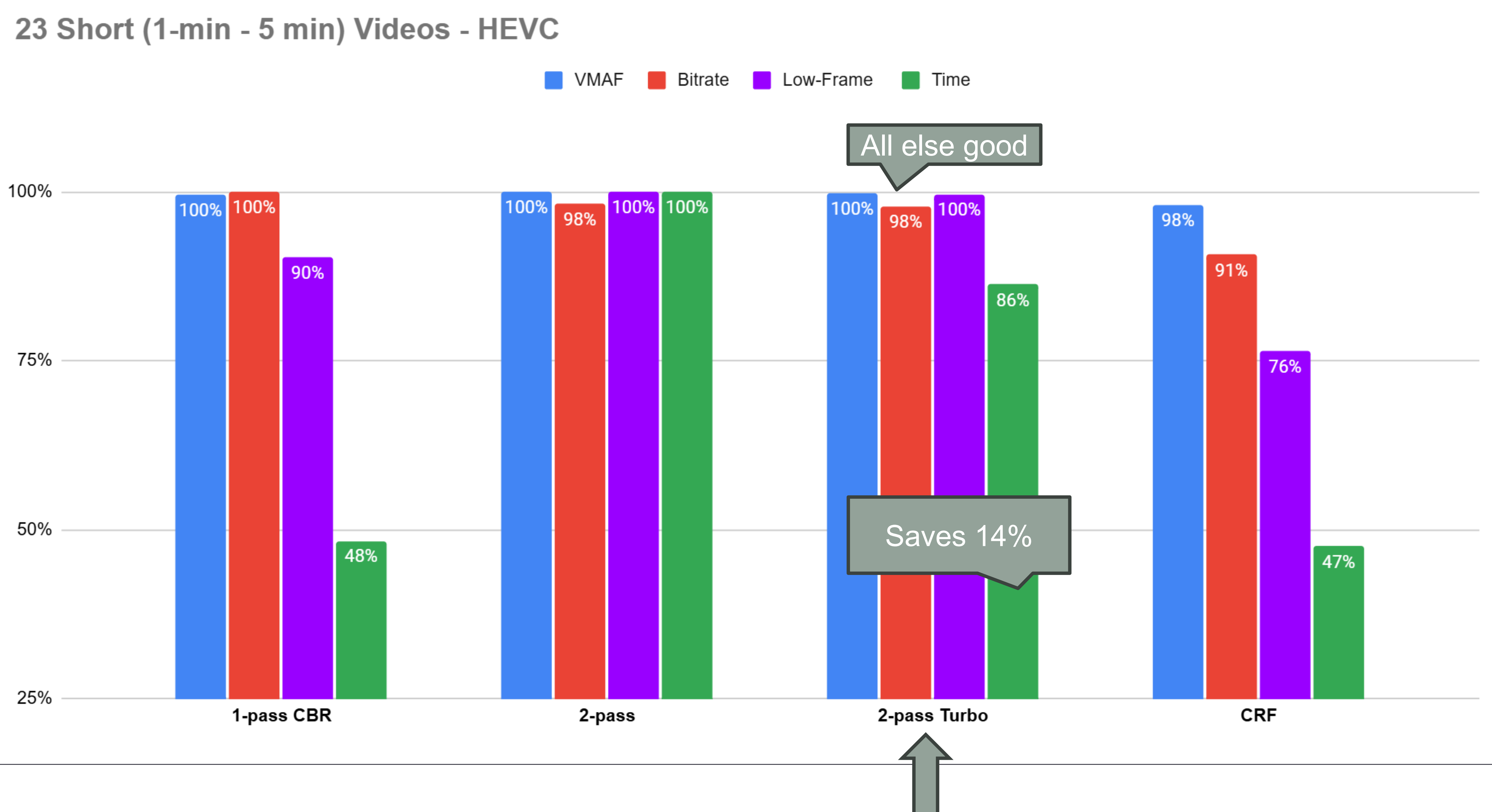
- Tested configurations
  - 1-Pass CBR
  - 2-Pass (200% constrained VBR)
  - 2-Pass turbo (200% constrained VBR)
  - Capped CRF (constant rate factor)

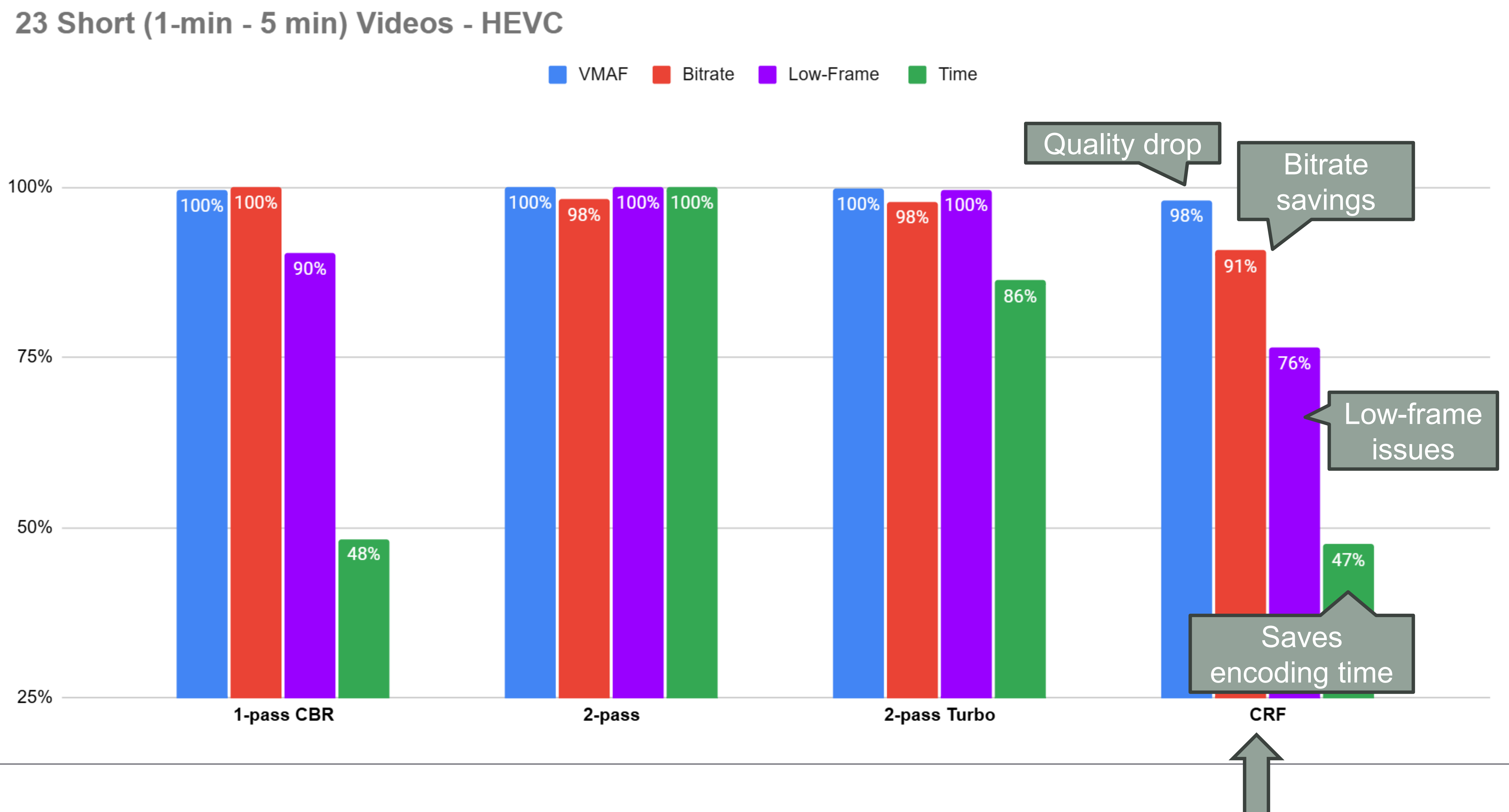
23 Short (1-min - 5 min) Videos - HEVC



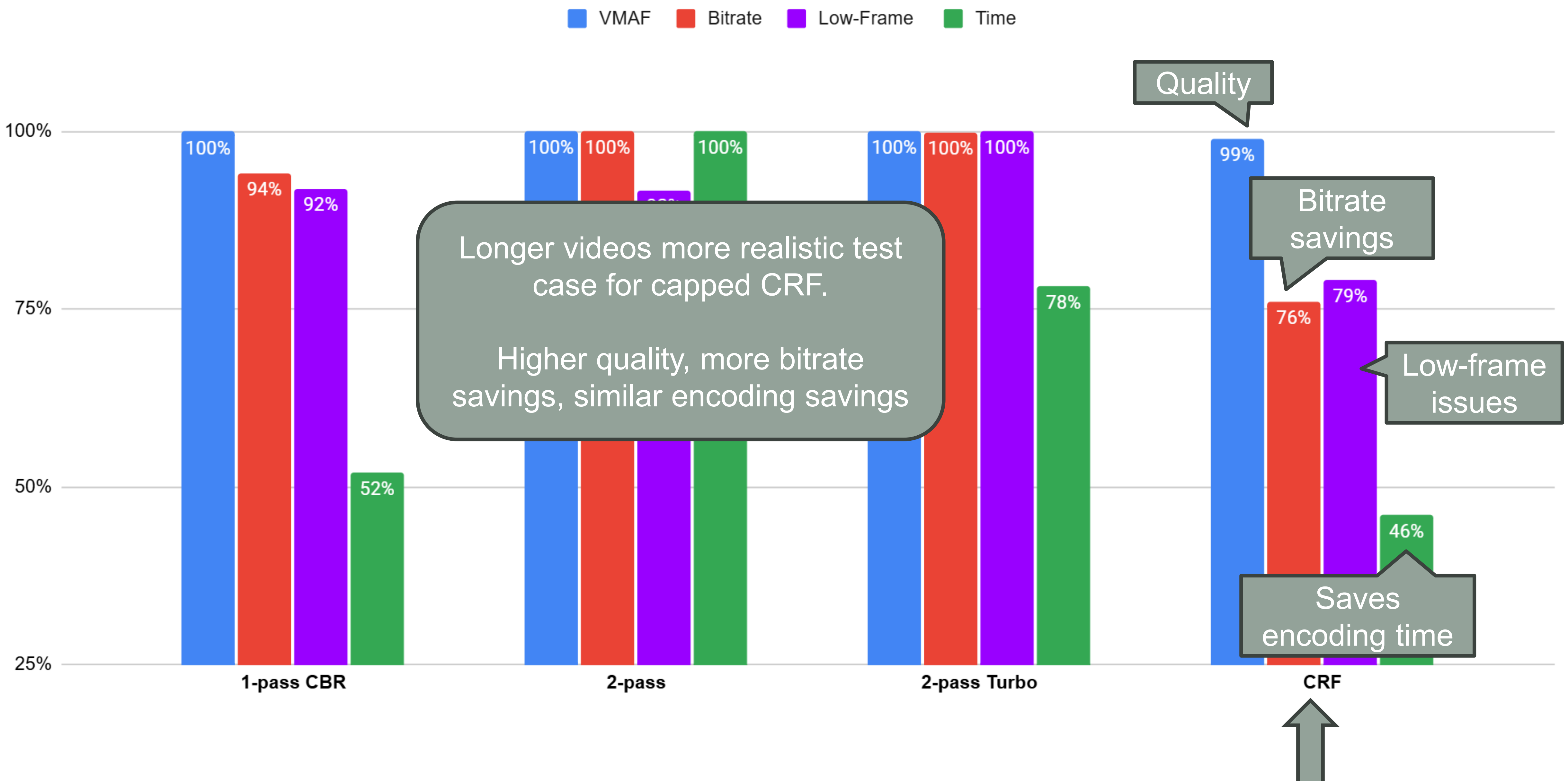


23 Short (1-min - 5 min) Videos - HEVC
VMAF
Bitrate
Low-Frame
Time
Quality high
No Low
-Frame issues
Bitrate high
Max
encoding time
100%
75%
50%
25%
100%
100%
90%
48%
100%
98%
100%
100%
100%
98%
100%
86%
98%
91%
76%
47%
1-pass CBR
2-pass
2-pass Turbo
CRF


23 Short (1-min - 5 min) Videos - HEVC
VMAF
Bitrate
Low-Frame
Time
All else good
Saves 14%
100%
75%
50%
25%
100%
100%
90%
48%
100%
98%
100%
100%
100%
98%
100%
86%
98%
91%
76%
47%
1-pass CBR
2-pass
2-pass Turbo
CRF


23 Short (1-min - 5 min) Videos - HEVC
VMAF
Bitrate
Low-Frame
Time
100%
75%
50%
25%
100%
100%
90%
48%
100%
98%
100%
100%
100%
98%
100%
86%
98%
91%
76%
47%
1-pass CBR
2-pass
2-pass Turbo
CRF
Quality drop
Bitrate savings
Low-frame issues
Saves encoding time

Two 12 Minute Videos



Bottom Line

- CBR
  - Only when essential
  - Live/tight connection bandwidths
- 2-Pass VBR
  - Most expensive
  - Best overall and low-frame quality
- 2-Pass Turbo
  - 14% cost/time savings
  - No negatives
- Capped CRF
  - Alluring technology - bandwidth savings can be significant (DIY content adaptive technique)
    - Overall quality good
    - Low-frame a concern
    - Saves 39% encoding time

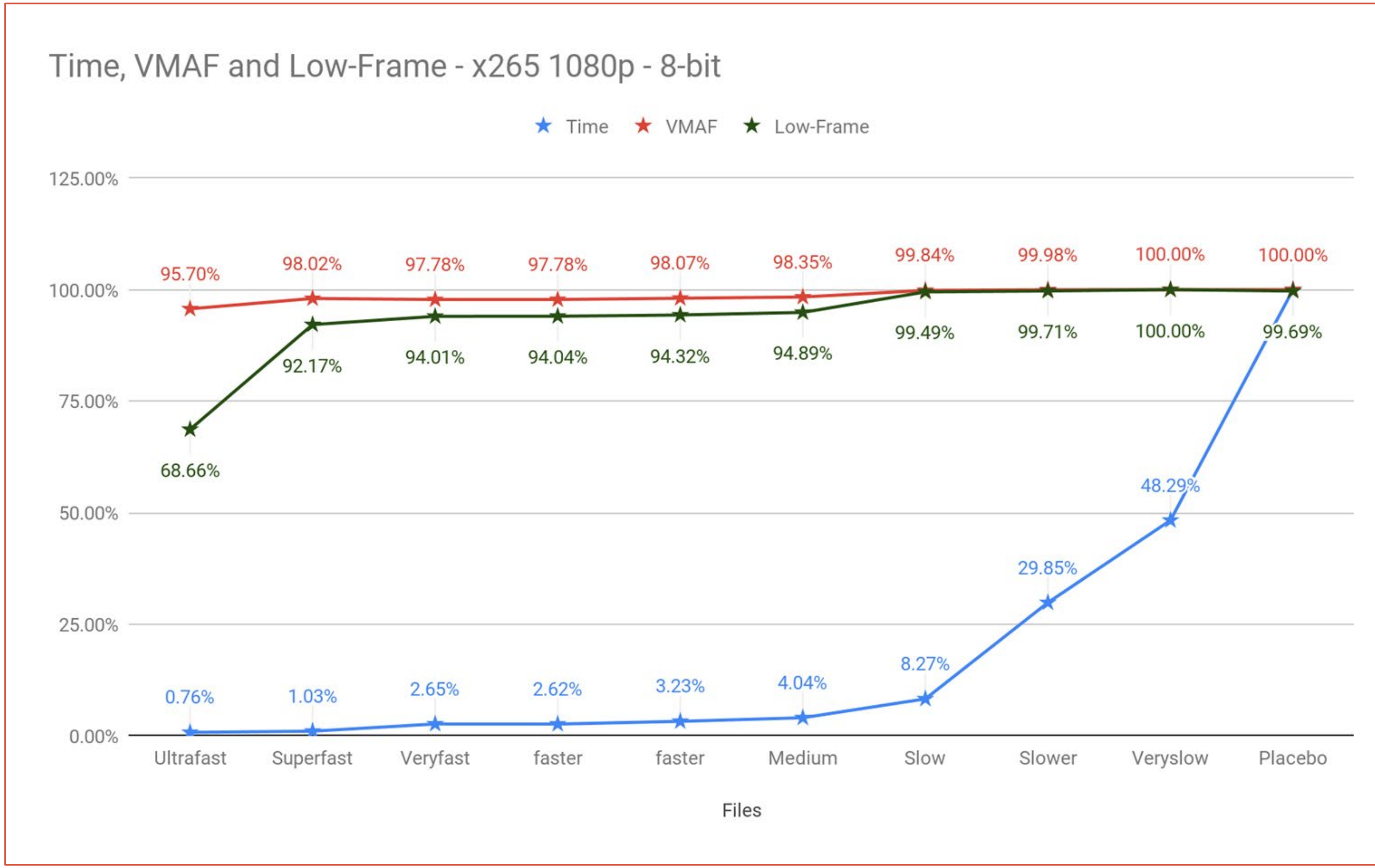
Best Practice 2: Bitrate Control

- Unlike H.264, 2-pass involves with substantial performance penalty

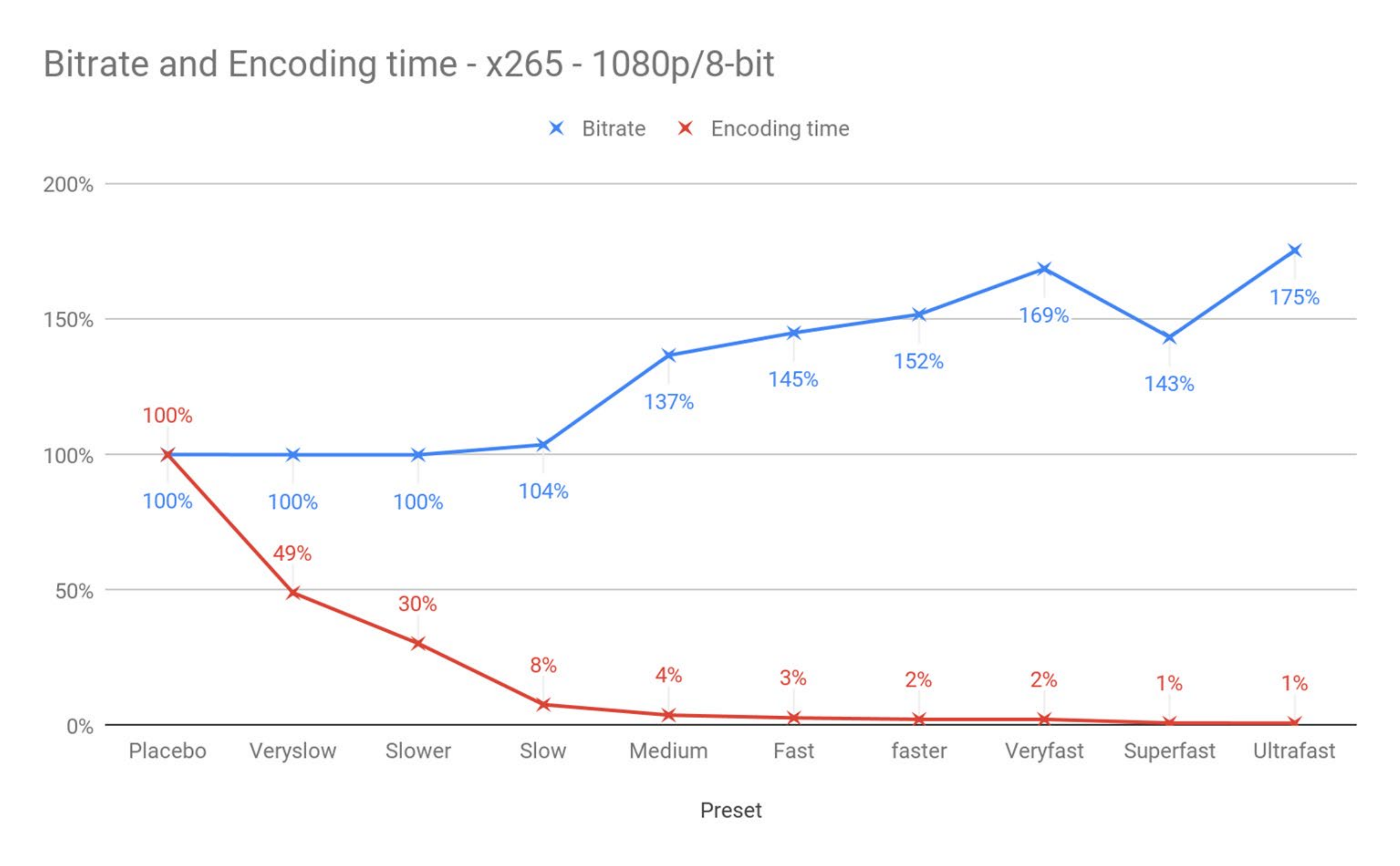
Best Practice 3 – Optimal Preset

- Controls ***encoding time/cost***, not ***quality***
- 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*** and impacts ***bandwidth 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?



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

Input parameters

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)

Default

		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

Best Practice - Presets

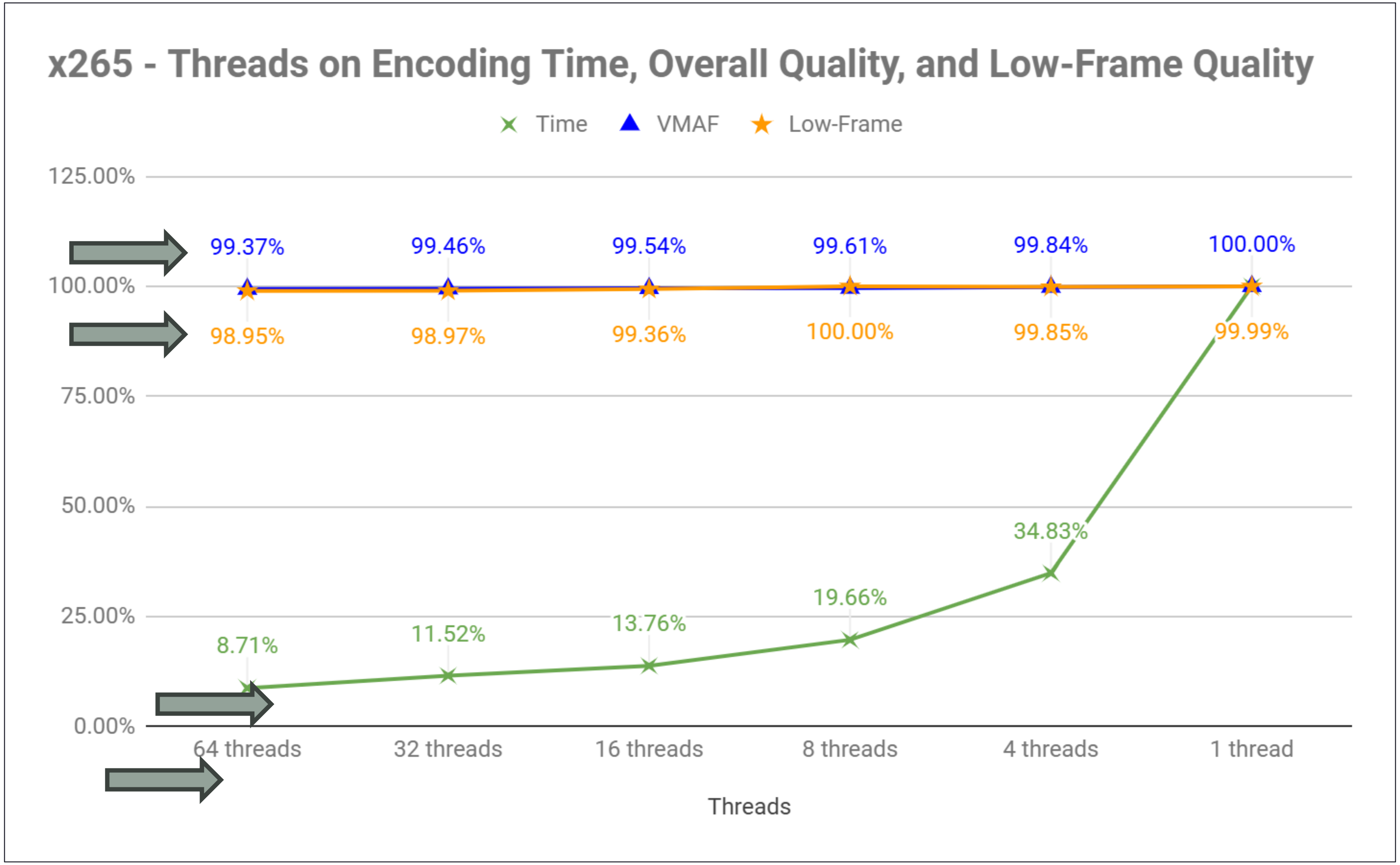
- Run tests on your own files (results will vary by content, resolution, etc)
- Perform your own calculations
- If your typical video is viewed over 10,000 times (or so), it almost always pay to use the veryslow preset
  - Placebo almost never delivers the best quality and almost always takes much, much longer to encode

Best Practice 4: Choose the Optimal Thread Count

- What are threads
- Impact on quality
- Impact on throughput
- Recommended for production
- Recommended for testing

Impact on Quality - Overall

- Overall
  - Max .59 VMAF delta - Harmonic
  - Max .99 VMAF - low frame



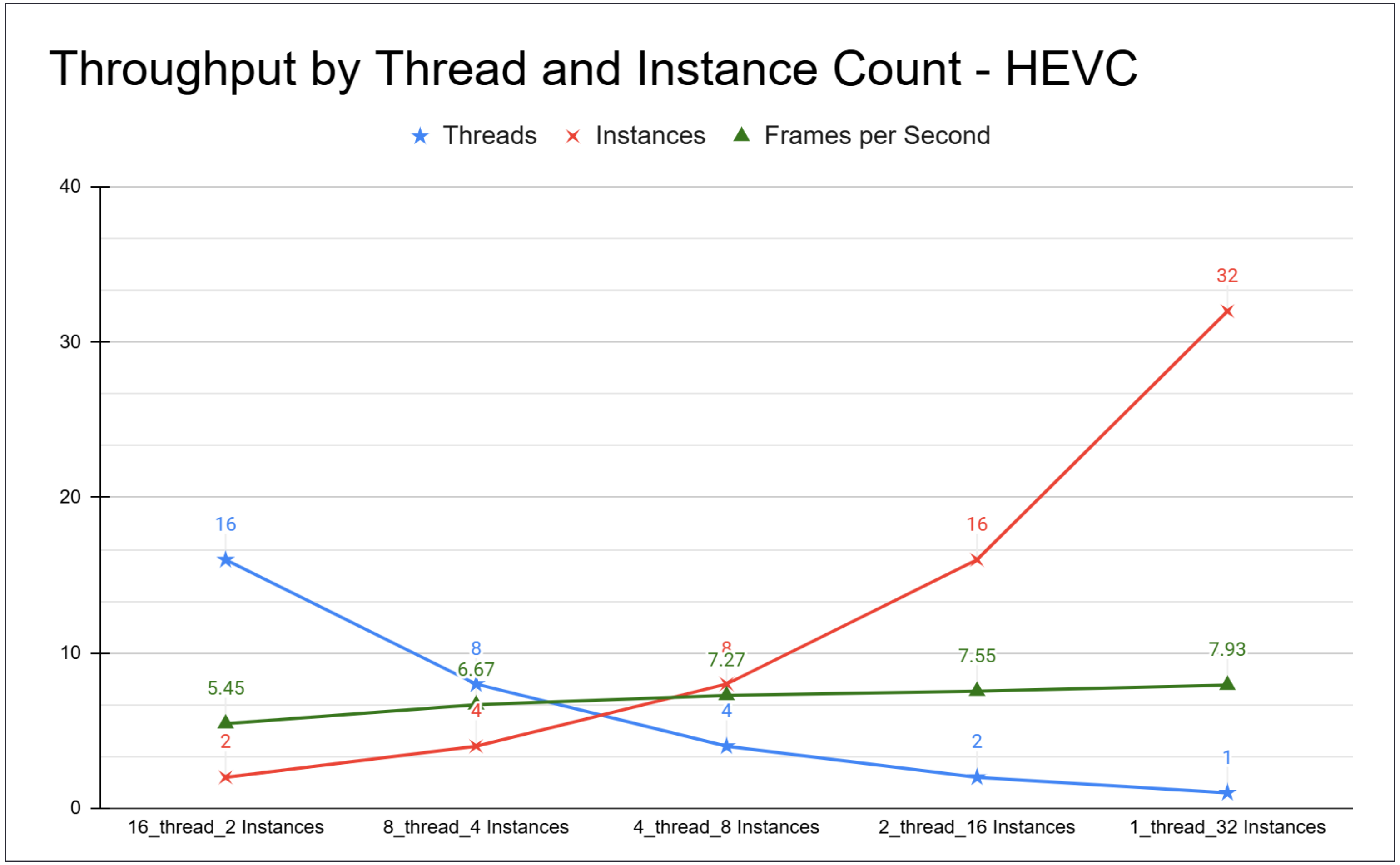
From a Quality Perspective

- Limit threads when encoding on multicore machine
- For production with x265, a single thread is always highest quality option

- What about performance?

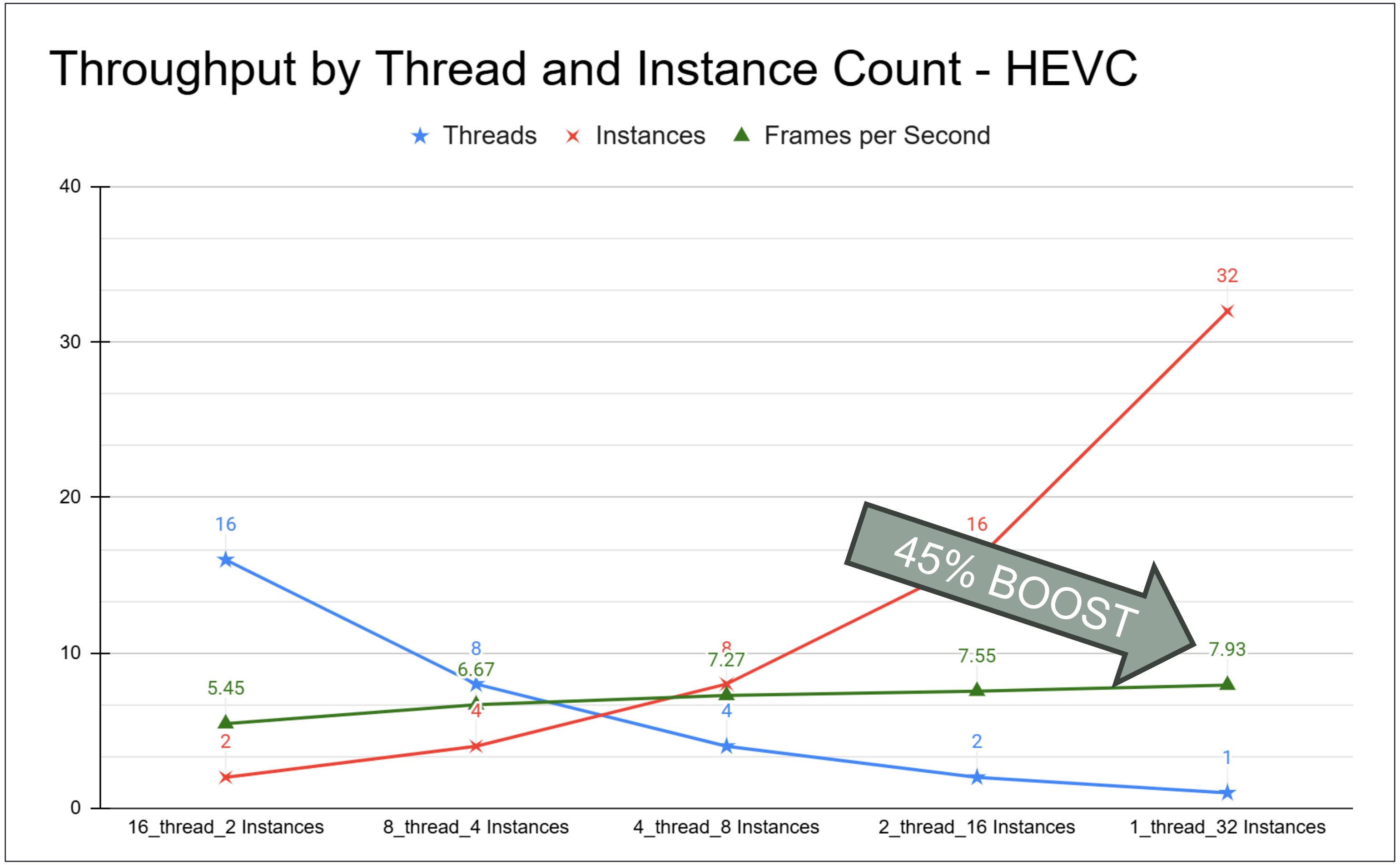
Cost Per Stream

- As instances increase
- And threads decrease
- FPS increases
- Until you oversaturate threads (> 32)
  - Crashing
- Quality increases as well



Cost Per Stream

- As instances increase
- And threads decrease
- FPS increases
  - Looks small – but 45%
- Quality increases as well



Best Practice – Threads

- Low thread count with high instances seems to deliver
  - Best throughput
  - Best quality
- Awful configuration for testing (files encode so slowly)
- I tested with eight threads

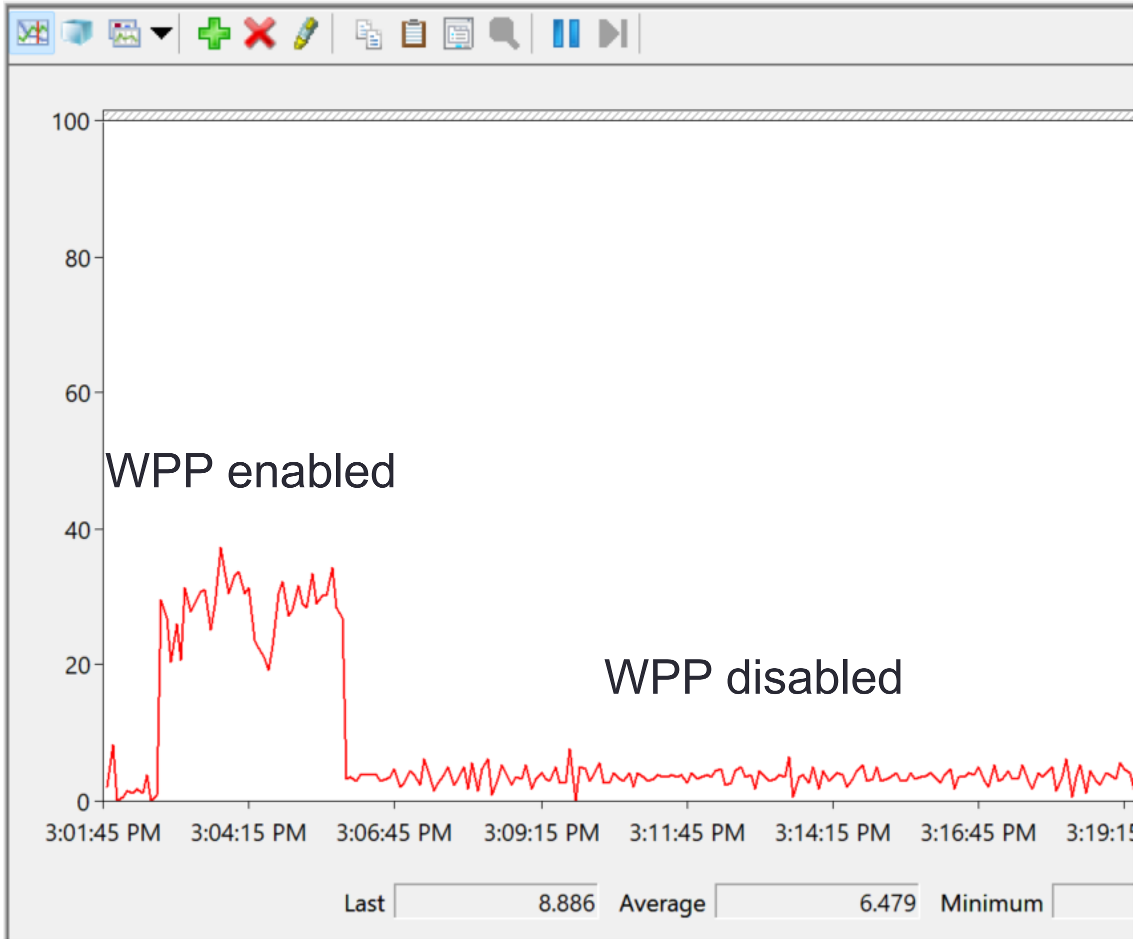
Best Practice 5 - Wavefront Parallel Processing (WPP)

	Encoding Time	VMAF	Low Frame
With WPP	03:15	90.23	77.50
No WPP	23:51	90.42	76.73
Delta	7.3x	-0.19	-0.77

- Enables parallel processing
  - Huge boost in encoding efficiency
  - Very slight drop in quality
- Question
  - Where is this additional performance coming from?

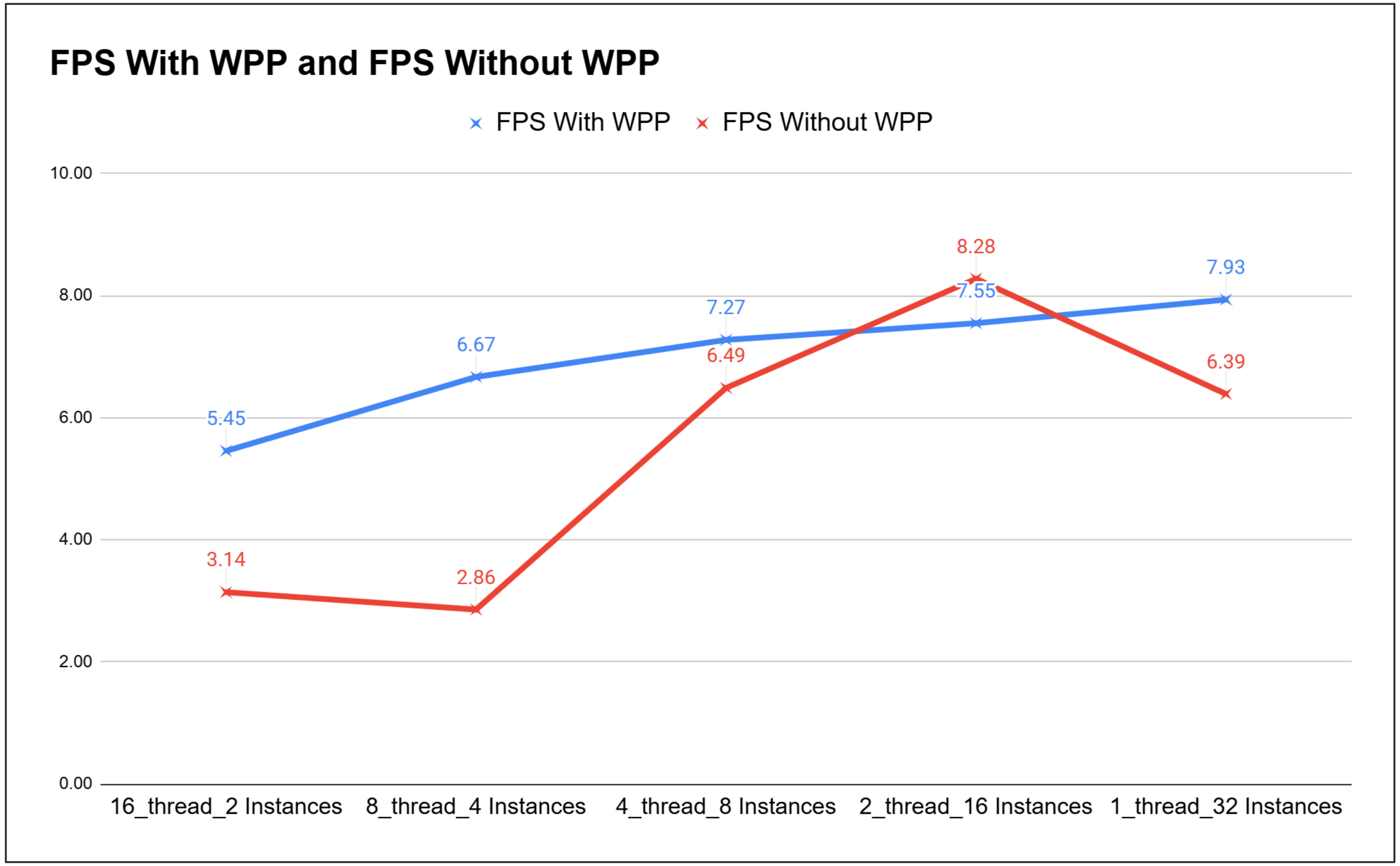
Wavefront Parallel Processing (WPP).

- WPP uses more cores; that’s why it’s faster (32-core workstation)
- Compare with and without WPP on the same system

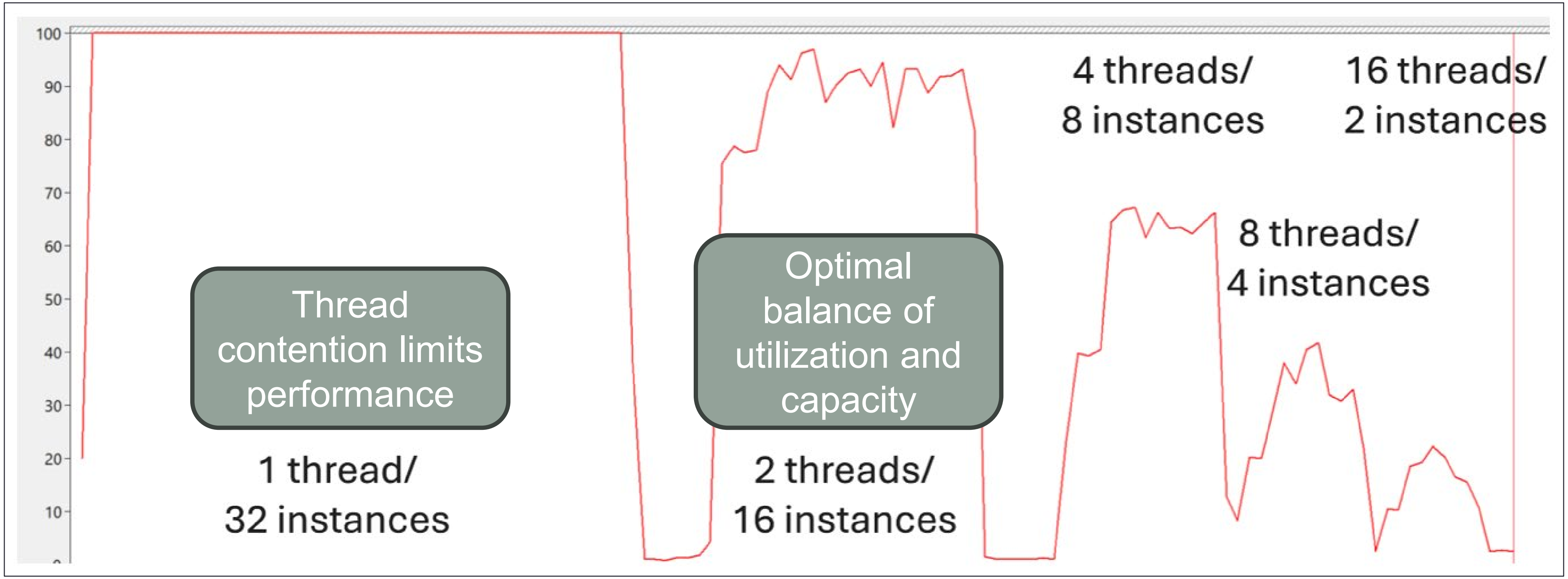


Throughput With and Without WPP

- Best without WPP
  - Very slightly better quality
  - Very slightly better performance
- Simpler jobs win when the system's pushed to the edge
  - Definitely system specific
- Bottom line: Don't assume that the faster single-file solution is the best for multiple files
  - Run your own tests



CPU Utilization – Different Configurations



Wasted capacity

Best Practice all: Scaling with Lanczos for Lower Rungs

MAXIMIZING QUALITY AND THROUGHPUT IN FFMPEG SCALING

Jan Ozer   February 11, 2023   FFmpeg   2 Comments   2,097 Views

The thing about FFmpeg is that there are almost always multiple ways to accomplish the same basic function. In this post, we look at four approaches to scaling.

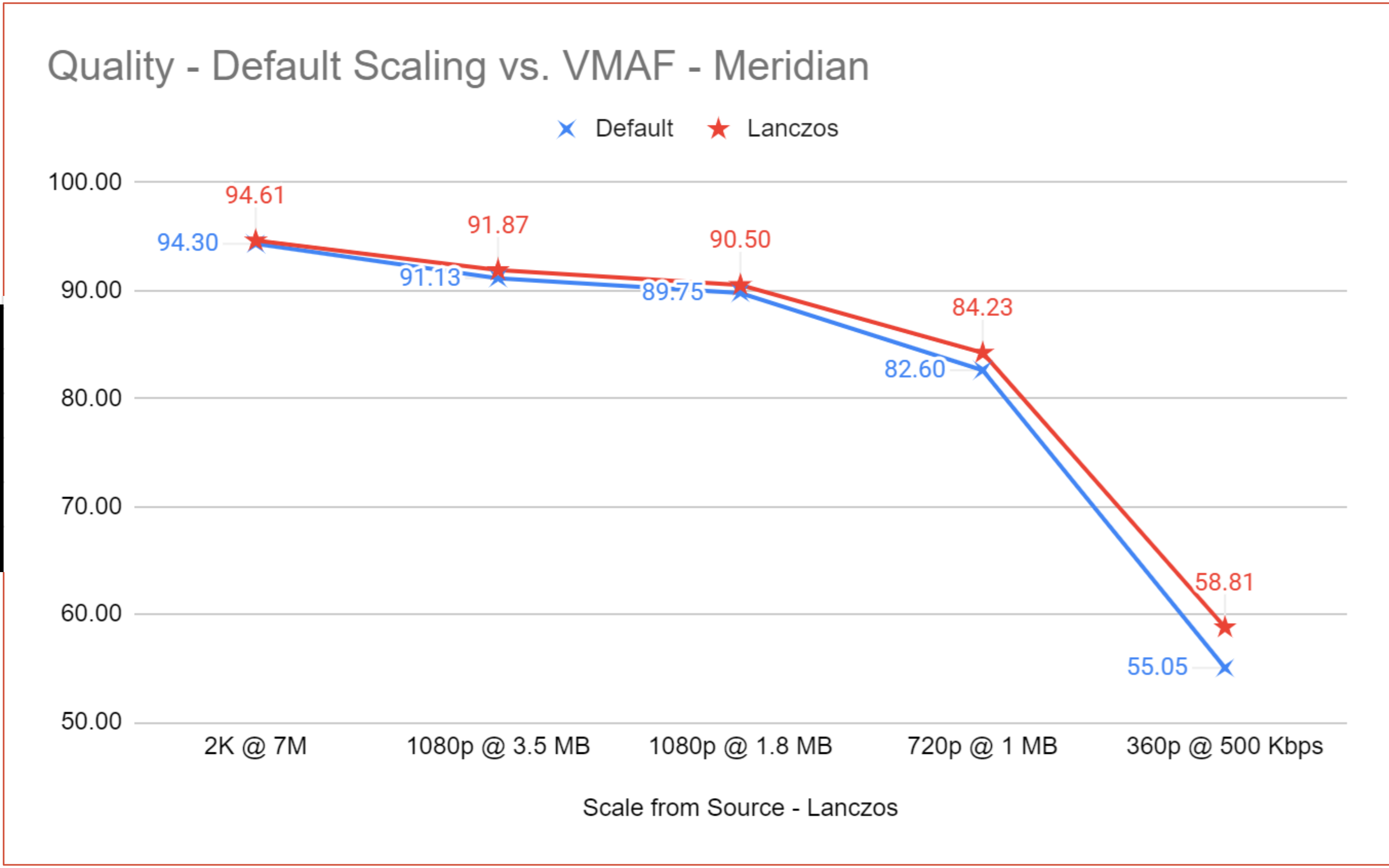
https://bit.ly/42pazmC

- FFmpeg default scaling is bilinear
- Tested three other methods, best was lanczos

- Ffmpeg presentation:
  - -vf scale=640×360 -sws_flags lanczos
  - Not –s 640x360 (which uses bilinear)

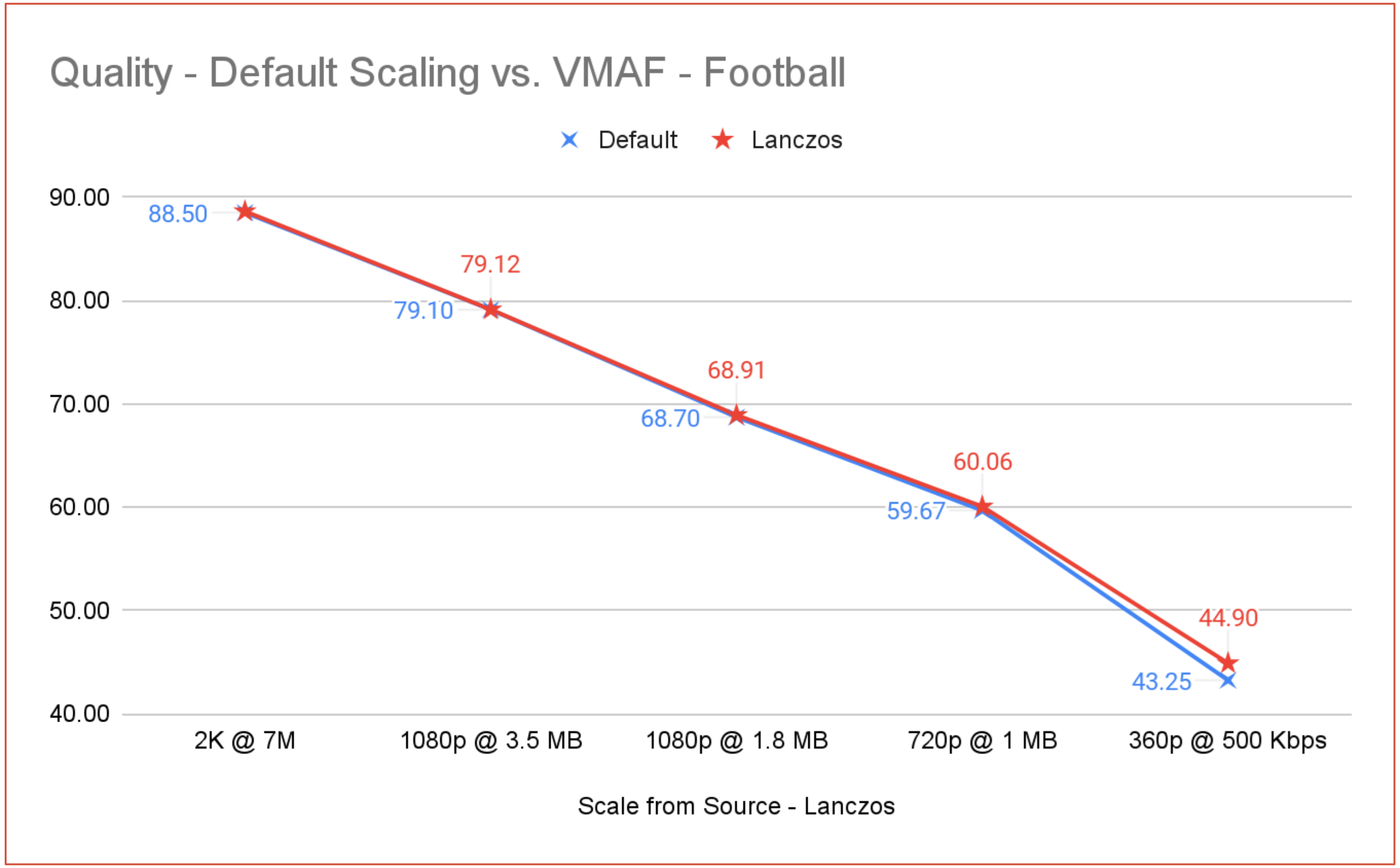
Scaling - Meridian

Meridian	Default	Lanczos	Delta
2K @ 7M	94.30	94.61	0.31
1080p @ 3.5 MB	91.13	91.87	0.75
1080p @ 1.8 MB	89.75	90.50	0.75
720p @ 1 MB	82.60	84.23	1.62
360p @ 500 Kbps	55.05	58.81	3.76



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



Best Practice Scaling – Use Lanczos Where Available

- Lanczos delivers .75 VMAF improvement @ 1080p in Meridian (movie clip)
  - 3.76 VMAF points @ 360p
- There's no downside – encoding time isn't impacted
  - At least with VOD presets (may be some impact live)