Cross-codec encoding optimizations for video transcoding

Gaurang Chaudhari, Hariharan Lalgudi, Harikrishna Reddy
Facebook Inc, 1 Hacker Way, Menlo Park, CA, USA 94025

ABSTRACT

A video on demand (VOD) server generates multiple video output qualities (bit rates), resolutions and codecs to best video quality for all viewers' internet connection. While each codec optimizes tools and does computation-quality trade-off there isn't much work to exploit computation reduction across codecs. In this work, we propose some methods to achieve this. Specifically, we use VP9 mode decision to reduce the computational requirements of AV1 encoding.

Keywords: video transcoding, cross codec encoding, computation reduction, rate distortion, mode decision, VP9, AV1, computation-quality trade-off

1. INTRODUCTION

In a world where video content is exploding and poor-quality video directly influences user engagement and quality of experience (QoE), it is paramount to invest more resources into system design for video related pipelines. To optimize for the viewer's device and available internet connection, for the same video content a VOD server generates multiple output quality and resolution (or bit rate) ladders, a process known as adaptive streaming.

The transcoding flow is broken into a number of distinct steps, outlined below in more detail. The first stage of video transcoding is called decoding, in which an uploaded file is decompressed to get raw frames. These uncompressed frames are then scaled - typically at a lower than the source resolution - to change their resolution, and each resolution is encoded again using, ideally, optimized settings. The output video is typically compared with the original source to calculate quality metrics, an operation that, depending on system design, can involve yet another decode and up-sample of encoded videos. In an ideal world, this is preferred to be done on all videos to ensure that the encoding settings used produces a good-quality output. But to optimize for scale, the last decodes and quality metrics calculations are usually done outside of the main transcoding pipelines for select few content, either for experimentation or based on low quality of experience (QoE) reported from the user. All transcoding steps are typically run in multi-core CPUs in video-on-demand (VOD) applications.

If we consider the popular codecs VP9[1][2]/AV1[3][4], the compute complexity of AV1 is much higher than VP9. The AV1 encoding time is measured to be 114 times the VP9 encoding time[5]. There have been considerable recent efforts to libaom[6] the reference, open-source codebase implementing AV1, to bring this factor down to 10x run-time increase from VP9. In addition to computation increase, changes to video pipelines depend on the market adoption and the available client devices to decode the encoded bitstreams. Due to this, the decision to modify and fully support newer codecs in a video infrastructure takes time. And even if it is supported, usually the best quality preset (or lower cpu_speed) of a newer codec is rarely used. A practical SW encoder preset is used as a trade-off based on the compute resource capacity/availability vs. the coding gain that the preset gives. In other words, a preset of newer codec is chosen that will give quality gains with a similar compute/power profile of the older (predecessor) codec preset currently deployed in a video processing system. This varies according to scale and requirements. In this work, we take an alternative approach of leveraging encoding decisions from one codec to another. This can be effective when a new codec is an evolution of older codec, such as VP9 to AV1 and HEVC to VVC, where more tools are added to realize bit-rate savings for the same quality.

The paper is organized as follows. Section 2 describes various methodologies to enable cross-codec data sharing. In Section 3, we give a brief description of VP9 and AV1 encoding toolsets. Section 4 describes RD-optimized mode decision performed in prevalent encoders (HEVC, VP9, AV1). Section 5 describes usage of VP9 data to AV1 encoding. We present results in Section 6 and conclude the paper in section 7.

2. CROSS-CODEC DATA SHARING

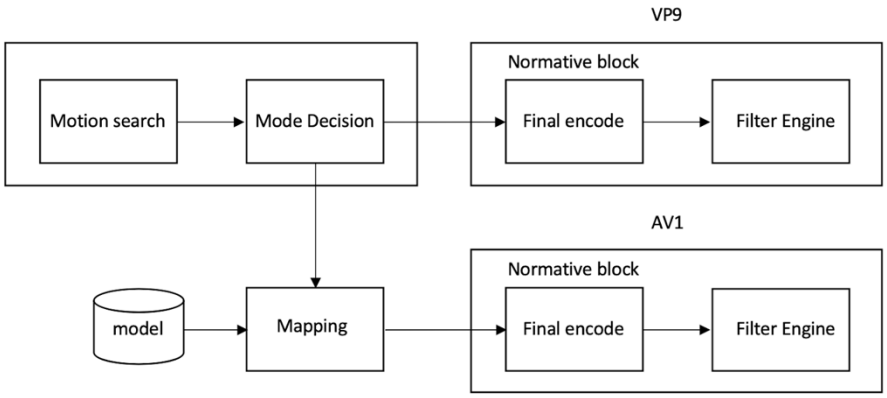


Figure 1: Key components in video encoding and cross codec sharing

Figure 1 shows the key components in a typical video encoding process, Motion search, Mode Decision, encode block (prediction, T, Q and entropy coding) followed by filtering. In this paper, we propose to share the decisions of motion estimation and mode decision data from one codec into another, thereby saving the computational requirements of the system. The qualified data to be shared should satisfy at least one if not all of the following –

(1) It should be compute-intensive

(2) It needs to conform directly or through a mapping with the standard (normative) requirement of the target codec/s

(3) The coding inefficiency is acceptable after reusing this cross-codec data

(4) The overhead it generates to the target codec can be absorbed seamlessly

The shared data can be categorized into 2 buckets:

(1) Directly use the data as long as it is applicable to auxiliary codec. Examples include motion vectors, block shapes, block type (inter/intra)

(2) Use the data as the starting point to do a mapping or for a refinement search to find the final motion, shapes and modes

Computational-quality trade-off is taken into account while deciding on which bucket to use for each of the encoding decisions. In the second method, for doing a refinement search or a mapping, there are multiple ways. Examples include first pass statistics, calculated block-based statistics, machine-learning (ML) based etc. In addition to the final encoding, many VOD systems employ a convex hull approach to decide the best rate control settings for encoding. This is another avenue where cross codec data can be shared.

This cross-codec data sharing not just helps in software encoding but also helps in a hardware ecosystem. The host server CPUs on which the transcode jobs get scheduled are not efficient as VOD gets scaled for the growing number of video contents. In response to the tremendous growth, on May 14, 2019, Facebook publicly announced a video transcoding custom-chip design, codenamed “Mount Shasta", to be deployed in our data centers for the acceleration and compute optimization of our various video products[7]. For codecs, which are getting adopted well in edge devices, ASIC implementation of well optimized video encoding is a solution that can help with the scale issue. One problem with offloading to ASIC is that current market adoption of codecs along with the forecasted adoption of newer codecs drive the ASIC specification. All market changes during and after deployment of ASICs will be absorbed by software encoders and the stop-gap solution falls on the general-purpose/host server CPUs. The time and cost to support a new codec is expensive. If we analyze current edge devices, AV1 HW decoders are much less than competing standards and hence it is natural to use AV1 software encoding for some contents to help codec adoption in the ecosystem, thereby enabling realization of significant bandwidth savings offered by AV1. Given that AV1 encoding computational requirements are much higher, using VP9 encoding decision is a good way to reduce workload.

The cross-codec qualified data can optionally be generated by ASICs and then reused by CPUs to do a minimal refinement search and the final encode (normative) part of the newer codec. Thus, cross-codec data sharing can be done completely in software (CPU) or if someone is planning for specialized hardware for few codecs, a hybrid HW-SW solution can be utilized, where HW-generated data can be hooked up to a software encoder for the newer codec.

3. VP9 AND AV1 ENCODER TOOLSETS

In this section, we want to give a brief introduction on the toolset/feature selected in VP9 (libvpx-1.8.0[8]) and AV1 (libaom-2.0.0[6]) codebases.

As mentioned in section 2, cross-codec data sharing not just helps in software encoding but also helps in a hardware ecosystem. For VP9 encoder, we have an internal experimental codebase, which uses libvpx-1.8.0[8] as baseline with the modifications to the toolsets and encoding algorithm as shown in the table.

Table 1. Modifications to libvpx-1.8.0[8] toolsets and encoding algorithm.

Feature	Modifications
Fixed Transform size	Transform size search depth was set to 0
Coefficient optimization	Disabled in both mode decision and final encode path
Motion estimation	Separated from mode decision rate-distortion optimization logic
For sb < 16x16	Only square partition shapes evaluated in inter search
For sb < 8x8	No compound prediction
Loop filter level	Decided by Q, instead of binary search
SSE/Distortion calculations in mode decision	Not performed in transform domain
Dynamic early termination in partition	Disabled: Threshold based rectangular partition search skip, Skip rectangular partition test when larger block size gives better rdcost
Dynamic early termination in mode search	Disabled: conditional_skipintra, mode_skip_start, mode_search_skip_flags
Static early termination in partition	Disabled: skip some partition based on source noise energy, skip computing inter modes if ARF is not available
Static early termination in mode search	Disabled: intra modes other than DC PRED for blocks with low variance threshold for intra skipping based on source variance, check if NEARESTMV / NEARMV / ZEROMV is the cheapest way to encode zero motion

These modifications are based upon HW complexity as opposed to SW which is what libvpx-1.8.0[8] is based upon. To experiment the idea of cross codec data sharing, we focus on re-using much of the data from VP9 to get an AV1 encoder implementation with minimal overhead for doing the normative parts - essentially, entropy coding, reconstruction loop and in-loop filtering. To this effect, we first take a look at an AV1 encoder without toolset enhancements from VP9 as well as features that do not require considerable additional computation, compared to VP9. The following table summarizes those features that can be used to get a VP9 equivalent version of AV1 encoder. We term this as "AV1_VP9eq" in the rest of the paper. Most toolset enhancements that can be controlled at sequence level are disabled

since they don't incur any rate penalty. One exception is post processing filters. These filters give good visual quality gains and can also be implemented effectively with parallel computations.

Table 2. Features to get VP9 equivalent version of AV1 encoder.

Feature	Brief Description	Usage in "*AV1_VP9eq*"
Super Block Size of 128x128	Maximum Block size where partitioning into different shapes begins	Restricted to 64x64
Intra Modes	3 additional smooth modes (non-directional) and 56 directional modes	Use HOG (Histogram of gradients) to get top 3 angles for consideration
Extended reference frames (7)	Last, Last2, Last3, Golden, Altref, Altref2, BwdRef	Restricted to 3 references per frame (2 previous and 1 bwd)
Dynamic referencing of MVs	Uses sophisticated weighting mechanism of spatial and temporal predictors	Enabled
Constrained Directional Enhancement Filter (CDEF) and Loop Restoration (LR)	Sophisticated in-loop post processing filters to remove compression artifacts	Enabled (low complexity search. Only Wiener filter is used in LR)
Extended partition type	Apart from square and rectangular shapes, AV1 allows T shape and 4:1 shapes	Disabled
Recursive-filter for Intra prediction	Filters neighboring pixels before using them for intra prediction	
Intra Block Copy	Block prediction from already coded pixels in the same frame (beneficial for screen content)	
Extended transform kernels	More choices of transform including rectangular transforms	
Warped Motion compensation	Local affine transforms limited to small degrees	
OBMC (Overlapped block Motion compensation)	Gives smooth combination of predictors from different MVs	
Advanced compound modes	Inter-intra Masked prediction Wedge prediction	
Other Misc tools	Dual filter, Palette mode, Chroma from Luma	

4. RD-OPTIMIZED MODE DECISION

In this section we describe algorithmic details of making rate-distortion optimized mode and partition decisions prevalent among many encoders such as HEVC, VP9, AV1 etc. These encoders start with a superblock of size 64x64

(128x128 in AV1) which can be split recursively using split flags. Figure 2 shows the block sizes at different levels. Each square split can be coded as is, a 2Nx2N block partition or 2 rectangular block partitions (2NxN / Nx2N) or 4 NxN block partitions. AV1 has further extensions of block shapes in each recursive split, but as mentioned earlier we use only rectangular and square shapes in AV1_VP9eq encoder. A partition block can be coded as an inter block or Intra block. In an inter block, a motion vector (MV) is used to find the prediction block from previously coded frames (reference frames). Intra block uses spatially neighboring pixels to get the prediction. All decisions are made using rate distortion cost expressed as $D + \lambda R$. Distortion (D) typically is the mean square error between original and reconstructed pixels. Rate R is bits taken by quantized coefficients. λ is a Langrangian multiplier parameter to trade off quality for bitrate and is typically determined based on the quantization step size.

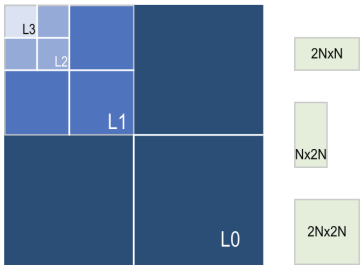


Figure 2: Prediction unit structure in VP9

Partition split decision process is done in a recursive z-scan order. For VP9, we support 13 different prediction sizes. For each square partition (2Nx2N) at level Li, there are 8 different partitions: 4 NxN, 2 - 2NxN, 2 - Nx2N which account for 3 rdcost values (Exception is at 64x64 level where an additional process for 2Nx2N partition is done). The NxN partition at next level is equivalent to the 2Nx2N partition at current level. This is explained in the flowchart in Figure 3.

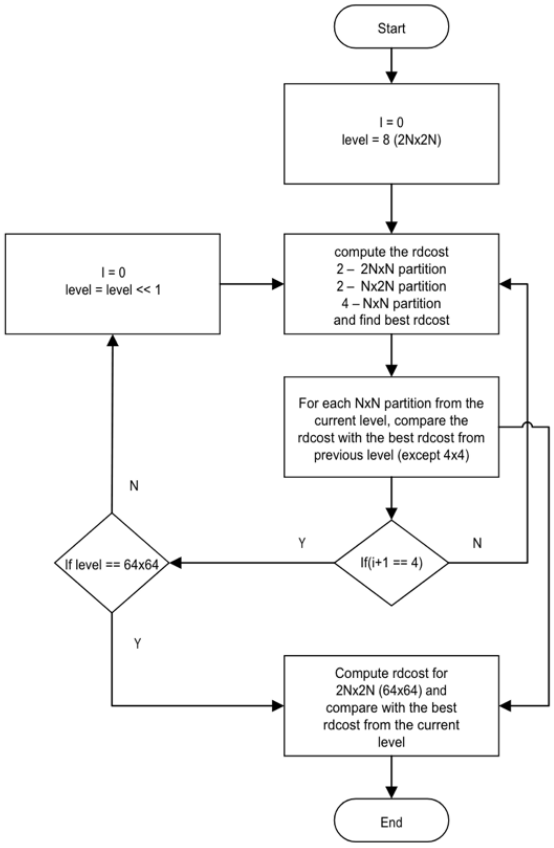


Figure 3: RD-optimized Mode Decision

5. AV1 ENCODING USING VP9 DATA

Though the algorithmic flow in mode decisions are similar between codecs, there are challenges when we try to make a common R-D optimized mode decision for multiple codecs to realize computational savings. 1. Intra/Inter Prediction, transform, quantization, rate and entropy coding are codec specific, resulting in different RD cost. 2. Predictive MV inter modes (Nearest, Near MV) derivation are different and directly using them from one codec to another will result in non-normative parts or have detrimental effect to coding efficiency. 3. Group of Pictures (GOP) structure has to be the same to share motion estimation results. Designing a common effective mode decision module is a complex problem. In this paper, we focus to keep best results for VP9 while still being able to leverage them for AV1 encoding. Our motivation for this choice is the practical use case, where we expect deployment of VP9 bitstream to be more prevalent in the immediate future and offering an AV1 encoder is targeting to enable and grow the video codec ecosystem. Hence, we choose to make best mode decisions for VP9 encoding and re-use and refine the results from it to have an AV1 encoder. We term this as "AV1_VP9share".

AV1_VP9share encoder is derived from AV1_VP9eq, where partitions and modes from VP9 are reused/refined to have a computationally inexpensive AV1 encoder. The partition split, block shape (2Nx2N, 2NxN, Nx2N), inter/intra decision is used as is from VP9. Remaining decisions are the intra mode and the inter mode. For intra mode, we do not use SMOOTH predictors. Non-angular modes are mapped from VP9 to AV1 directly. (True motion to PAETH, DC->DC and Planar to Planar). Histogram of gradient (HOG) is a well-known method to determine dominant directions for selective angular intra modes. In libaom[6], it is used to prune the number of angles that are searched using full RD-cost. It gives around 1.2% quality loss but can reduce the number of modes from 56 to 6. If VP9 picks an angular mode, we use HOG to get the top 2 angles and add them to the best angle from VP9 mode to get a total of 3 candidate angles. A total of 9 angles (3 candidates with +/-1) are searched using full RD optimization. For inter frames, NEW MV along with reference ID from VP9 is used as is and its RD-cost is established to compare with other inter modes. Nearest and Near MVs are computed based on AV1 specification and its RD-cost is calculated. Though the inter mode decision involves calculating accurate RD-cost of AV1, the number of shapes and partitions for which this is computed is substantially reduced to give necessary computational savings.

6. RESULTS

Figure 4 presents coding efficiency results using Bjontegaard delta bit rate (BD-rate)[9] with respect to three popular video quality metrics on 3 different datasets[10][11][12]. First dataset shown in the figure is Facebook (FB) internal dataset[10] which are 400 top-viewed public videos from Facebook (FB) Pages and has video contents across different resolutions. These videos were tested in an anonymized manner without subjective analysis. Second and third are the widely accepted JVET[11] and DERF[12] datasets. We have excluded content with resolution higher than 1080p, since most current uploads in VOD do not exceed HD. BD-rate metric gives average bit-rate reduction (-ve) or increase (+ve) for the same quality between 2 encoding methods for Constant Quality mode. Three different video quality metrics are presented: SSIM[13], PSNR, VMAF[14]. The CRF/QP values used are {33, 39, 45, 51}[10] and for VMAF calculations, we use the vmaf_v0.6.1.pkl model. All comparisons are made with respect to libvpx-1.8.0[8] preset 1 (cpu_speed 1). Command Lines used are given below -

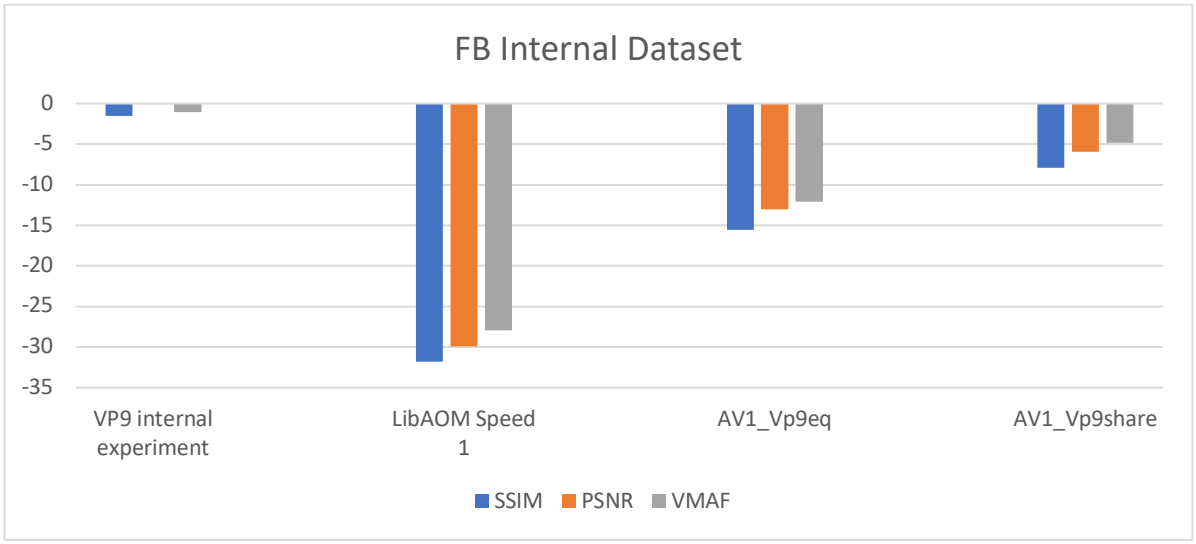
--ivf -o <output_video.vp9> <Input y4m> --codec=vp9 --verbose --passes=2 --limit=300 --i420 --profile=0 --cpu-used=1 --fps=30 --kf-min-dist=60 --kf-max-dist=60 --arnr-maxframes=7 --arnr-strength=5 --lag-in-frames=19 --aq-mode=0 --bias-pct=100 --minsection-pct=1 --maxsection-pct=10000 --end-usage=q --cq-level=<val> --min-q=0 --max-q=63 --auto-alt-ref=5 --max-gf-interval=16 --min-gf-interval=4 --frame-parallel=0 --threads=1 --tile-columns=0

<Input y4m> --ivf 1 --codec=av1 --passes=2 --cpu-used=1 --threads=0 --profile=0 --lag-in-frames=19 --min-q=0 --max-q=63 --auto-alt-ref=1 --kf-max-dist=60 --kf-min-dist=60 --drop-frame=0 --static-thresh=0 --bias-pct=50 --minsection-pct=0 --maxsection-pct=2000 --arnr-maxframes=7 --arnr-strength=5 --sharpness=0 --undershoot-pct=100 --overshoot-pct=100 --tile-columns=0 --frame-parallel=0 --test-decode=warn -v --psnr --end-usage=q --cq-level=<val> -o <output_video.av1> --limit=300

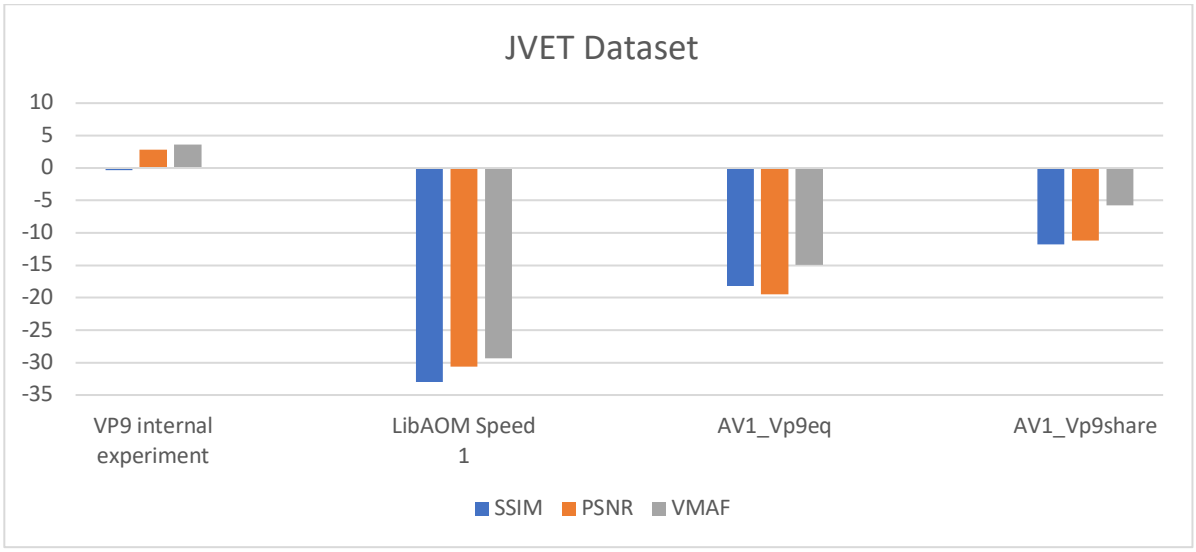
First column set in figure 4 shows that our internal VP9 experimental codebase is close to libvpx-1.8.0[8]. Second column set shows BD-rate reduction of libaom-2.0.0[6] at preset (cpu_speed) 1. As expected AV1 has significant gains (30-35%) compared to VP9. Since libaom[6] implementation is active in getting optimized, we haven't compared its computational

speed. But it is well accepted that AV1 toolset compute requirements are significantly higher than VP9. Third column set shows the quality of AV1_VP9eq encoder described in Section 3. It is interesting to note that even without computationally expensive features we can get good gains (10-16%) which comes from better design of intra/inter modes, adaptive entropy coding, inter referencing structures and post processing filters. Reducing the number of reference frames from 7 to 3 gives 0.7% loss w.r.t SSIM. Adding post-processing filters gives 5.6% gain w.r.t SSIM.

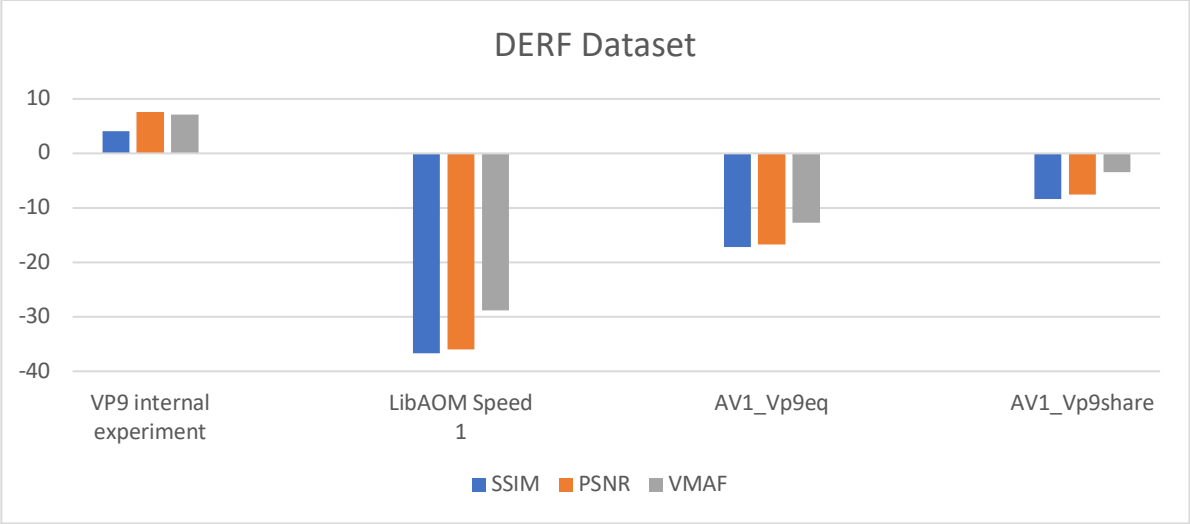
AV1_VP9share quality is shown in the last column set. We get a loss of 6-8% when we compare the BD-rate number of AV1_VP9share w.r.t AV1_VP9eq. This comes primarily from inaccurate RD-cost usage since VP9 codec datapath is used to make those decisions. Nevertheless, we still get a gain of 5-10% compared to VP9 with little additional computations.



(a)



(b)



(c)

Figure 4: Video quality results. (a) results for FB internal dataset (b) results for JVET dataset (c) results for DERF dataset

7. CONCLUSION AND FUTURE WORK

In this paper, we presented the idea of leveraging encoding decisions across codecs. This is very useful for reducing computation in VOD servers to scale for rising video usage. We applied VP9 encoding decisions to AV1 and demonstrated feasibility of this approach to get computation-quality trade-off. A more effective approach would be to prune the partitions and intra/inter selection to a few candidates and use accurate RD-cost of the codec to make the final decision. One other avenue for reducing computation load at VOD would be to use data from decode to limit the search during encoding. Mapping of data from one codec to another can be improved by using learning algorithms by training on the data from decode.

REFERENCES

[1] Mukherjee, Debargha et al., (2015). A Technical Overview of VP9--the Latest Open-Source Video Codec. SMPTE Motion Imaging Journal. 124. 44-54. 10.5594/j18499.
[2] VP9 Bitstream and Decoding Process Specification, https://www.webmproject.org/vp9/
[3] Y. Chen et al., "An overview of core coding tools in the AV1 video codec," in Proc. Picture Coding Symp., 2018.
[4] AV1 Bitstream & Decoding Process Specification, https://aomediacodec.github.io/av1-spec/av1-spec.pdf
[5] D. Grois, T. Nguyen, and D. Marpe, "Performance Comparison of AV1, JEM, VP9, and HEVC Encoders," SPIE, vol. 10396, Sep 2017.
[6] Libaom-2.0.0, https://aomedia.googlesource.com/aom/+/refs/tags/v2.0.0
[7] Lee, K. and Rao, V., "Accelerating facebook's infrastructure with application-specific hardware." https://engineering.fb.com/data-center-engineering/accelerating-infrastructure/, 2019.
[8] Libvpx-1.8.0, https://chromium.googlesource.com/webm/libvpx/+/refs/tags/v1.8.0
[9] G.Bjøntegaard,"Calculation of average PSNR differences between RD-curves (VCEG-M33)," in VCEG Meeting (ITU-T SG16 Q. 6), 2001.
[10] Yu Liu, Open Source, Video Engineering, "AV1 beats x264 and libvpx-vp9 in practical use case", https://engineering.fb.com/video-engineering/av1-beats-x264-and-libvpx-vp9-in-practical-use-case/, April 2018
[11] JVET video dataset, https://jvet.hhi.fraunhofer.de/
[12] Xiph.org Video Test Media [derf's collection], https://media.xiph.org/video/derf/

[13] Zhou Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," in IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-612, April 2004, doi: 10.1109/TIP.2003.819861.

[14] Li, Z., Aaron, A., Katsavounidis, I., Moorthy, A., and Manohara, M., “Toward a practical perceptual video quality metric,” https://netflixtechblog.com/toward-a-practical-perceptual-video-quality-metric-653f208b9652, 2016.