GUIDED JUST-IN-TIME TRANSCODING FOR CLOUD-BASED VIDEO PLATFORMS

Thomas Rusert, Kenneth Andersson, Ruoyang Yu, Harald Nordgren

Ericsson Research
164 80 Stockholm, Sweden

ABSTRACT

Consumption of streamed video on-demand (VoD) content is driving rapid growth of fixed and mobile network traffic. VoD services are commonly run on cloud-based video platforms, where all processing is software-based. The VoD content is typically delivered using adaptive bit rate (ABR) streaming techniques. As the amount of content in an asset library grows, the required storage space and the associated cost increase. This is emphasized if several ABR representations per content are stored. To save storage space, lower quality ABR representations may be eliminated and re-generated based on the corresponding high quality representation as the content is requested, which is referred to as just-in-time (JIT) transcoding. In this paper, we investigate a scheme that can significantly reduce the computational complexity associated with JIT transcoding, called guided JIT transcoding. We analyze its performance when multiple HEVC-coded ABR representations with different spatial resolutions are utilized, showing that for a configuration with seven video representations with resolutions ranging from 1080p to 360p, storage requirements can be reduced by about 24% while software-based transcoding from 1080p to 720p can be performed at 46 fps on average using a single execution thread.

Index Terms— video, transcoding, ABR, cloud, HEVC

1. INTRODUCTION

Video consumption is driving rapid growth of fixed and mobile network traffic. Being the dominant traffic type already today, video is expected to drive the overall network traffic to a multiple of today's volume and account for more than 70% of all network traffic within few years [1][2]. The growth is primarily driven by streamed video on-demand (VoD) content, as consumers increasingly demand access to any content on any device at any time [3].

VoD services are commonly operated on cloud-based video platforms, wherein all processing is executed in software running on generic servers, as such platforms can provide beneficial properties related to scalability, cost efficiency, and ubiquitous availability. VoD content is typically delivered using adaptive bit rate (ABR) streaming techniques, where each video asset is made available in several different representations coded at different bit rates and quality levels (which in this paper is used synonymously unless stated otherwise) so that video clients can choose representations according to bandwidth availability and device capabilities [4]. Video transcoding is a key element in ABR video preparation, and it is known to be very computationally demanding, in particular due to mode decision and motion estimation processes associated with it [5]. This is particularly true for the latest video coding standard HEVC with its high amount of coding flexibility [6].

Fig. 1 depicts a simplified workflow on a video platform, where original video content is ingested into the system, transcoded into several representations, and stored for later delivery to ABR clients. Since transcoding is done immediately after content ingest, we denote the workflow as *upfront transcoding*. As the amount of assets in a content library grows, the required storage and the associated cost increase. This is emphasized for ABR video since several representations per content are stored. At the same time, VoD libraries may exhibit large spread in asset popularity, and some VoD content may be requested infrequently. For instance, in cloud-based DVR (digital video recording) systems that enable time-shifting functionality for linear TV, the majority of viewing is expected to happen within few days after a content asset has been ingested into the cloud DVR. In order to save storage space for infrequently requested content, only the highest quality representation may be kept while the corresponding lower quality representations are eliminated and re-generated on the fly as they are requested. The approach is referred to as *just-in-time (JIT) transcoding*, see Fig. 2. Compared to upfront transcoding, JIT transcoding is significantly more computationally demanding since transcoding needs to be performed every time the content is requested, and has to be done in real-time or faster in order to serve user requests without lag. In contrast to that, upfront transcoding is done only once per content asset, and it does not have strict real-time requirements. The high computational complexity associated with JIT transcoding is a major obstacle in deploying it.

In this paper, we investigate a transcoding scheme that significantly reduces the computational complexity associated with JIT

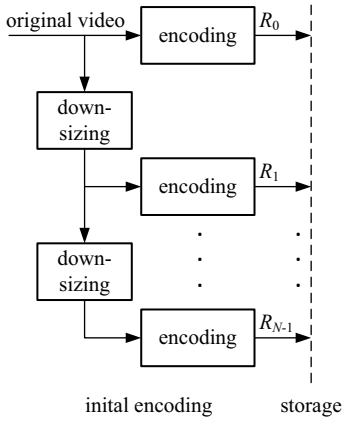

Fig. 1: Video workflow with upfront transcoding and N ABR representations. The original video is downsized to each target resolution and subsequently encoded at different bit rates R_i, where R_0 corresponds to the highest bit rate representation, and $R_i < R_{i-1}$ for $i = 1 \ldots N\text{-}1$. All representations are stored.

transcoding. We call it *guided JIT transcoding*. Basic idea with guided JIT transcoding is to execute the most complex part of the transcoding process, i.e. mode decision and motion estimation, in an upfront fashion for each representation below the highest quality representation, storing modes and motion vectors obtained as control streams to be utilized when the actual JIT transcoding operation is performed. A similar scheme has been presented in [7]. Compared to [7], we make several new contributions. First, we investigate the performance of guided JIT transcoding in the context of HEVC-coded ABR video. Second, we provide results both for storage reduction, transcoding-induced overhead and computational complexity, the latter two aspects not being investigated in [7]. Third, we introduce an approach named *partial pruning* which can further significantly reduce the computational complexity of the guided transcoding process while only slightly reducing the storage savings obtained. Unlike scalable video coding [8], which can in principle serve as alternative solution for reducing storage requirements for ABR video, the solutions considered in this paper do not require specific decoding functionality at the ABR clients, thus minimizing client-side deployment complexity.

The paper is organized as follows. In Sect. 2, application of JIT transcoding for ABR video is discussed. In Sect. 3, guided JIT transcoding and partial pruning are described. Sect. 4 presents experimental results. Concluding remarks are given in Sect. 5.

2. JIT TRANSCODING FOR ABR VIDEO

Fig. 1 and Fig. 2 depict video workflows with upfront transcoding and JIT transcoding, respectively, using N representations. Bit rates per representation are denoted as R_i and R'_i, respectively, where $i = 0$ indicates the highest quality representation. It is assumed that the ingested original video, which would typically be compressed at high quality, has already been decoded. Also, operations such as ABR segmentation, multiplexing and encryption are omitted from the illustrations for simplicity. In case of upfront transcoding as depicted in Fig. 1, all encodings can use the original video as input, after it has possibly been downsized (in this paper "downsizing" should be interpreted as an optional operation, i.e. subsequent representations may or may not have identical resolutions). The N bitstreams resulting from the upfront transcoding operation are stored and delivered to video clients on demand.

In case of JIT transcoding, we assume that only the highest quality bitstream is stored, and each target bitstream is generated based on the highest quality bitstream at the time when it is requested by an ABR client, see Fig. 2 (note that a content asset may initially be upfront-transcoded, and at some point when the content popularity has gone down, the initial lower quality representations may be eliminated for storage saving). We further assume that the respective highest quality bitstreams are identical for upfront transcoding and JIT transcoding, thus $R'_0 = R_0$. In contrast to their counterparts from upfront transcoding, the lower quality representations generated by the JIT transcoding are created based on an already degraded input signal. Therefore, assuming equal quality (in terms of PSNR using the original video as reference) between corresponding lower quality representations, it can be expected that $R'_i > R_i$ for $i = 1...N\text{-}1$. We denote the difference between R'_i and R_i as *transcoding-induced overhead*.

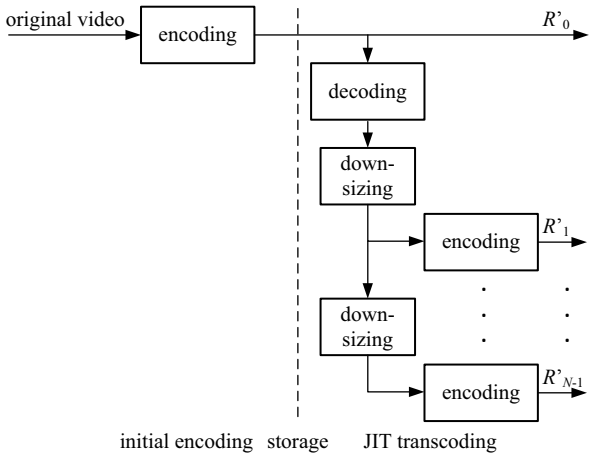

Fig. 2: Video workflow with JIT transcoding and N ABR representations. The highest bit rate representation is encoded at bit rate R'_0, based on the original video. It is the only representation that is stored. When a lower bit rate representation is requested, the highest bit rate representation is decoded, downsized to the target resolution, and encoded at target bit rate R'_i for $i = 1...N\text{-}1$.

3. GUIDED JIT TRANSCODING FOR ABR VIDEO

3.1. Basic Guided Transcoding Approach

Fig. 3 depicts a video workflow with guided JIT transcoding using N representations. The highest bit rate representation is encoded at bit rate R''_0, based on the original video. The reconstructed highest bit rate video is downsized to each target resolution and subsequently encoded at different bit rates R^*_i with $R^*_i < R^*_{i-1}$ for $i = 2\ldots N\text{-}1$. In order to realize storage savings, e.g. at a time when the content popularity has gone down, the lower quality bitstreams are pruned by removing transform coefficients from the streams,

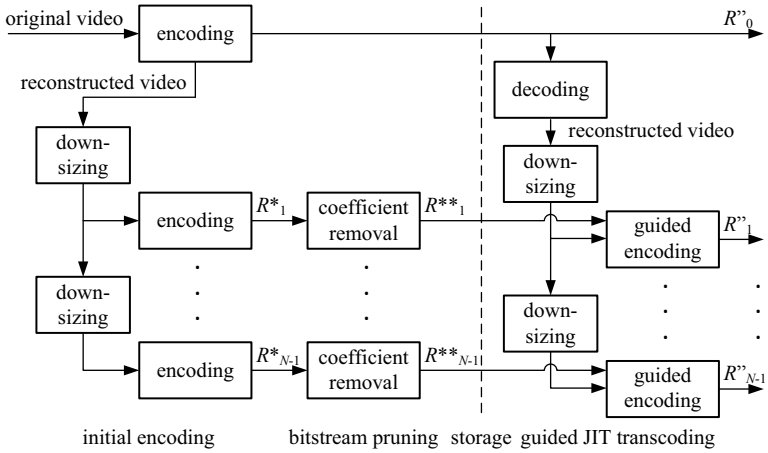

Fig. 3: Video workflow with guided JIT transcoding and N ABR representations. The highest bit rate representation is encoded at bit rate R''_0, based on the original video. For each lower quality representation, $i = 1...N\text{-}1$, a control stream at bit rate R^{**}_i is generated by means of initial encoding followed by bitstream pruning. When a lower bit rate representation is requested, the highest bit rate representation is decoded, downsized to the target resolution, and then encoded by utilizing mode information and motion vectors from the respective control stream, yielding a bitstream with bit rate R''_i.

yielding control streams that contain only mode and motion data (including related parameters such as block partitioning data, QP values, deblocking and SAO parameters) [6][7]. The control streams have bit rates R^{**}_i with $R^{**}_i < R^*_i$ for $i = 1…N\text{-}1$. When a lower bit rate representation is requested, the highest bit rate representation is decoded, downsized to the target resolution, and encoded by utilizing mode and motion data from the respective control stream, yielding a bitstream with bit rate R''_i, $i = 1…N\text{-}1$.

Assuming the highest quality bitstreams for JIT transcoding (Fig. 2) and guided JIT transcoding (Fig. 3) are identical (and thus $R''_0 = R'_0$), the lower quality bitstreams before the pruning step in Fig. 3 can be assumed to be identical to the respective lower quality bitstreams in Fig. 2 as well (and thus $R^*_i = R'_i$ for $i = 1…N\text{-}1$) since they have been encoded based on the same input signal (utilizing the reconstructed highest quality representation). We further assume that downsizing operations prior to the initial encoding and guided encoding steps in Fig. 3 are identical, and the initial and guided encoding processes for each respective representation are identical except that the initial encoding operation includes mode decision and motion estimation processes whereas the guided encoding operation only applies the mode and motion data that is obtained in the corresponding initial encoding. Therefore, the output of each corresponding initial encoding and guided encoding are identical, and thus $R''_i = R'_i$ for $i = 1…N\text{-}1$.

3.2. Guided Transcoding with Partial Pruning

Fig. 4 depicts a dyadic hierarchical B picture prediction structure as commonly used in high efficiency video coding applications [9]. Each picture is assigned to a temporal layer, where the temporal layers reflect the temporal scalability properties of the bitstream (e.g. for the given case of a dyadic prediction structure, removing pictures assigned to the highest temporal layer cuts the video frame rate in half) [6]. We denote the number of temporal layers as T. We assume that identical prediction structures are used for high quality representation and lower quality representations, so that the transcoding operation can be performed on a picture-by-picture basis as indicated by the "decoding/transcoding order index" in the figure.

Due to the fact that the distance between predicted picture and closest reference picture in decoding order goes down with increasing temporal layer, and due to the fact that typically the quantization step size in hierarchical B picture prediction structures is increasing with increasing temporal layer, it can be expected that for typical video sequences the amount of non-zero transform coefficients is lower for pictures assigned to higher temporal layers and higher for pictures assigned to lower temporal layers. Therefore, for guided JIT transcoding, the storage saving that can be expected by removing transform coefficients from e.g. a picture associated with the highest temporal layer can be expected to be comparably small. This motivates us to introduce *L level partial pruning* in the guided transcoding approach, wherein in the pruning step (see Fig. 3), transform coefficients are removed only for pictures assigned to temporal layers $0…T\text{-}L\text{-}1$, while they are not removed for pictures assigned to temporal layers $T\text{-}L…T\text{-}1$. Advantage of the partial pruning approach is that for those pictures that have not been pruned, the transform coefficients do not have to be regenerated in the guided encoding step (see Fig. 4). Additionally, due to the fact that pictures associated with a given temporal layer are not required for decoding of pictures associated with lower temporal layers, the decoding and downsizing operations in the guided transcoding process can be omitted for the L highest temporal layers as well. As a consequence, the computational complexity of the guided transcoding process can be significantly reduced.

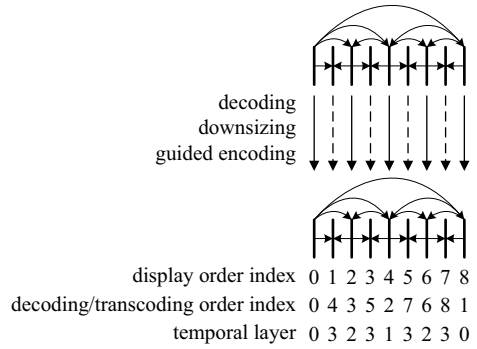

Fig. 4: Hierarchical B picture prediction structures for high (top) and lower quality representation (bottom) in guided JIT transcoding, and relationships between coded pictures of high and lower quality representations. Vertical strokes represent coded pictures. Horizontal arrows represent coding dependencies between pictures, where each respective arrow points from a reference picture to the picture that requires that reference picture for decoding. Vertical arrows represent the processing flow for transcoding a high quality picture into a corresponding lower quality picture, where dashed arrows indicate the processing flow for pictures of the highest temporal layer. Display order index, decoding/transcoding order index, and temporal layer are indicated below each respective picture (the depicted prediction structures have four temporal layers).

4. EXPERIMENTAL RESULTS

Encodings are performed using HM-16.6 with HEVC Main profile random access settings according to [9] and RDOQ turned off. Class B test sequences according to [9] are used. For each test sequence, we generate seven ABR representations with resolutions ranging from 1080p to 360p, with hierarchical B picture prediction structures aligned across representations as depicted in Fig. 4. The 1080p sequences are encoded at four different QP values $QP_{base} \in \{22, 26, 30, 34\}$. For each value of QP_{base}, the lower quality representation encodings according to Figs. 1-3 are performed using QP values as specified Tab. 1. All numbers are averaged over the results obtained for the different settings of QP_{base}. All PSNR numbers are calculated using the original (and possibly downsized) video as reference.

Tab. 2 depicts results for transcoding-induced overhead due to encoding of already degraded input signals in case of both JIT transcoding and guided JIT transcoding, specified as $(R'_i - R_i) / R_i$ according to Fig. 1 and Figs. 2-3 (as mentioned in Sect. 3.1, the encoded bitstreams delivered to the video clients are identical for the cases of JIT transcoding and guided JIT transcoding, thus $R''_i = R'_i$). The numbers are obtained by measuring PSNR and bit rate R'_i according to Fig. 2, and using interpolation [10] to obtain the bit rate R_i that would be required to achieve the same PSNR when using the downsized original video as input to the encoder. As can be expected, the highest overhead is observed for 720p, and the overhead goes down as the target bit rate goes down.

Tab. 3 shows the storage savings achieved through transform coefficient removal in guided JIT transcoding, specified as $(R_i - R^{**}_i) / R_i$ according to Figs. 1 and 3. The numbers are obtained by measuring PSNR and bit rate R^{**}_i according to Fig. 3, and using

Tab. 1: QP settings and target resolutions of ABR representations.

Video resolution	QP
1080p	QP_{base}
720p	QP_{base}, $QP_{base} + 2$
540p	QP_{base}, $QP_{base} + 2$
360p	QP_{base}, $QP_{base} + 2$

Tab. 2: Transcoding-induced overhead in case of JIT transcoding and guided JIT transcoding.

	720p QP_{base}	720p $QP_{base} + 2$	540p QP_{base}	540p $QP_{base} + 2$	360p QP_{base}	360p $QP_{base} + 2$
Kimono	17.0%	11.0%	9.7%	6.3%	4.4%	2.6%
ParkScene	15.3%	9.7%	8.2%	5.1%	3.8%	2.4%
Cactus	14.3%	9.0%	7.6%	4.8%	3.5%	2.1%
BasketballDrive	15.8%	9.9%	8.8%	5.5%	4.1%	2.5%
BQTerrace	16.9%	10.8%	8.8%	5.7%	3.7%	2.4%
Average	15.9%	10.1%	8.6%	5.5%	3.9%	2.4%

interpolation to obtain the bit rate R_i that would be required to achieve the same PSNR using upfront transcoding according to Fig. 1. As can be seen from Tab. 3, the average storage saving for each lower quality representation is around 50%. Considering all seven ABR representations (including the highest quality representation for which no coefficient removal is performed), average storage savings of 29.3% are achieved. In case of conventional JIT transcoding, i.e. if all lower quality representations are eliminated ("No LQ streams"), 57.8% of storage is saved, however even with highly optimized HEVC encoders, the JIT transcoding operation can be expected to require use of massive multi-threading on multi-core CPUs to achieve high compression efficiency under real-time constraints [11].

Tab. 4 depicts average storage savings and relative transcoding complexity for guided JIT transcoding with L level partial pruning for $L = 0 \ldots 3$ (note that $L = 0$ corresponds to the case where all pictures are pruned, see Tab. 3). In order to assess the speed of the guided transcoding operation, we created a software implementation based on which in case of $L = 0$, guided transcoding from 1080p to 720p can be performed at 18 fps on average using a single execution thread on an Intel Core i7 3.3 GHz based PC platform. The relative complexity numbers in Tab. 4 are given as average runtime compared to the case of $L = 0$. Storage savings go down as L goes up since fewer transform coefficients are removed from the lower quality bitstreams. However, in particular for low values of L, the majority of storage savings is maintained, while the relative transcoding complexity is significantly reduced. For example, in case of $L = 2$, storage savings of 24.3% are obtained while relative complexity is reduced to 39.4%, i.e. our single-threaded software implementation transcodes from 1080p to 720p at 46 fps on average.

5. CONCLUSION

We have analyzed transcoding-induced overhead, storage savings, and computational complexity associated with JIT transcoding for ABR video on cloud-based video platforms. Using JIT transcoding, by eliminating lower quality representations, storage requirements for ABR video are reduced at the cost of computational complexity that applies whenever the content is requested. Guided JIT transcoding provides moderately less storage savings than conventional JIT transcoding at substantially lower complexity. This is enabled by removing transform coefficients of lower quality representations while retaining mode and motion data to guide the JIT transcoding process. Transform coefficients associated with higher temporal layers in hierarchical B picture prediction typically exhibit only marginal storage requirements. This has motivated us to introduce partial pruning for guided JIT transcoding, disabling coefficient removal for higher temporal layers thus providing further significant complexity reduction for guided JIT transcoding. Cost-optimal decision on use of upfront transcoding, JIT transcoding, guided JIT transcoding, or combinations thereof, has not been addressed in this paper. It requires consideration of cost for storage and compute as well as content access frequency. In particular for infrequently accessed content, guided JIT transcoding with partial pruning represents a significant alternative to conventional JIT transcoding, providing substantial storage savings while having reasonable computational complexity. In our tests using HEVC-coded ABR video with seven representations, about 24% of storage space is saved while transcoding from 1080p to 720p runs at 46 fps using a single thread on a standard PC platform.

Tab. 3: Storage savings through transform coefficient removal per lower quality representation, resulting total storage saving considering all representations, and storage saving in case lower quality representations are entirely eliminated ("No LQ streams").

	720p QP_{base}	720p $QP_{base} + 2$	540p QP_{base}	540p $QP_{base} + 2$	360p QP_{base}	360p $QP_{base} + 2$	All seven representations	No LQ streams
Kimono	58.2%	57.4%	56.4%	55.1%	55.0%	52.5%	35.5%	63.0%
ParkScene	48.1%	47.4%	49.1%	47.5%	51.7%	48.8%	28.1%	58.0%
Cactus	50.4%	49.1%	51.1%	49.0%	53.4%	50.4%	29.4%	59.2%
BasketballDrive	46.8%	45.7%	47.2%	45.3%	49.4%	46.2%	27.8%	60.2%
BQTerrace	54.0%	53.6%	54.7%	53.2%	56.8%	54.3%	25.8%	48.6%
Average	51.5%	50.6%	51.7%	50.0%	53.2%	50.4%	29.3%	57.8%

Tab. 4: Storage savings and relative transcoding complexity for guided JIT transcoding from 1080p to 720p with L level partial pruning.

	$L = 0$ (no partial pruning)		$L = 1$		$L = 2$		$L = 3$	
	Storage saving	Relative complexity	Storage saving	Relative complexity	Storage saving	Relative complexity	Storage saving	Relative complexity
Kimono	35.5%	100%	33.9%	60.1%	28.3%	37.0%	21.9%	22.8%
ParkScene	28.1%	100%	27.5%	61.5%	25.2%	41.1%	22.3%	27.4%
Cactus	29.4%	100%	27.9%	63.3%	24.3%	41.3%	20.0%	26.8%
BasketballDrive	27.8%	100%	25.6%	61.4%	19.4%	38.2%	13.3%	23.9%
BQTerrace	25.8%	100%	25.5%	62.1%	24.3%	39.7%	22.7%	26.2%
Average	29.3%	100%	28.1%	61.7%	24.3%	39.4%	20.0%	25.4%

6. REFERENCES

[1] Cisco Visual Networking Index Forecast, May 2015.

[2] Ericsson Mobility Report, Nov. 2015.

[3] Ericsson ConsumerLab TV & Media Report, Sep. 2015.

[4] T. Stockhammer. Dynamic adaptive streaming over HTTP--: standards and design principles. In *Proc. ACM Conf. Multimedia Systems MMSys '11*, pp. 133–144, Feb. 2011.

[5] J. Xin, C. W. Lin, and M. T. Sun. Digital video transcoding. *Proc. IEEE 93.1* (2005):84–97, Jan. 2005.

[6] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand. Overview of the High Efficiency Video Coding (HEVC) standard. *IEEE Trans. Circuits Syst. Video Techn.*, 22(12):1649–1668, Dec. 2012.

[7] G. Van Wallendael, J. De Cock, and R. Van de Walle. Fast transcoding for video delivery by means of a control stream. In *Proc. IEEE Int. Conf. Image Proc. ICIP '12*, vol. 2, pp. 733–736, Orlando, FL, USA, Oct. 2012.

[8] J. M. Boyce, Y. Ye, J. Chen, and A. K. Ramasubramonian. Overview of SHVC: Scalable Extensions of the High Efficiency Video Coding Standard. *IEEE Trans. Circuits Syst. Video Techn.*, 26(1):20–34, Jan. 2016.

[9] F. Bossen. Common conditions and software reference configurations. Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, doc. JCTVC-L1100, Geneva, CH, Jan. 2013.

[10] G. Bjøntegaard. Calculation of average PSNR differences between RD curves. ITU-T SG 16 WP 3, doc. VCEG-M33, Austin, TX, USA, Apr. 2001.

[11] HEVC Video Codecs Comparison. MSU Graphics & Media Lab, http://compression.ru/video/codec_comparison/hevc_2015/, Oct. 2015.