Identifying Energy-Saving Opportunities for AV1 Video Coding in Streaming Services

Caroline Souza Camargo, Alex Borges, Guilherme Correa
Video Technology Research Group (ViTech)
Federal University of Pelotas (UFPel)
Pelotas, Brazil
{caroline.sc, amborges, gcorrea}@inf.ufpel.edu.br
0000-0002-7563-4556, 0000-0002-1264-5055, 0000-0002-2739-6194

***Abstract*—Encoding high-quality videos comes at a high energy cost, mainly due to the various encoding techniques implemented in modern codecs, such as the AOMedia Video 1 (AV1) codec. When considering streaming service data centers, this energy consumption becomes much higher, since the encoding is performed several times for the same video to generate bitstreams for adaptive bitrate. In this work, we seek to explore the block partitioning structures of AV1 to identify energy-saving opportunities in the video encoding process in streaming service providers. A statistical analysis is performed to identify the occurrences of blocks encoded at each partitioning depth in the AV1 reference encoder for different quantization configurations. Then, by comparing partitioning depths used at lower quantization levels to depths at higher quantization levels, a proposal to reduce the computational cost and energy consumption of AV1 video encoders in streaming data centers is presented.**

***Index Terms*—video coding, AV1, acceleration, energy consumption**

I. Introduction

The consumption of video streaming has grown considerably in recent years, as can be observed with the popularization of streaming platforms such as YouTube, Netflix, Disney+, and others. According to the report published by [1], the streaming industry grew by 26% in the year 2020. This is due to cultural changes that have taken place in recent decades and also, partially, to the COVID-19 pandemic, which has contributed as a catalyst for this new form of entertainment [2], because of the worldwide "stay home" campaigns. At the same time, streaming services also try to offer a better image and video quality. The operation of these platforms is only possible thanks to efficient algorithms and video compression formats. Also, video transcoding is an important functionality for streaming services workflows since some different video formats need to be supported to guarantee compatibility with different multimedia devices, in addition to a great variability and heterogeneity of user bandwidth [3]. Without video compression, streaming videos over the internet would be unfeasible as the bandwidth requirements for the uncompressed video would be extremely high.

Streaming service providers usually use standards such H.264/AVC or H.265/HEVC in their data centers to generate the encoded bitstreams. However, some of these encoders are surrounded by complex royalty systems [4]. In 2015 the Alliance for Open Media (AOMedia) started developing the AOMedia Video 1 (AV1) [5], a video coding format which offers an increase in compression efficiency of around 30% compared to the VP9 video coding format [5], while maintaining the same level of image quality. Even though the first version of the AV1 reference software (libaom) was launched in July 2018, its adoption is significant, being currently present in 15% of the world market [6]. That way, it is noted that AV1 is a strong candidate to become an increasingly popular video coding format in the coming years, as the streaming service needs efficient encoders to perform video compression in high quality with low cost.

However, to achieve high compression rates AV1 requires a high computational cost, which leads to high energy consumption. According to [5], the increase in AV1 complexity is evident when compared to its predecessor, the VP9, mainly because of the number of tools supported by AV1. Also, the number of block partitions is a very important contributor for this high computational cost: while VP9 allows only four types, AV1 introduced ten partition types, resulting in an increase from 13 block sizes in VP9 to 22 block sizes in AV1. Thus, the set of possibilities that can be tested during encoding is greater than the previous generation format, which leads to an impact both in computational cost and energy consumption of AV1 codecs. In [5] this issue is highlighted, showing that AV1 requires more than 100 times longer encoding times than libvpx, the reference software of VP9.

It is worth mentioning that data servers in streaming providers store a single video in several bitstreams representations, which are encoded under different conditions (e.g., video resolution, quality levels) in order to provide the best bitstream according to the user's requirements. Thus, as there is a very high computational cost and energy consumption in this process, more research and development of techniques to reduce AV1 energy consumption is necessary to allow for greater adoption of the format. This work aims to present a study of how it may be possible in the context of streaming data centers, by exploring the block partitioning similarities and characteristics when encoding different representations of the same content.

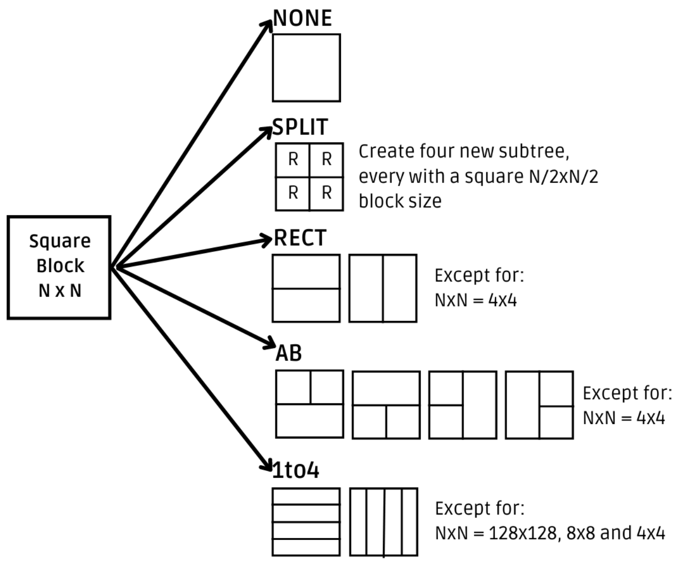
NONE
SPLIT
R
Create four new subtree, every with a square N/2xN/2 block size
Square Block N x N
RECT
Except for: NxN = 4x4
AB
Except for: NxN = 4x4
1to4
Except for: NxN = 128x128, 8x8 and 4x4

Fig. 1. AV1 block partitioning.

II. AV1 Block Partitioning Complexity

AV1 is based on the structure known as the Hybrid Video Coding Model [7], which means that it uses the same type of operations as previous video coding formats. Intra-frame prediction, inter-frame prediction, transform, quantization, and entropy coding are part of this model. Most of these are block-based functions, which means that they operate using blocks of pixels as the basic data structure. Despite these similar characteristics between video coding formats, the complexity of AV1 has increased significantly when compared to the previous format VP9. One of the features increased in AV1 is the block partitioning structure. In order to find the best block size for each region of the video, AV1 allows block sizes from 128×128 down to 4×4. In more complex regions of the video (e.g., complex texture or motion), smaller blocks are usually used, since the amount of detail to represent it is greater. The AV1 block partitioning structure starts from a Superblock (SB), which can be the size 128×128 (maximum block size allowed) or 64×64 [5]. The SB can then be recursively subdivided according to other square or rectangular shapes, as shown in Fig. 1.

The simplest partitioning type present in Fig. 1 is the quadratic block *NONE* of size $N{\times}N$ (where $N \in \{128, 64, 32, 16, 8, 4\}$), which means that this block has not been divided into any other parts. Two other types of divisions that can happen are the horizontal type (*HORZ*), which divides the block in half horizontally, and the vertical type (*VERT*) which divides the block vertically. In these cases the block size will be $N{\times}{}^{N}\!/_{2}$ and ${}^{N}\!/_{2}{\times}N$, respectively. Of these two types of partitioning, they are subdivided into four new ones, categorized as partitioning type *ab*, namely *HORZ_A*, *HORZ_B*, *VERT_A* and *VERT_B*. The *ab* variation is characterized by having a rectangular block on one side, which can be of size $N{\times}{}^{N}\!/_{2}$ or ${}^{N}\!/_{2}{\times}N$, and on the other two square blocks of size ${}^{N}\!/_{2}{\times}{}^{N}\!/_{2}$. Finally, in Fig. 1, we have the *1to4* type that has two variations: first, there is the presence of four horizontal blocks of size $N{\times}{}^{N}\!/_{4}$, which is called *HORZ_4*; second, there is the type *VERT_4*, which is known to have four vertical blocks of size ${}^{N}\!/_{4}{\times}N$.

TABLE I
AV1 PARTITION TYPES ALLOWED FOR EVERY TREE DEPTH

tree depth	**partition types allowed**
0	*NONE*, *HORZ*, *VERT*
1	*NONE*, *HORZ*, *VERT*, *ab*, *1to4*
2	*NONE*, *HORZ*, *VERT*, *ab*, *1to4*
3	*NONE*, *HORZ*, *VERT*, *ab*, *1to4*
4	*NONE*, *HORZ*, *VERT*
5	*NONE*

It is important to note that not every partitioning tree depth will have access to all these partitioning types. For example, at depth 5, only type *NONE* is allowed, and at depths 0 and 4, types *ab* and *1to4* are not accepted, as summarized in Table I. There is also a last type of partitioning presented in Fig. 1, called *SPLIT*, which indicates that no prediction mode will be performed at that depth level of the partitioning tree. That is, an advance in the depth level of the AV1 partitioning tree was triggered, creating four new subtrees that will repeat the search procedure recursively. As there are several ways to perform a block partition and the encoder performs an exhaustive search with all available options, this process ends up being highly expensive.

It is also worth noting the amount of video representations that needs to be stored and processed in streaming data centers. Each video is encoded and stored under different resolutions and quality levels [8], in order to provide a better user experience. In order to implement these techniques, such as Adaptive Bitrate Streaming [8], used by YouTube, and Per-Title Encode Optimization [9], used by Netflix, the video needs to be encoded under different quantization levels, which is controlled by the Constrained Quality (CQ) parameter in libaom. The CQ can assume an integer value between zero and 63 [5] – the closer this value is to 63, the smaller is the resulting bitrate (and the worse is the image quality). Another important aspect is that lower CQs usually lead to longer encoding times, since encoding decisions are performed under greater demands to significantly reduce the loss of quality of the encoded video.

This work aims to analyze statistical data obtained during encoding time to observe the occurrence of block partitions along the encoding process under different quantization levels (CQ). The idea is to observe partitions distributions and devise rules that can be employed in the development of heuristics to reduce encoding time and energy consumption in video streaming data centers.

III. Methodology

As already presented, the block partitioning structure is the basis of every hybrid video coding model and its execution impacts the entire coding process. Therefore, in this work,

TABLE II
IDENTIFYING THE PARTITIONING TREE DEPTH FROM AV1 BLOCK SIZES

tree depth	block size
0	128×128, 128×64, 64×128
1	64×64, 64×32, 32×64, 64×16, 16×64
2	32×32, 32×16, 16×32, 32×8, 8×32
3	16×16, 16×8, 8×16, 16×4, 4×16
4	8×8, 8×4, 4×8
5	4×4

a statistical analysis will be carried out on how the choice of depths throughout the encoding process behaves under different quantization levels and thus try to seek better energy efficiency. This will be done by comparing the occurrences of the depths of a smaller CQ with a larger CQ. That is, by looking at the coding tree depth of the high CQ encoding, it will be possible to understand the probability of the low CQ encoding choosing the same of any other partitioning tree depth. This way, in future implementations it will be possible to provide libaom with such information and avoid coding depths that are unlikely to be chosen for certain regions of the video.

To perform the experiments, 1920×1080 resolution videos available in the recommended test set for AV1 were used [5], since this resolution is one of the most common in streaming services. Of the videos available for testing, five of them were selected: RushHour, NetflixCrosswalk, Netflix-TunnelFlag, RushFieldCuts, and TouchdownPass. To perform the video encodings, the AV1 version 3.3 of reference software, libaom, was used (under the hash code `f9babb`). As recommended [5], we replicated the command line used to run the experiment, under the parameters present below, where CQ represents the quantization level (22, 32, 43, 55), `W` and `H` represent the video resolution, `A` indicates the name of the encoded file and `F` the number of frames per second. As in streaming data centers, where videos are split into chunks of some few seconds, only the first 60 frames of each video were encoded.

```
./aomenc --verbose --psnr --frame-parallel=0
--tile-columns=0 --cpu-used=0 --threads=1
--end-usage=q --lag-in-frames=0 --kf-min-dist=1000
--kf-max-dist=1000 --i420 --width=W --height=H
--fps=F/1 --cq-level=CQ -o A.webm A.yuv
```

To obtain the data that will be used in the statistical analysis, the libaom software was modified to allow the export of information from the partitioning tree along with the encoding with libaom. For each visible frame being encoded, information such as block position (row and column) and block size are exported. After encoding all the videos, an algorithm was implemented in order to interpret the data exported from each frame into its matrix, where each cell of this matrix represents the depth of the partitioning tree of a 4×4 region of the frame. Through the block size, it is possible to identify the depth level of the partition tree, as shown in Table II.

TABLE III
STATISTICAL ANALYSIS ON THE CQ RELATION 32-20, IN PERCENTAGE

		depth on CQ 20						
		0	**1**	**2**	**3**	**4**	**5**	**rule sum**
depth on CQ 32	**0**	0.00	0.00	0.00	0.00	0.00	0.00	–
	1	4.89	45.72	42.59	6.66	0.14	0.00	88.31
	2	2.98	3.31	61.43	31.42	0.86	0.00	92.85
	3	5.64	1.89	8.30	76.16	7.97	0.05	84.13
	4	10.24	2.70	6.44	40.25	39.76	0.61	80.02
	5	8.64	4.36	8.64	24.44	38.11	15.80	78.35
							average all	84.73

TABLE IV
STATISTICAL ANALYSIS ON THE CQ RELATION 43-32, IN PERCENTAGE

		depth on CQ 32						
		0	**1**	**2**	**3**	**4**	**5**	**rule sum**
depth on CQ 43	**0**	0.00	0.00	0.00	0.00	0.00	0.00	–
	1	0.00	72.66	25.65	1.62	0.07	0.00	98.31
	2	0.00	5.10	70.76	23.44	0.70	0.00	94.20
	3	0.00	8.49	10.07	74.97	6.42	0.04	81.39
	4	0.00	17.49	3.02	39.64	39.18	0.67	78.82
	5	0.00	10.15	1.29	32.94	42.29	13.33	88.56
							average all	88.26

IV. RESULTS AND DISCUSSION

The results generated by the experiments can be seen in Tables III, IV, and V, respectively for the CQ relations 32-20, 43-32, and 55-43. The columns of the tables correspond to the data coming from the encoding with the lower CQ and the rows, in turn, correspond to the higher CQ. The established relationship (cell) refers to the probability of choosing a depth when encoding the lower CQ representation given that a certain depth is observed in the higher CQ representation.

For example, in Table III (CQ relation 32-20), we see that a frame region encoded with depth 2 in the CQ 32 representation (line 2) will also be encoded with depth 2 in the CQ 20 representation (column 2) in 61.43% of the cases. Similarly, Table V shows that this number grows to 74.06% in the relationship 55-43.

In the tables, the columns with the highest probability of occurrence were highlighted for each line of depth observed. In this way, it is possible to identify that the highest probability of some level of depth occurring for any relationship tends to be the same depth observed at the lower CQ encoding. The exceptions are for depths 4 and 5, which are most likely to occur at the previous depth level (i.e., depths 3 and 4, respectively). The hypothesis for this exception is the use of better choices of partitioning types and/or predictive modes would be made in a lower CQ. It is also observed the absence of probabilities at depth 0. The hypothesis in this case is that the videos chosen for the analysis were not encoded with depth 0 in any case.

If the depth level observed during coding was reused at a lower quantization level, forcing the same depth level of the tree to be chosen, an approximate average assertiveness of 47.77%, 54.18% and 59.03% would be obtained, respectively, for the relations 32-20, 43-32 and 55-43. That is, the proba-

TABLE V
STATISTICAL ANALYSIS ON THE CQ RELATION 55-43, IN PERCENTAGE

		depth on CQ 43						
		0	**1**	**2**	**3**	**4**	**5**	**rule sum**
depth on CQ 55	**0**	0.00	0.00	0.00	0.00	0.00	0.00	–
	1	0.00	81.76	17.27	0.95	0.02	0.00	99.03
	2	0.00	5.23	74.06	20.19	0.51	0.01	94.25
	3	0.00	8.45	10.39	74.72	6.33	0.11	81.05
	4	0.00	11.00	0.94	44.35	42.58	1.13	86.93
	5	0.00	5.05	1.38	24.31	47.25	22.02	93.58
							average all	90.97

bility of getting the partitioning level right is low, bordering on chance. In this way, a rule is suggested to increase the probability of success of this information reuse: if the observed depth is less than or equal to 3, it allows the encoder to apply the same depth level or one more level. Otherwise (depths 4 and 5), it prevents the encoder from considering depth levels lower than 3. The option to choose a depth level more than what was observed in the first rule is caused by the perception that there is a tendency for the encoding with the lowest CQ to choose a partition under the current level, in order to obtain a better image quality. Furthermore, choosing the next depth level, to be applied in the first rule, is justified by the observation that, at depths up to level 3, it is observed that the second highest probability of occurrence is precisely the next level of the observed depth.

Based on this rule proposal, the "rule sum" column on the Tables III, IV, and V presents its probability of assertiveness. Notice that the proposed rule would decide correctly for the depth in 84.73%, 88.26%, and 90.97% of the cases, respectively, for the CQ relations 32-20, 43-32, and 55-43. This rule could be used in a future work as basis for a heuristic to reuse partitioning structures between different CQ representations, aiming at encoding acceleration. In other words, it will avoid that libaom spends time and energy processing depths that are statistically not recommended according to previous encodings of the same video content. In this way, considerable energy consumption savings are expected, especially in the context of video streaming service data centers.

V. CONCLUSIONS AND FUTURE WORK

The AV1 video encoder presents a high computational cost due to a large number of encoding tools and block partition types allowed. In the context of video streaming data centers, the encoding is performed several times for the same video, generating bitstream representations for various resolutions and qualities, aiming to provide compatibility with different types of clients and network conditions. Therefore, through a statistical analysis, this work identified opportunities to reuse block size decisions throughout the encoding under different quantization levels. More specifically, the analysis allowed identifying that it is possible to reduce the test of partitioning possibilities when encoding high-quality videos (low quantization) based on characteristics observed when encoding low-quality videos (high quantization). Even when avoiding certain coding tree depths according to the observed at higher quantization levels, there is still a possibility greater than 84% of getting a correct block partition. In future work, we intend to implement an heuristic approach based on the analysis carried out in this paper, in order to obtain the resulting compression efficiency (BD-Rate) and estimated energy consumption.

ACKNOWLEDGMENT

This study was financed in part by the *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior* – Brasil (CAPES) – Finance Code 001, Foundation for Research Support of the State of Rio Grande do Sul (FAPERGS), and National Council for Scientific and Technological Development (CNPq).

REFERENCES

[1] Motion Picture Association, "Theme report: A comprehensive analysis and survey of the theatrical and home/mobile entertainment market environment for 2020," mar 2020. [Online]. Available: https://www.motionpictures.org/wp-content/uploads/2021/03/MPA-2020-THEME-Report.pdf

[2] A. Kempf, M. Silva, S. Onesseken, and B. Garcia, "Consumo das plataformas de streaming antes e durante a pandemia da covid-19," 2021. [Online]. Available: https://admpg.com.br/2021/anais/arquivos/09122021_230937_613eb539011b0.pdf

[3] I. Ahmad, X. Wei, Y. Sun, and Y.-Q. Zhang, "Video transcoding: an overview of various techniques and research issues," *IEEE Transactions on Multimedia*, vol. 7, no. 5, pp. 793–804, 2005, doi: 10.1109/TMM.2005.854472.

[4] J. Ozer, "Hevc advance cuts content fees on streaming," mar 2018. [Online]. Available: https://www.streamingmedia.com/Articles/News/Online-Video-News/HEVC-Advance-Cuts-Content-Fees-on-Streaming-123828.aspx

[5] J. Han, B. Li, D. Mukherjee, C. H. Chiang, A. Grange, C. Chen, H. Su, S. Parker, S. Deng, U. Joshi, Y. Chen, Y. Wang, P. Wilkins, Y. Xu, and J. Bankoski, "A technical overview of av1," *Proceedings of the IEEE*, pp. 1–28, 2021, doi: 10.1109/JPROC.2021.3058584.

[6] A. Francis, "Top video technology trends 2022: The future of streaming is about device reach," 2020. [Online]. Available: https://bitmovin.com/top-video-technology-trends/

[7] I. E. Richardson, *Video codec design: developing image and video compression systems*. John Wiley & Sons, 2002.

[8] M. Ombura Jr., "How youtube handles streaming 4,000,000,000+ daily videos without a hitch," 2019. [Online]. Available: https://medium.com/@martinomburajr/how-youtube-handles-streaming-4-000-000-000-daily-videos-without-a-hitch-8542741e957a

[9] Netflix Technology Blog, "Per-title encode optimization," dec 2015. [Online]. Available: https://netflixtechblog.com/per-title-encode-optimization-7e99442b62a2