PARALLEL PERFORMANCE AND ENERGY EFFICIENCY OF MODERN VIDEO ENCODERS ON MULTITHREADED ARCHITECTURES

R. Rodríguez-Sánchez[1], F. D. Igual[2], J. L. Martínez[3], R. Mayo[1], E. S. Quintana-Ortí[1]

[1]Depto. de Ingeniería y Ciencia de Computadores, Universidad Jaume I (UJI), Castellón, Spain
e-mails: {rarodrig,mayo,quintana}@uji.es
[2] Depto. de Arquitectura de Computadores y Automática, Universidad Complutense de Madrid, Madrid, Spain e-mail: figual@pdi.ucm.es
[3]Albacete Research Institute of Informatics, University of Castilla-La Mancha, Albacete, Spain
e-mail: joseluis.martinez@uclm.es

ABSTRACT

In this paper we evaluate four mainstream video encoders: H.264/MPEG-4 Advanced Video Coding, Google's VP8, High Efficiency Video Coding, and Google's VP9, studying conventional figures-of-merit such as performance in terms of encoded frames per second, and encoding efficiency in both PSNR and bit-rate of the encoded video sequences. Additionally, two platforms equipped with a large number of cores, representative of current multicore architectures for high-end servers, and equipped with a wattmeter allow us to assess the quality of these video encoders in terms of parallel scalability and energy consumption, which is well-founded given the significant levels of thread concurrency and the impact of the power wall in todays' multicore processors.

***Index Terms*—** Video encoding, high performance computing, energy consumption, multicore processors

1. INTRODUCTION

Video coding is nowadays the key technology spurring a wide range of applications, most of them built on top of the "old" H.264/MPEG-4 Advanced Video Coding (AVC) standard [1], established in 2003 by the Joint Collaborative Team on Video Coding (JCT-VC). More recently, in 2010, Google Inc. acquired the VP8 codec [2], promoting this video compression format as a royalty-free alternative to H.264/AVC.

In the past few years, the demand for higher quality and video resolution has steadily increased, yielding a breathtaking increment of multimedia traffic in general, and digital video in particular. In this scenario, the JCT-VC and Google have responded to the urge for higher video compression performance by working on the next generation of video codecs, which has resulted in the introduction of the High Efficiency Video Coding (HEVC) standard [3], finalized in January 2013, and the VP9 codec [4], released in June that same year. Both codecs significantly depart from their predecessors, but improve the encoding efficiency by introducing new tools or by improving the existing ones.

Previous comparative analyses of video encoders mainly focus on execution time and encoding efficiency. However, the considerable levels of hardware parallelism in current multicore servers, combined with the support for significant thread-level concurrency in these architectures, clearly asks for a detailed evaluation of the parallelism of modern video encoders. In particular, the number of threads/cores employed in video encoders in general affects the execution time, but may also influence the encoding efficiency more subtlety, as some encoding dependencies can be broken in a parallel execution. Moreover, energy is now recognized as the crucial factor that will constrain the design of future computer architectures [5] and, therefore, this performance metric should be also taken into account when analyzing video encoders.

This paper assesses the efficiency of recent implementations of the aforementioned modern video encoders along four dimensions, namely execution time, encoding efficiency, parallelism, and energy consumption. Following the trend towards the integration of large core numbers in current High Performance Computing (HPC) servers, the analysis is performed on two platforms, with processors from Intel and AMD, and representative of current high-end multicore technology.

The rest of the article is organized as follows. In Section 2 we briefly discuss related work and in Section 3 we concisely review the background in video codecs. The major contribution of this paper is in Section 4, where we present the evaluation methodology jointly with the experimental results. Finally, a number of conclusions are outlined in Section 5.

2. RELATED WORK

There exist some recent work comparing different video codecs, most of them limited to two video codecs and an analysis of execution time and encoding efficiency. In 2011,

Feller et al. [6] present an overview of Google's VP8 codec, comparing this technology against the H.264/AVC standard. The VP8 implementation leveraged in their work was in an early development stage while the H.264/AVC implementation, on the other hand, corresponded to the highly optimized x264 implementation [7]. This explains that VP8 was reported to be up to 350% slower and, furthermore, offered considerable worse encoding efficiency (up to 1.7dB) than the x264 encoder. Also in 2011, Bankoski et al. [8] presented an VP8 overview, but in this case the evaluation was made in terms of decoding speed, using the FFmpeg decoder in libavcodec [9] with support for VP8 and H.264/AVC decoding. Their results show that the decoding speed of VP8 is faster, around 30%, than that of H.264/AVC.

Two more recent comparative analyses between VP8 and the x264 implementation of H.264/AVC, from 2012, were presented in [10, 11], where a more optimized version of VP8 was used. In these cases, the performance gap between the VP8 and x264 encoders is reported as considerably more reduced, to the point where the encoding speed of both encoders is shown to be similar. Additionally, in [11] the VP8 encoder is evaluated against the scalable extension of H.264/AVC (H.264/SVC).

In 2013, Bankoski et al. [12] presented an overview of the late VP9 encoder, including a comparison against implementations of the H.264/AVC and HEVC standards. The results show that the VP9 encoder attains better coding efficiency than both the x264 implementation of H.264/AVC and the reference implementation of HEVC, when using the *veryslow* preset and the *low delay Main profile* configurations, respectively. However, more favorable HEVC encoding efficiency is expected when using the *random access Main profile* configuration as shown by Grois et al. in [13].

Compared with our work, only [6] and [10] consider the use of multiple cores in their studies, with an old Intel Core 2 Duo (only 2 cores) in the former and a quad core Intel i7-2600k in the latter. However, the authors of [10] employ a relatively "old" version of VP8, significantly less optimized than that leveraged in our analysis. Furthermore, none of these past works considers the effect that thread-level concurrency has on encoding efficiency nor they evaluate the codecs in an sceneario with a large number of cores, usual in HPC servers.

3. BACKGROUND

Current video encoders employ a hybrid block-based compression technique which is able to eliminate temporal (inter-prediction) and spatial (intra-prediction) redundancies in video sequences. Additionally, by means of a transform module, the output of the aforementioned predictions is converted into a different domain and is quantified to remove irrelevant information. Finally, an entropy encoder is applied to the quantified data to remove statistical redundancies. Internally, there is a complete decoder within the encoder since the inter-prediction is carried out using reconstructed frames which have been previously encoded. The decoder includes an entropy decoder, inverse quantization and transform, and some filters to improve the quality of the reconstructed frames.

To conclude this short overview, the block size on which the predictions are carried out, the concrete kind of predictions performed, the concrete kind of transforms applied to the predicted data or which filters are used, depend on the specifications of each video encoder.

4. RESULTS AND COMPARISONS

4.1. Methodology and Setup

In this section we compare four recent video encoders from the perspective of *i)* execution time (in encoded frames per second, or FPS); *ii)* parallelism (i.e., ability to increase the FPS rate proportionally to the growth in the number of cores); *iii)* encoding efficiency (in terms of both Peak Signal-Noise Ratio, or PSNR, and bit-rate of the encoded sequences); and *iv)* energy consumption (in KJoules).

The target video encoders are the x264 (v0.135.x) [7] implementation of the H.264/AVC standard, the HM (v12.0) [14] reference code of the HEVC standard, and the VP8 and VP9 (v1.2.0-4256) implementations derived from the master branch of the WebM project [15]. We can expect that the x265 project [16] will provide a fully optimized implementation of the HEVC standard in the near future (as was done with x264). Unfortunately, at present the x265 encoder does not fully implement all the new mechanisms/tools defined by the HEVC standard, so we decided to evaluate instead the HM encoder. Note that the HM encoder does not include any code optimizations (other than those applied automatically by the compiler), because its purpose is to serve as a reference HEVC encoder. The VP9 encoder is in an early development stage and, like the HM, it is not optimized for speed.

Setting the ground for a fair comparison of codecs is a delicate task. To deal with this difficulty, the experiments were conducted following common test conditions and software reference configurations recommended by the JCT-VC [17]. Concretely, for each video sequence (see the complete list in the first column of Table 1), the Rate Distortion (RD) curves were built using four points, and the comparisons were made using these execution points. For that purpose, the HM encoder was run to encode the five B class sequences (1080p format) using four quantization parameters (namely, QP=22, 27, 32 and 37) for a total of 20 experiments. The other encoders were run in 2-pass mode to achieve a PSNR as high as possible for the bit-rate delivered by the HM encoder for each QP value for each sequence. The HM encoding parameters used for the evaluation were those included in the main profile (random access configuration) of the mentioned reference software. The x264 encoder was run using the *–preset veryslow* configuration and the *–tune psnr* option, in order to

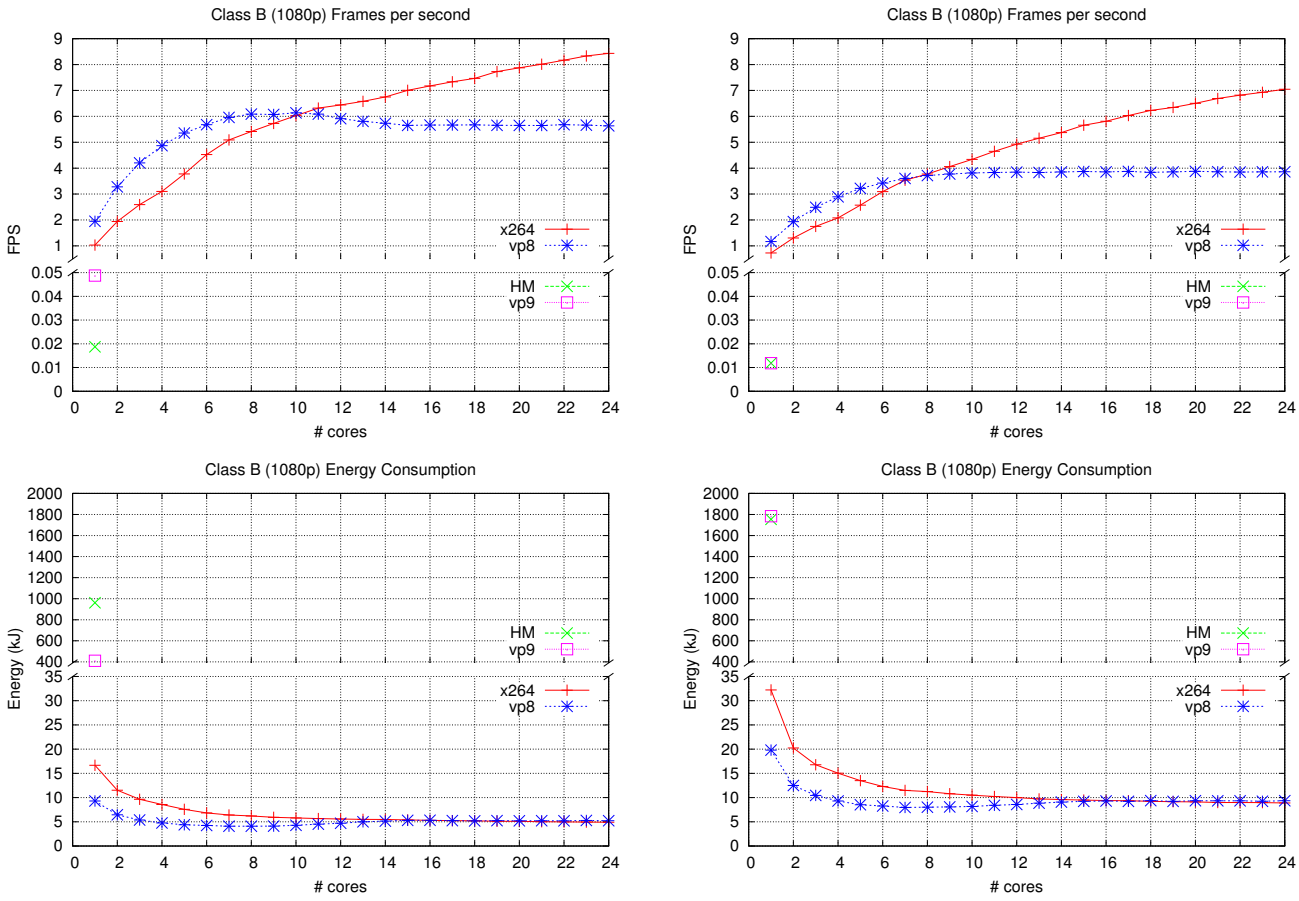


Fig. 1. Execution time (top) and energy consumption (bottom) of the video codecs on the Intel (left) and AMD (right) servers.

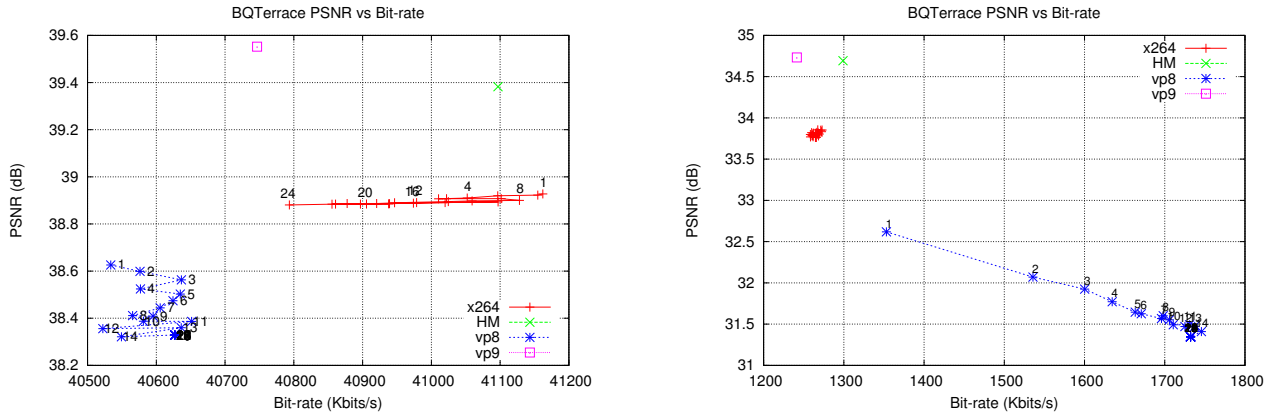
maximize the PSNR results. Finally, the VP8 and VP9 encoders were run using the *2-Pass Faster VBR Encoding* configuration, which is available in the webM website [15], except that the *–cpu-used* parameter was set to 0, and the option *–tune=psnr* was used to maximize the PSNR results. Additionally, the encoders with multithread support (x264 and VP8) were tested for any number of CPU cores, from 1 to the maximum available in the target platform.

The evaluation was carried out on two servers, equipped with Intel and AMD multicore technology. The Intel platform is composed of four Intel Xeon E5-2620 processors (with 6 cores each), running at 2.00 GHz, and 32 GB of DDR3 memory. The AMD platform comprises two AMD Opteron 6172 processors (with 12 cores each), at 2.10 GHz, with 248 GB of DDR3 memory. The power dissipation was measured using an APC 8653 PDU (Power Distribution Unit) which samples power once per second. This device is directly attached to the cable that connects the electric socket to the computer power supply unit. A daemon application ran on a separate tracing server, collecting power samples from the PDU. The samples were then averaged and the result multiplied by the execution time to obtain the energy consumption.

4.2. Experimental results

Let us consider the results in Figure 1. Each point in all four graphs there correspond to the average value of the 20 experiments for a concrete number of threads. The top two plots report the FPS delivered by the four encoders executed on both testing platforms, showing that the FPS rates attained by HM and VP9 are two orders of magnitude below those for x264 and VP8. This was actually expected, as the HM and VP9 implementations are not optimized for speed and, furthermore, currently they offer no support for multithreaded encoding. More interestingly, this experiment illustrates that the VP9 encoder is superior to the HM encoder, and the x264 encoder is faster than the VP8 encoder when more than 7–10 threads are employed, depending on the server.

Before analyzing the encoder's parallelism, we note that x264 leverages frame-level parallelism, while vp8 applies a strategy in which rows of macroblocks are simultaneously encoded by different threads. However, the vp8 entropy encoding module supports up to 8 token partitions, each of them executed by a different thread, restricting the parallelism of this solution. The experiment reveals that little or no benefit is gained from using more than 8–10 threads for the VP8



(a) HM - QP=22 (b) HM - QP=37

Fig. 2. PSNR vs bit-rate for the BQTerrace sequence.

codec, as the FPS rate does not increase or it can even decrease from there (e.g., in the Intel server). This may explain why, if VP8 is configured to launch more than 15 threads, internally the encoder only spawns 15. On the other hand, the x264 codec delivers a steady performance improvement as the number of cores is increased, demonstrating higher parallelism than VP8.

The bottom row in Figure 1 reports the energy consumption of the encoders. The first aspect to note is the high energy figures for the HM and VP9 encoders, two orders of magnitude superior to those reported for x264 and VP8, which is due to the linear dependence between the energy usage and the execution time. Focusing the analysis on the multi-threaded VP8 encoder, we can observe an increase of energy when more than 8–10 cores are employed, mostly explained by the null effect or even the increase in the execution time for these configurations. For the x264 codec, there are important energy savings to be gained by using up to 14 threads. However, the reduction of execution time as the number of threads grows from there is canceled by the increase in power dissipation, resulting in a negligible effect on the energy efficiency.

Due to space limitations, Figure 2 only includes the RD results for two concrete experiments, in particular those corresponding to the BQTerrace sequence encoded with the lowest and highest QP values in HM. The graphs mainly show that achieving the target bit-rate is not an easy task. Also, the use of multiple threads exerts a considerable effect on the RD results for VP8, but the impact is minor for x264. Specifically, when the number of threads is raised from 1 to 24, (independently of the platform,) the quality may decrease by up to 1.4 dB for VP8 and only by 0.1 dB for x264. This behavior is due to the specific strategy that each encoder applies to exploit concurrency, discussed previously.

Table 1 reports the RD performance in terms of the Bjøntegaard Delta Rate metric (BDRate) [17] with all the

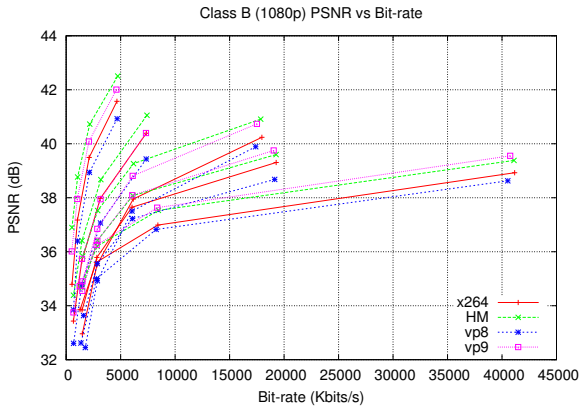


Fig. 3. RD results for class B (1080p).

codecs executed using a single thread. In these results, the VP8 RD curves are set as the reference, since this encoder offers the worst encoding efficiency (see Figure 3). The RD values for the remaining curves are normalized to this reference. Although the x264 encoder improves the RD results of the VP8 encoder in all cases, the HM and VP9 encoders provide even higher performance, depending on the video content. This behavior is further detailed in Figure 3. Specifically, when high bit-rates are required to encode a sequence the VP9 encoder renders more favorable PSNR values. On the other hand, when low bit-rates are necessary the HM encoder delivers superior PSNR values.

5. CONCLUSIONS

We have presented a detailed comparison of four encoders that represent the state-of-the-art for video compression, from the points of view of execution time, parallelism, energy consumption, and encoding efficiency, on two top-of-the-shelf HPC multicore servers.

Sequence	x264			HM			VP9		
	Y (%)	U (%)	V (%)	Y (%)	U (%)	V (%)	Y (%)	U (%)	V (%)
BasketballDrive	-12.6	-17.4	-17.0	**-51.4**	**-55.9**	**-54.2**	-39.6	-47.7	-44.5
BQTerrace	-24.4	-12.3	-12.1	-48.1	-32.2	-33.7	**-50.1**	**-45.0**	**-45.9**
Cactus	-19.4	-19.7	-1.5	**-36.1**	-34.5	-27.8	-35.2	**-37.5**	**-31.2**
Kimono	-20.4	-23.0	-12.2	**-50.6**	**-46.2**	**-43.1**	-39.2	-29.9	-26.8
ParkScene	-27.0	-22.3	-22.9	**-42.0**	**-46.2**	**-44.6**	-28.7	-25.8	-26.0
Average	-20.8	-14.0	-13.2	**-45.6**	**-43.0**	**-40.7**	-38.6	-37.2	-34.9

Table 1. BDRate for class B sequences normalized with respect to VP8.

The main conclusion that can be extracted from this study is that the "best" enconder strongly depends on the dimension (metric) of interest. Specifically, in terms of execution time (i.e., FPS), if the target is a conventional server equipped with 4–8 cores, the VP8 encoder offers the fastest technology. However, the most parallel codec is x264 which, in consequence, turns this solution into the fastest one when the number of cores is considerably large. The linear dependence between execution time and energy renders similar conclusions for the energy efficiency. In general, a decrease of execution time brings along an improvement in energy consumption, except when a large number of cores is employed for the x264 encoder, where we observed no effect on the energy. Finally, the recent VP9 and HM encoders are not optimized for performance, but these standards provide the best coding efficiency by a wide margin over x264, which in turn clearly outperforms VP8.

Acknowledgments

This work has been jointly supported by the Ministry of Science and Competitiveness and European Commission (FEDER funds) under the projects TIN2012-38341-C04-04 and TIN2011-23283, and project P1-1B2013-20 of the Fundació Caixa Castelló-Bancaixa and UJI.

REFERENCES

[1] ITU–T, "ITU–T Recomendacin H.264, Advanced Video Coding for Generic Audiovisual Services," 2003.

[2] J. Bankoski, P. Wilkins, and Y. Xu, "VP8 Data Format and Decoding Guide," 2011, http://datatracker.ietf.org/doc/.

[3] ITU–T and ISO/IEC, "High efficiency video coding (HEVC) text specification draft 10," January 2013.

[4] A. Grange and H. Alvestrand, "A VP9 Bitstream Overview," 2013, http://datatracker.ietf.org/doc/.

[5] M. Duranton and *et al*, "The HiPEAC vision for advanced computing in horizon 2020," 2013.

[6] C. Feller, J. Wuenschmann, T. Roll, and A. Rothermel, "The VP8 video codec - overview and comparison to H.264/AVC," in *2011 IEEE Int. Conf. Consumer Electronics (ICCE-Berlin'11)*, 2011, pp. 57–61.

[7] VideoLAN Organization, "x264 project," 2013, http://www.videolan.org/developers/x264.html.

[8] J. Bankoski, P. Wilkins, and Y. Xu, "Technical overview of VP8, an open source video codec for the web," in *2011 IEEE International Conference on Multimedia and Expo (ICME '11)*, 2011, pp. 1–6.

[9] FFmpeg, "The FFmpeg project," 2013, http://www.ffmpeg.org/.

[10] Y.O. Sharrab and N.J. Sarhan, "Detailed Comparative Analysis of VP8 and H.264," in *2012 IEEE International Symposium on Multimedia (ISM '12)*, 2012, pp. 133–140.

[11] Y. Yoon, M. Kim, S. Lee, B. Lee, S.-J. Hyun, and K. Lee, "Performance analysis of H.264/AVC, H.264/SVC, and VP8 over IEEE 802.11 wireless networks," in *2012 IEEE Symposium on Computers and Communications (ISCC '12)*, 2012, pp. 151–156.

[12] J. Bankoski, R. S. Bultje, A. Grange, Q. Gu, J. Han, J. Koleszar, D. Mukherjee, P. Wilkins, and Y. Xu, "Towards a next generation open-source video codec," in *Proc. of SPIE, Visual Information Processing and Communication IV*, 2013, vol. 8666, pp. 1–13.

[13] D. Grois, D. Marpe, A. Mulayoff, B. Itzhaky, and O. Hadar, "Performance comparison of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC encoders," in *Picture Coding Symposium (PCS), 2013*, Dec 2013, pp. 394–397.

[14] Joint Collaborative Team on Video Coding (JCT-VC), "HEVC Test Model," 2013, http://hevc.hhi.fraunhofer.de/.

[15] The WebM Project, "WebM: an open web media project," 2013, http://www.webmproject.org/.

[16] The MultiCoreWare team, "x265 project," 2013, http://x265.org/.

[17] Joint Collaborative Team on Video Coding (JCT-VC), "Common test conditions and software reference configurations," Join collaborative Team on video Coding 9th meeting, Doc. JCTVC-K1100, October 2012.