www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 6 Issue 11 November 2017, Page No. 23224-23228
Index Copernicus value (2015): 58.10 DOI: 10.18535/ijecs/v6i11.24

Data-Level Parallel Approaches for the H.264/AVC: A Review from the Encoder and the Decoder Perspectives

Mohammed Faiz Aboalmaaly

Computer Techniques Engineering, Alsafwa University College,
Alma'amlachi, Karbala, Iraq

Abstract:
In order to enable easier transmission and storage of videos, video-coding techniques are used as data compression process that is intended to reduce the size of raw video without sacrificing its visual quality. The H.264 is relatively one of the recent video compression standards, which has proved to outperform former standards in terms of compression efficiency. However, it's associated with mush higher computational complexity. Several software-based as well as hardware-based approaches have been suggested to tackle this problem by using several flavours of data-level parallel approaches for the encoder and decoder sides. In this paper, these approaches are presented and compared in form of a comparative review. The suitability of one particular approach is determined based to the architecture used.

Keywords: video coding, distributed shared memory architectures, encoding latency, parallel scalability, GPU.

1. Introduction

Video compression is a process intending at reducing the size of raw video without sacrificing the visual quality of video in order to enable easier transmission and storage of videos [1]. Video compression is a process that requires the existence of two complementary systems; the encoder and the decoder. Prediction, transformation and quantization, and entropy coding are the common techniques in video compression algorithms. The encoder system carries on the processes above, while the decoder system involves the same processes in reverse order [2]. With regard to the emergence of the HEVC standard, the H.264 is still relatively considered as one of the recent video compression standards, which has proved to outperform former standards in terms of compression efficiency. However, What makes the H.264 more resource-intensive when compared to previous video compression solutions is the added new features that are intended to further increase the compression efficiency while keeping the visual quality saturated [3]. In a comparison with former standards, the introduction of the new features has noticeably improved the compression efficiency. Consequently, the computational complexity of the H.264 video coding standard has increased by a factor of ten for the encoder and about 2 to 4 times better for the decoder side [4]. To mitigate the drawbacks of its higher complexity, parallelism is adopted to lessen the encoding or decoding time of the H.264 codec.

In a view of the H.264 codec, it has been proved that the data-level parallelization has outperformed the task-level parallelization [5] due to the several kinds of dependencies among the coding components of the H.264 codec. This paper presents reviews some of the attempts aimed at lessening the complexity of the H.264 vide coding standard based on data-level parallelism utilizing different parallel architectures. The rest of this paper is arranged as follows: Section 2 gives the necessary background covering some basic terminology; section 3 covers the different parallel granularities of the H.264 and finally the conclusions are given in the last section.

2. Background

Both the encoder and the decoder must handle the processes of prediction, transformation and quantization, and entropy coding. Predication is a technique in which information of a given pixels in a video frame can be predicted from neighbors pixels in the same frame (intra-prediction) or from a corresponding position of previously compressed frames (inter-prediction). This technique depends of the correlation of the neighbors' pixels, which is normally high in videos. Transformation is the process that transforms the spatial domain to an equivalent frequency domain. The purpose of transformation in video coding is to make video signals are amenable to be compressed. The quantization process is achieved by divides the transform coefficients resulted from the transformation process by an integer value (quantization parameter) to achieve the targeted bitrate. Finally, entropy coding is purposely utilized to explore the statistical redundancy of videos in order to further reduce their size [1].

Approaches targeting lessening the computational complexity of the H.264 are numerous. These approaches differ in terms of the orientation used. In this aspect, hardware-oriented along with software-oriented approaches have been examined in pervious works. Example to the former is achieved by utilizing a special hardware as an accelerator. The application specific integrated circuit (ASIC) and the field programmable gate array (FPGA) are accelerators' examples to this hardware-oriented category. Hardware-oriented methods have shown to achieve good performance efficiency. However, there main problem is the difficulty in reconfiguration and reprogramming [6]. Further, this approach takes longer time to market when compared to the software-oriented approaches. In the software-oriented approach, lessening the complexity of H.264 is attained by the aid of complexity reduction as well as parallel computing. In the

complexity reduction approach, reducing the complexity is done by removing some of the features of the video compression algorithm of the H.264 that are subjectively deemed as redundant. Meanwhile, parallel video compression is achieved when the compression components are decomposed with regard to the task, data or a combination of them among several available computing resources. However, because of its scope of standard, where the bit stream and the decoding processes are tightly defined, opportunities of applying complexity reduction or parallel computing on the decoding side would be limited in comparison with the encoding side of the H.264 standard.

3. Parallel Granularities of the H.264 Standard

Before exploring the data-levels parallel attempts of the H.264 standard, it is essential to know how these levels are defined. In fact, in terms of parallelism, decomposing data into smaller parts is possible if no or weak dependency among these parts could be identified. Thus, group of pictures (GOP), frames, slices, macroblock (MB), and blocks are the five possible granularities, which are normally exploited, in the H.264 video coding standard. Figure 1 shows their relative size with regard to one video sequence.

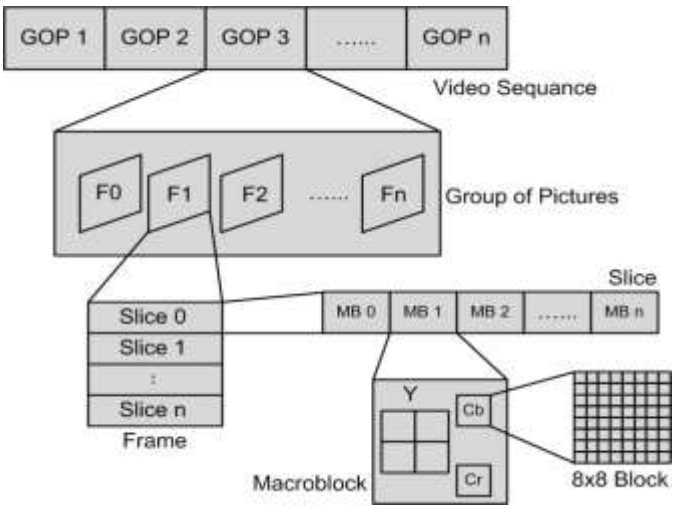

Figure 1: Granularities of the H.264 Standard

GOPs are used for synchronization purposes because there are no temporal dependencies among them. Each GOP is composed of a set of frames. These frames are possibly having temporal dependencies based on their types due to the motion prediction among frames. Each frame is further divided up into one or more slices. The slice is a standalone unit for encoding and decoding and there are no spatial dependencies between slices. Moreover, each slice is further composed of a set of MBs. MBs are the basic units of prediction. H.264 allows variable sizes of each MB. Additionally, MBs are composed of few blocks wherein each block is composed of picture samples, and these pictures samples can be processed in parallel.

3.1 GOP-level

Few studies have investigated the adoption of this level of parallelism for the H.264 encoder. For instance, a hierarchical parallelisation approach for H.264 encoder is introduced in [7]. In the hierarchy, shown in Figure 2.14, a GOP-level parallelism and a slice-level parallelism are combined together to overcome the latency problem of using a GOP-level only. Using MPI and multithreaded parallelism, the implementation parallelises H.264 encoder on a cluster machine. Synchronisation was the main problem that produces a loss in the encoding speedup, which is believed to be due to the double layers of parallelism that introduces several points of barriers [8].

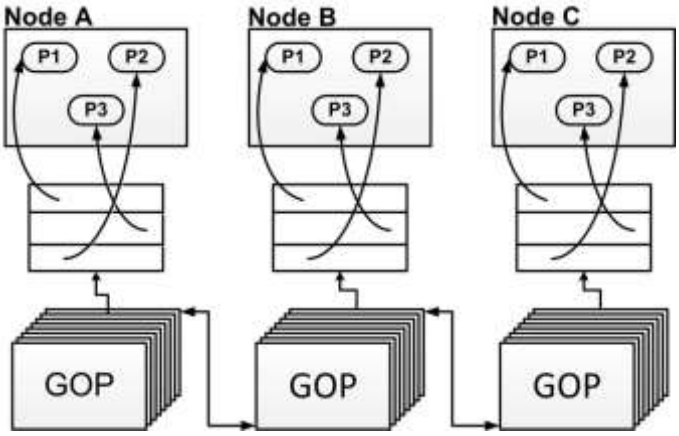

Figure 2: Hierarchical H.264 Parallel Encoder

Further, a GOP-level parallelization for the H.264 decoder is proposed in [9]. Different sizes of GOP are examined (4, 20, and 40) on a shared memory clustering machine equipped with 40 cores. Regardless the size of a GOP, linear speedup is achieved with up to 11 cores. However, when new processing elements are added, a saturated speedup was noticed for the 20 and 40 GOP sizes, while a sub-linear speedup is achieved for the parallel decoder with 4 frames per GOP. These outcomes show that the size of GOP has a direct impact on the parallel speedup. In fact, the memory bottleneck is remaining with coarse-grained parallelism even using a state of the art computing processors.

However, because GOP-level is the coarsest level for the H.264 codec, the invention at this level of parallelism is limited and that explain the scarce number of studies related to this level. The standard has already specified that each GOP is a coding independent unit of both spatial as well as temporal dependencies. The parameters that would be amenable to modify are the number of frames in each GOP [10] and the frames' type sequence. In addition, the way each GOP processed in parallel would vary and such variation can be motivated by the hardware architecture used.

3.2 Frame-level

Frame-level parallelism is achieved by the simultaneous coding of independent frames in one GOP. Few numbers of frames can be coded in parallel. This limitation is imposed by the existence of B frames (prediction can be from previous and next encoded frames). Hence, frame-level parallelism would be more promising at the H.264 baseline profile, since baseline profile does not support B frames. Furthermore, this level dose not incurs any increase in the bit rate or degradation in the video quality.

The frame-parallel encoding scheme based on encoding picture frames that share no data dependency is proposed in [11]. Up to three concurrent encoding frames only can be reached due to the dependency among frames. However, a reduction of 66% is achieved of the system bandwidth and no time measurements were shown.

3.3 Slice-level

Independently, numerous studies have relied on this level of parallelism. As previously mentioned, this level maps well on the shared memory architectures. For instance, an adaptive slice control scheme is proposed by [12] to parallelize the H.264 encoder. The encoder decides the number of slices before encoding each time at per-frame bases. Using a four-core machine, a speedup of 3.03x in the encoding speed is achieved over the serial implementation. The proposed solution relies on the fact that the encoding complexity over some parts of the frame (motion) is significantly different from other parts (low motion). While when each frame shows normality in the complexity of encoding among slices, the solution will not show any speedup gain, and the proposed solution will not be more than an extra overhead in deciding the number of slices that will lead to an extra encoding time.

At the same level of this parallel unit (slice), the study conducted by [13] proposes an implementation for the parallel algorithms of H.264 encoder based on Intel CPU with hyper-threading architecture. The idea is to split a frame into several slices, which are processed by multiple threads, resulting speedups ranging from 3.1x to 3.7x on a system of 4 Intel Xeon processors with Hyper-Threading disabled.

Additionally, a strip-wise parallel approach is proposed in [14]. The idea is based on statically decomposing the entire video frame into strips. Each strip may contain one or more slices. Each strip is encoded by one processer. These strips are overlapped to guarantee that no break in data dependency will occur at the strip boundaries. However, data are still required to be transferred (synchronization) from one processer to another. In particular, the neighbor data is 16 pixels above the top of the strip and 16 pixels below the bottom of the strip. Test results are compared with the JM reference software [15]. The hardware platform was equipped with two processers each with four cores. The speedup achieved was up to six times more in comparison with the serial implementation of the video encoder. A decrease in the parallel efficiency was noticed due to the increase in the data exchange.

Similar to the work proposed in [12], an adaptive slice size control is proposed in [16]. The idea behind this scheduling technique is using an MB mode selection a pre-processing step in order to determine the size of slice. This step is suggested to provide a uniformity of the per core encoding workload. Simulation results based on the JM software reference shows up to 57.30% reduction in the encoding time over the fixed-size slice-level approach.

Decoding using slice-level parallelism has been presented in [17]. The idea of the work was based on applying decoding time prediction at the encoder stage. This requires the decoder to inform the encoder about the time required for each slice to decode within the frame so the encoder can accordingly adjust the size of each slice to approximate the decoding time of slices. Speedup has been achieved compared to the parallel uniform slice approach. However, essentially, this approach requires that the encoder and the decoder are presented which apart from the online coding is not possible.

In terms of parallel H.264, slice-level is a trade-off method and it is also the most universal parallelization method employed to parallelize H.264 codec [12].

3.4 MB and block-level

Place At a finer level, in [18], an MB region partitioning is proposed to explore the parallelisation at the MB-level. A one-dimensional (vertical) partitioning is suggested to the frame and maps each partition to different processors, as shown in Figure 3. Then, a wavefront technique, shown in Figure 4, is used for each partition for encoding. However, in order to avoid data dependency, processors start to encode data after a short time one by one, and during the time a processor could encode a row of MBs in a MB region and transfer required reconstructed data to the next adjoining processor, which will propagate a synchronisation overhead. This synchronisation will become more annoying if the workload of each MB region is significantly unequal. Simulation results of four processors show a speedup up to 3.33 compared to the sequential reference encoder JM 10.2 using a CIF (352 × 288) video sequence.

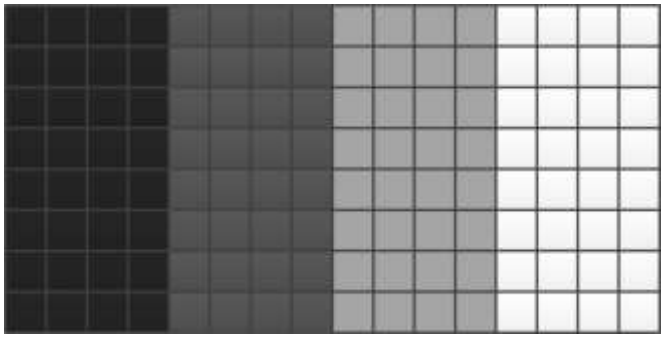

Figure 3: MB Region Partitioning of a Frame

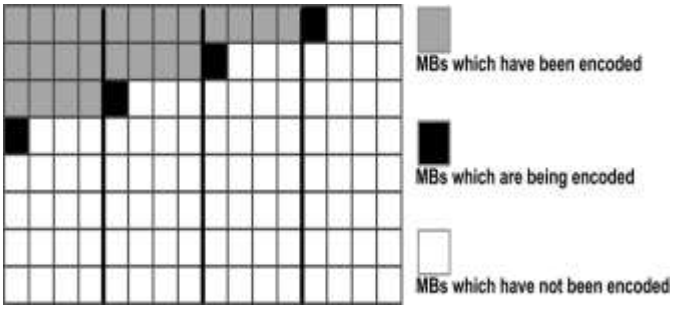

Figure 4: MB-Level Parallelism using Wavefront Method

The work in [19] proposed a parallel algorithm with a wavefront-based technique on the analysis of data dependencies in the H.264 baseline encoder. Data were mapped onto different processors at the granularity of frames or MB rows, and the final speedups were up to 3.17 on a software simulator with four processors. This method of data partitioning with the wavefront technique avoids damage on the compression ratio by splitting frames into slices or vertical partitions. However, the way how the motion estimation is treated across the vertical partitions with previously encoded frames is not presented. In fact, this treatment is essential if the damage on the compression ratio was not allowed. Thus, missing information has been identified.

Moreover, in several studies such as [20-22], the graphical processing units (GPUs) have been utilised to parallelise part of the encoding stages. In particular, motion estimation is usually ported to be run by using GPUs at the MB-level. However, in spite of the speedup achieved by these studies, for instance, in [20], a 20% speedup has been achieved over the serial implementation the H.264 encoder. The drawback of

memory transfer between the CPU and the GPU is still challenging.

A data parallelism at the MB-level for the H.264 decoder processes is proposed in [23]. By using the Cell BE as the target hardware, dependencies between Intra-coded MBs are addressed by partitioning each video frame into rows of MBs and assigning one full row of MBs to each secondary processing unit. This partition is implemented once the entropy decoding stage for each frame is serially decoded using the primary processing unit. However, several synchronisation barriers are added to cope with the standard decoder as well as the memory model of the Cell BE. Results indicate a better decoding time. However, the speedup achieved is not promising for a processer with nine processing nodes such as the Cell BE.

In [24], a dynamic load balancing approach is proposed for the decoding processes of the H.264. Based on the separation of the decoding modules into entropy decoding, inverse quantization and transformation, prediction, and deblocking filter, the load balancing is achieved when each of the processers is exciting one or more of the modules. This separation is considered with regard to the MB dependency within each frame. Parallel implementation is made using the POSIX multithreading model on a dual core machine. Results indicate a speedup up to 1.74 in comparison to the serial implementation.

Finally, in several works, the block-level H.264 parallelization was mentioned as a possible data-level parallelization approach, however, we fail to identify specific studies that have dedicatedly adopt this level of parallelism.

4. Comparison of the H.264 Data-Level Parallelisms

Based on the revision made in this paper, few remarks have been identified. Firstly, GOPs are a coding-independent unit. Therefore, the GOP level is easy to implement; however, it has long latency [25] and large memory requirements [5]. Thus, paralleling the GOP level is inappropriate for shared memory architecture because of limited on-chip memory [25]. Secondly, frame-level coding does not increase bit rate; however, complex interdependencies, which are caused by very flexible usage of reference pictures, limit its parallel scalability [13, 16, 26]. Moreover, this level of coding is associated with large memory requirements. Thirdly, slice-level coding has been associated with minimal synchronization cost, normal memory requirements, and good performance scalability [5]. The only drawbacks associated with this level are the increasing bit rate and degradation of visual quality when the number of slices increases [13]. Fourthly, MB-level and block-level coding incur no bit rate degradation; nevertheless, both are associated with high synchronization costs because of the small-sized parallel unit, dependency among them [17], and poor scalability [5], which render them incompatible with the current trend of multicore.

Given this remarks, parallel granularity in video coding could potentially reflect the performance of a parallel system in terms of scalability, synchronization cost, and memory requirements.

5. Conclusion and Future Directions

As the H.264 is associated with high computational complexity, several works has attempted to reduce its complexity. In this paper, a review to the data-level parallelisms for the H.264/AVC codec is presented. In particular, five levels of parallelism were identified. However, few levels inspired from the original five levels were included as additional previous works. As a conclusion, the type of parallel architecture is strongly determining the level of parallelism where the distributed memory is more suitable for the coarse-parallelism and the shared memory is more suitable for fine-parallelism. Further, following to most of the parallel algorithms, the parallel efficiency was limited when the synchronizations are used frequently. Thus, in order to improve the parallel performance, it is highly recommend avoiding the utilization of synchronizations. As a future direction, exploring more variant parallel attempts for video compression would provide a more comprehensive review work.

References

1. Woods, J.W., Chapter 12 - Digital Video Compression, in Multidimensional Signal, Image, and Video Processing and Coding (Second Edition), J.W. Woods, Editor 2012, Academic Press: Boston. p. 467-528.
2. Waggoner, B., Chapter 14 - H.264, in Compression for Great Video and Audio (Second Edition), B. Waggoner, Editor 2010, Focal Press: Boston. p. 223-255.
3. Dhanani, S. and M. Parker, 15 - From MPEG to H.264 Video Compression, in Digital Video Processing for Engineers, S. Dhanani and M. Parker, Editors. 2013, Newnes: Oxford. p. 125-140.
4. Kwon, S.-k., A. Tamhankar, and K.R. Rao, Overview of H.264/MPEG-4 part 10. Journal of Visual Communication and Image Representation, 2006. 17(2): p. 186-216.
5. Jo, S., S.H. Jo, and Y.H. Song, Exploring parallelization techniques based on OpenMP in H.264/AVC encoder for embedded multi-core processor. Journal of Systems Architecture, 2012. 58(9): p. 339-353.
6. Choi, K. and E.S. Jang, Leveraging Parallel Computing in Modern Video Coding Standards. IEEE MultiMedia, 2012. 19(3): p. 7-11.
7. Rodriguez, A., A. Gonzalez, and M.P. Malumbres. Hierarchical Parallelization of an H.264/AVC Video Encoder. in PAR ELEC 2006. International Symposium on Parallel Computing in Electrical Engineering. 2006.
8. Li, J., et al., Analysis of factors affecting execution performance of openMP programs. Tsinghua Science and Technology, 2005. 10(3): p. 304-308.
9. Gurhanli, A., C.C.P. Chen, and H. Shih-Hao. GOP-level parallelization of the H.264 decoder without a start-code scanner. in 2nd International Conference on Signal Processing Systems (ICSPS). 2010.
10. Hsu-Feng, H. and W. Chen-Tsang, Balanced Parallel Scheduling for Video Encoding with Adaptive GOP Structure. IEEE Transactions on Parallel and Distributed Systems, 2013. 24(12): p. 2355-2364.
11. Yi-Hau, C., et al. Frame-parallel design strategy for high definition B-frame H.264/AVC encoder. in IEEE International Symposium on Circuits and Systems. 2008.
12. Lili, Z., et al. A dynamic slice control scheme for slice-parallel video encoding. in 19th IEEE International Conference on Image Processing (ICIP). 2012.

13. Yen-Kuang, C., et al. Towards efficient multi-level threading of H.264 encoder on Intel hyper-threading architectures. in Proceedings of the 18th International Parallel and Distributed Processing Symposium. 2004.
14. Gu, J. and Y. Sun, Optimizing a Parallel Video Encoder with Message Passing and a Shared Memory Architecture. Tsinghua Science & Technology, 2011. 16(4): p. 393-398.
15. JVT. H.264/MPEG-4 AVC JM Reference Software. 2009; Available from: http://iphome.hhi.de/suehring/tml/.
16. Jung, B. and B. Jeon, Adaptive slice-level parallelism for H.264/AVC encoding using pre macroblock mode selection. Journal of Visual Communication and Image Representation, 2008. 19(8): p. 558-572.
17. Roitzsch, M., Slice-balancing H.264 video encoding for improved scalability of multicore decoding, in Proceedings of the 7th ACM & IEEE international conference on Embedded software2007, ACM: Salzburg, Austria. p. 269-278.
18. Sun, S., D. Wang, and S. Chen, A Highly Efficient Parallel Algorithm for H.264 Encoder Based on Macro-Block Region Partition, in High Performance Computing and Communications, R. Perrott, et al., Editors. 2007, Springer Berlin Heidelberg. p. 577-585.
19. Zhuo, Z. and L. Ping. A Highly Efficient Parallel Algorithm for H.264 Video Encoder. in IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP 2006 Proceedings. 2006.
20. El-Shehaly, M.H., et al., Use of CUDA streams for block-based MPEG motion estimation on the GPU, in ACM SIGGRAPH 2012 Posters2012, ACM: Los Angeles, California. p. 1-1.
21. Takano, F. and T. Moriyoshi. GPU H.264 motion estimation with contiguous diagonal parallelization and fusion of macroblock processing. in Consumer Electronics (ICCE), 2013 IEEE International Conference on. 2013.
22. Youngsub, K., Y. Youngmin, and H. Soonhoi. An efficient parallel motion estimation algorithm and X264 parallelization in CUDA. in 2011 Conference on Design and Architectures for Signal and Image Processing (DASIP). 2011.
23. Baker, M.A., et al., A scalable parallel H.264 decoder on the cell broadband engine architecture, in Proceedings of the 7th IEEE/ACM international conference on Hardware/software codesign and system synthesis2009, ACM: Grenoble, France. p. 353-362.
24. Ding-Yun, C., et al. A novel parallel H.264 decoder using dynamic load balance on dual core embedded system. in 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2012.
25. Fernandez, J.C. and M.P. Malumbres, A Parallel Implementation of H.26L Video Encoder (Research Note), in Proceedings of the 8th International Euro-Par Conference on Parallel Processing2002, Springer-Verlag. p. 830-833.
26. Zrida, H.K., et al., High Level Optimized Parallel Specification of a H.264/AVC Video Encoder. International Journal of Computing & Information Sciences, 2011. 9(1): p. 34-46.

Author Profile

Mohammed Aboalmaaly received the B.Sc. degree in Software Engineering from Almansour University College in 2005, M.Sc. and PhD. in Computer Science from Universiti Sains Malaysia in 2009 and 2015 respectively. He is currently working as head of Computer Techniques engineering Department at Alsafwa University College. His research interest is mainly focusing on multimedia computing, parallel processing, with special interest on the Internet of things (IoT).