International Journal of Multimedia and Ubiquitous Engineering
Vol.10, No.4 (2015), pp.81-90
http://dx.doi.org/10.14257/ijmue.2015.10.4.09

Empirical Analysis of Video Partitioning Methods for Distributed HEVC Encoding*

Byoung-Dai Lee

Department of Computer Science, Kyonggi University, Suwon, Korea
blee@kgu.ac.kr

Abstract

As cloud computing has emerged as a promising technique in mainstream application domains, significant attention has been paid to distributed video encoding, in which resource-intensive encoding tasks are distributed across unlimited computational resources available in the cloud environment. For distributed video encoding, the input video must be partitioned into several segments. This approach decreases the total encoding time but may suffer from quality degradation associated with a lack of information, such as the coding complexity of the previous video segment. In this paper, two well-known video partitioning methods are explored from different performance perspectives, including encoding time, bitrates, and peak signal-to-noise ratio (PSNR).

Keywords: *Cloud Computing, Distributed Video Encoding, HEVC, MapReduce*

1. Introduction

The H.264/Moving Picture Experts Group (MPEG)-4 Advanced Video Coding (AVC) standard [1] is one of most widely used video coding standards for many applications. However, with the recent market trend toward large-sized display devices for Ultra High-Definition (UHD) video services, the increasing demands for higher-quality, higher-resolution digital video necessitates a new, high-performance with superior coding efficiency to that of H.264/MPEG-4 AVC. To meet such needs, the ITU-T Video Coding Experts Group (VCEG) and ISO/IEC MPEG have established the Joint Collaboration Team on Video Coding (JCT-VC) and developed the High Efficiency Video Coding (HEVC) standard [2]. On April, 2013, HEVC was approved as an ITU-T standard and is now available as a free download from the official ITU-T website.

The main advantage of HEVC is its coding efficiency. Several evaluations showed that HEVC requires only half the bitrate of H.264/MPEG-4 AVC at the same level of video quality [3][4]. As such, it can support 8K UHD and resolutions up to 8,192×4,312 with moderate bitrates. As with H.264/MPEG-4 AVC, HEVC also uses the block-based hybrid coding architecture that incorporates a motion estimation and compensation stage, a transform stage, and an entropy encoder. However, the use in HEVC of various block structures with flexible subpartitioning mechanism, such as Coding Unit (CU), Prediction Unit (PU), and Transform Unit (TU), improves coding efficiency, yet introduces significantly increased computational complexity and thus longer encoding times. For instance, according to [4], in many test cases, HEVC can achieve the same visual quality as H.264/MPEG-4 AVC at half the bitrate, but at the cost of 2–10 times increase in computational complexity. Similar experimental results [5] showed that up to 12 hours were needed to encode a single 10-second test case with the HEVC software codec known as HEVC Test Model (HM 8) [6].

* This paper is a revised and expanded version of a paper entitled "Cloud-based Distributed HEVC Encoding" presented at the 7th International Conference on Signal Processing, Image Processing and Pattern Recognition (SIP 2014) on Dec. 20-23, 2014 at Hainan China.

ISSN: 1975-0080 IJMUE

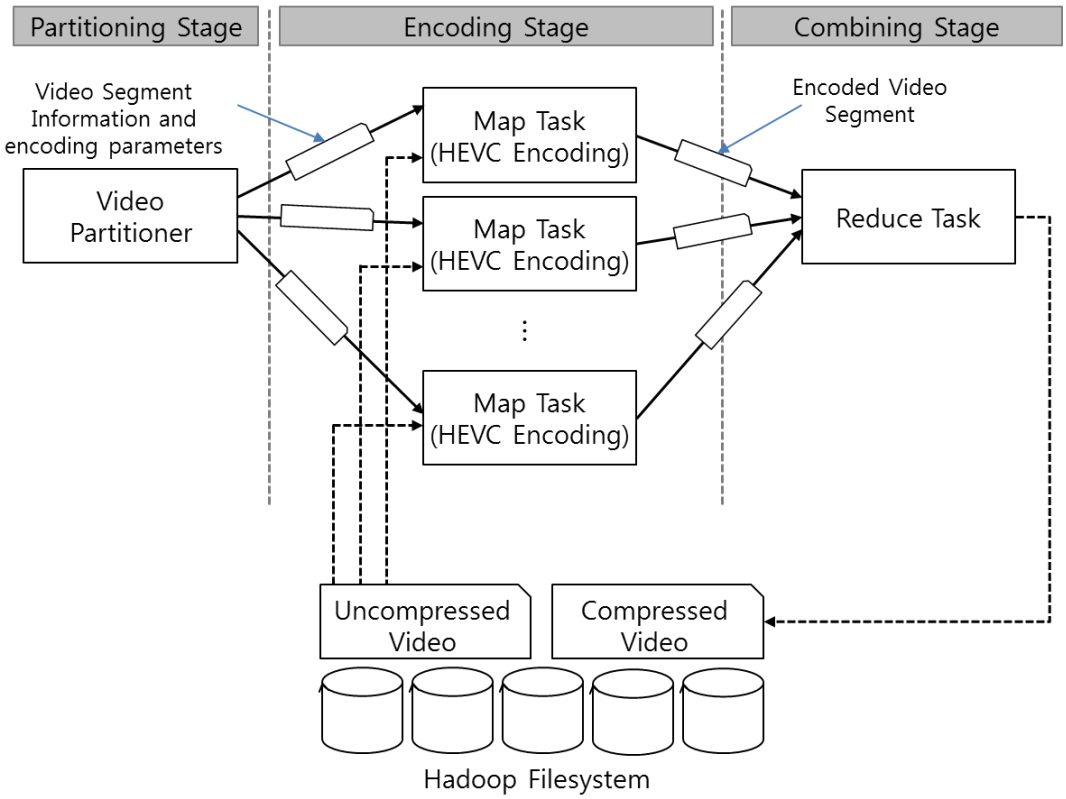


Figure 1. The Architecture of the Proposed MapReduce-based Distributed HEVC Encoder

Many studies have developed algorithms and methods to reduce HEVC encoding time. One end of the spectrum for such efforts is to optimize the internal processes of HEVC. For example, [7] introduced a fast inter prediction mode decision method, whereas [8] focused on rapid CU selection for inter prediction since in HEVC inter prediction involves the greatest computational complexity. The other end of the spectrum involves parallel and distributed processing of HEVC. Although HEVC provides parallel processing tools such as tiles and wavefront parallel processing (WPP), these are still limited in reducing encoding time. Recently, as cloud computing has emerged as a promising technique for mainstream applications, significant attention has been paid to distributed video encoding, in which encoding tasks that involve high computational complexity are distributed across the unlimited resources available in the cloud computing environment. This approach more effectively reduces encoding time than does the use of existing standalone HEVC encoders.

For distributed encoding, the input video is partitioned into individual segments that are then encoded independently via distributed computational resources. Therefore, the chosen method of video partitioning plays an important role in determining overall encoding performance such as encoding time, bitrate, and peak signal-noise ratio (PSNR). In this paper, we conduct empirical comparison of the uniform partitioning and the Group of Picture (GOP)-based partitioning methods, which are frequently used in many distributed video coding systems, for different type of videos. In the former method, the input video is uniformly partitioned into the number of nodes available to perform encoding process, whereas in the latter method the input video is partitioned on the basis of the GOP size which is provided as an encoding parameter. The comparison identifies important factors that must be taken into account when developing efficient video partitioning methods.

The remainder of this paper is organized as follows: Section 2 summarizes related studies on distributed video encoding. Section 3 provides detailed information on two video partitioning methods and Section 4 describes the architecture of the developed system for distributed HEVC encoding. Section 5 presents detailed empirical analysis of the video partitioning methods. Finally, Section 6 provides a summary and suggestions for future work.

2. Related Work

Parallel distributed video processing is not a new concept and there have been many attempts to apply this idea to multi-core processors or cluster computers. Recently, due to many advantages in term of costs as well as performance, cloud computing has emerged as a potential platform for distributed video processing. In particular, several studies [9]-[13] applied the MapReduce paradigm to distributed video encoding and transcoding. These consist of a *Map* stage, in which the input video is divided into small segments, and individual nodes run tasks to encode or transcode the assigned segments in parallel; and a *Reduce* stage, where processed segments are collected and combined into the final output video.

In [11], it is assumed that individual computing nodes used for encoding video segments have heterogeneous computing capabilities, and a scheduling algorithm is proposed to allocate complex segments to powerful nodes. This approach reduces potential capacity wastage associated with task-launching overhead. Another study explored the merits of implementing cloud computing concepts in the transcoding stage of delivering video content, by comparing the performance of three use cases with differing roles of the cloud storage [12]. In addition, the impact of video partitioning methods was studied from the point of content availability. Experimental results showed that transcoding delay was low when the video content was split into 10-second segments.

[14] and [15] focused on the length of segments in optimizing the transcoding speed: [14] proposed a multi-resolution load balancing algorithm for distributed transcoding. In that proposal, the foreside and the tail of the source media file are split into segments of different sizes, so that heavily loaded computing nodes are more likely to receive fewer finer video segments for processing than the lightly loaded nodes at the back of transcoding phase. In [15], segment length is determined so that the total transcoding time is minimized and a performance model is developed for the optimum length of segments.

3. MapReduce-based Distributed HEVC Encoder

In order to analyze the impact of video partitioning methods, we developed a MapReduce-based distributed HEVC encoder. As shown in Figure 1, the workflow of the developed system consists of three stages. In the partitioning stage, the *Video Partitioner* partitions the uncompressed video file into several non-overlapping segments according to the given video partitioning algorithm and the encoding parameters provided by users. Note that instead of retrieving the uncompressed video file, dividing it into segments, and then sending individual segments to corresponding *Map* tasks, the *Video Partitioner* simply identifies the start and end points of each segment and sends those sets of information to the *Map* tasks, thus reducing network transmission overheads.

In the encoding stage, individual *Map* tasks perform HEVC encoding for the received video segments. The present study used the HEVC test model (HM 14.0) [16] for HEVC encoding. Individual segments have different complexity and therefore require differing encoding time, despite having the same duration or number of frames. This introduces unbalanced computational loads on different nodes and eventually affects the availability time of encoded segments in the *Reducer* task. Therefore, it is necessary for the encoding stage to implement an efficient segment allocation algorithm in consideration of segment complexity and resource status. Given that our implementation is based on the

MapReduce framework, we used its default scheduling algorithm in which each *Map* task receives video segments on standby once the current encoding task is complete. One of our future research objectives is to optimize the partitioning and encoding stages in order to produce high-quality results within an acceptable encoding time.

Finally, in the combining stage, the *Reduce* task receives HEVC-encoded segments from *Map* tasks and merges them into a final HEVC-encoded video file, which is stored within the cloud.

4. Video Partitioning Methods

As with other coding standards such as H.264/MPEG-4 AVC, the HEVC-encoded video consists of multiple coded video sequences, each of which in turn comprises a sequence header and one or more GOPs. The GOP starts with an I-frame followed by a number of P- and B-frames. There are two types of GOP: Closed-GOP and Open GOP. In Closed-GOP, all reference frames belong to the same GOP, and can thus be decoded independently without referencing any frames of other GOPs. In contrast, as shown in Figure 2(a), Open-GOP can use reference frames from other GOPs. In general, Open-GOP is more efficient in coding than Closed-GOP because it reduces the temporal redundancy between consecutive frames around GOP boundaries [15], with the cost of increased delay and larger frame-storage requirements.

It is obvious that encoding many video segments in parallel can significantly reduce total encoding time. The downside is that parallel encoding suffers from quality discontinuity and degradation, due to lack of information about the coding complexity of previous video segments. In the following section, we describe two well-known video partitioning methods—uniform partitioning and GOP-based partitioning—and compare the coding structures of encoded video generated from these methods to those generated without partitioning.

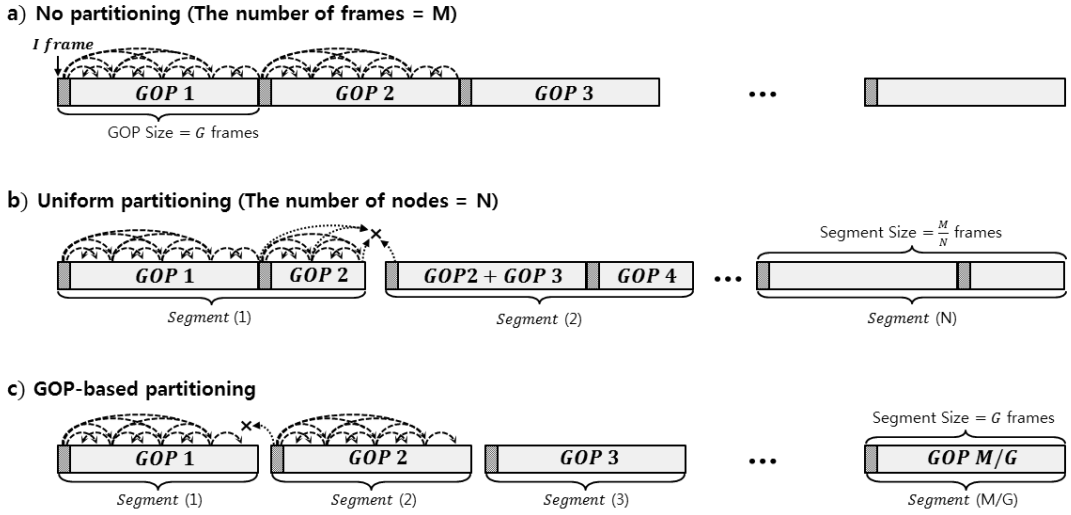


Fig. 2. Internal coding structures of encoded videos

4.1. Uniform Partitioning

Uniform-partitioning is a basic partitioning method used in many of existing distributed encoding systems. In this method, the input video is partitioned into *N* video segments of the same size (*M*/*N*), where *M* and *N* represent the total number of frames to encode and the total number of nodes to participate in encoding, respectively. Therefore, individual segments can have frames belonging to different GOPs. For instance, according to Figure 2(b), the first segment consists of frames of "GOP 1" and some

frames of "GOP 2". In this case, those frames from "GOP 2" cannot reference frames at the tail area of "GOP 2", thus leading to inefficient inter-coding. In addition, as each node encodes the received segment independently, the first frame of the encoded video must be an I-frame. Therefore, the number of I-frames may increase in accordance with the number of segments formed. Reducing the number of I-frames is crucial in reducing the bitrate of the encoded video, because I-frames have considerably higher bitrates than those of P- or B-frames.

4.2. GOP-based Partitioning

Figure 2(c) shows an example coding structure of a video encoded using GOP-based partitioning. In this method, the unit of distribution is the GOP; therefore, the size of the segment is equal to the GOP size and there are M/G segments in total, where G denotes the GOP size. Similarly to the uniform partitioning method, GOP-based partitioning also suffers from limited use of reference pictures between GOPs. For instance, "GOP 1" cannot reference the I-frame of "GOP 2" (which could be used previously as shown in Figure 2(a)) because of partitioning. Unlike uniform partitioning, however, the GOP-based method does not allow the GOP itself to be partitioned, so there is no additional increase in the number of I-frames. Note that the problem of limited use of reference pictures occurs only in the Open-GOP structure, as the Close-GOP is an independent unit that can be coded without having any frames of other GOPs. In this paper, we assume that all video streams have Open-GOP, because it provides higher coding efficiency. Depending on the duration of the video, the number of segments to be encoded may exceed the number of encoding nodes. In such cases, this method can take more encoding time than the uniform partitioning method, due to increased overhead of *Map* task creation and data transmission.

5. Empirical Analysis of Video Partitioning Methods

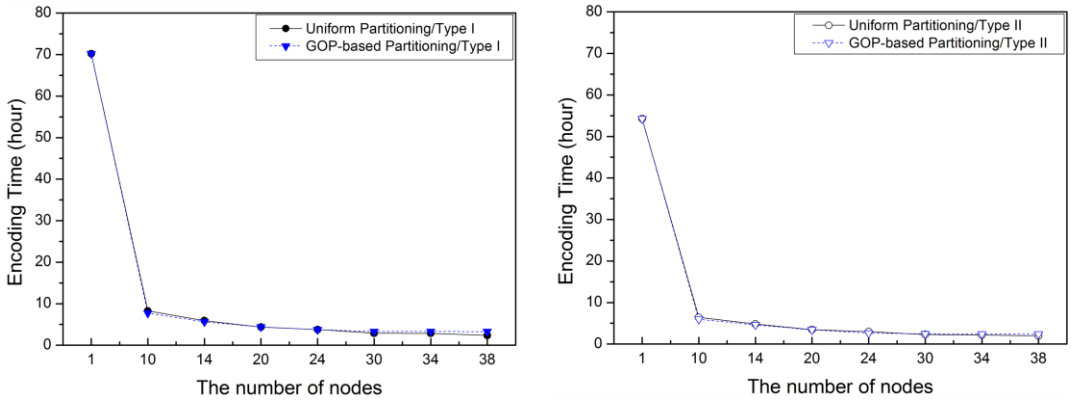
We implemented two video partitioning methods in the *Video Partitioner* of the proposed system and compared their performances using different types of videos. Table 1 describes the testbed on which the proposed system was deployed, and the input videos used in the experiments.

Table 1. Information on the Testbed and Input Videos

Category	Description
Testbed	Hadoop Version: 1.0.3 CPU: Intel Xeon E3-1220 V2 RAM: 8GB Nodes: 20
Input Video	Resolution: 4K (3,840×2,160) Total Frames: 1,400 Video Format: YUV420p Content Type: Type I, Type 2
HEVC Encoding Parameters	Mode: Random Access (CLA) FPS: 60 GOP Size: 32

Note that the Intel Xeon E3-1220 V2 CPU consists of four cores and can run four tasks in parallel without significant performance degradation. However, we limited the system to two *Map* tasks per node for minimizing task interference and conducting controlled experiments. Therefore, the maximum number of allowable *Map* tasks is 38. We differentiated input videos based on their contents. Type I videos represent scenes with

many changes in motion between frames (e.g., a soccer game), whereas Type II videos have comparatively few movements and, thus, consist of similar frames (e.g., a nature documentary). As mentioned in Section 3, Each *Map* task invokes the HM Test model for HEVC encoding. For Open-GOP structure, we used the CRA mode of the HM Test model.



(a) Type I (b) Type II

Figure 3. Changes in Encoding Time based on Video Partitioning Methods

Figure 3 shows the encoding times for two different types of videos according to the uniform and GOP-based partitioning methods. In general, encoding Type II video require less time than for Type I, which has high dissimilarity between successive frames and therefore requires more computational cycles for inter coding. In addition, the uniform partitioning outperforms the GOP-based method in terms of the encoding time. In uniform partitioning, the number of segments formed is the same as the number of nodes performing HEVC encoding. Therefore, all segments are processed in parallel. In contrast, the GOP-based method forms 44 segments, which exceeds the maximum allowable nodes. Therefore, only 38 segments can be processed in parallel, and the remaining 6 segments must stand by for processing as soon as nodes finish encoding their current segments. However, as the number of nodes increases, encoding times of two methods tend to converge as a result of network I/O rates.

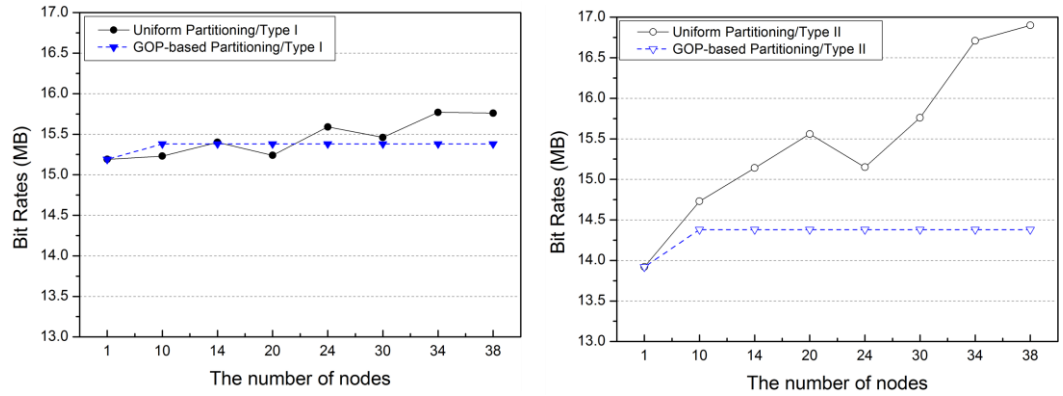


Figure 4. Changes in Bitrate based on Video Partitioning Methods

Figure 4 shows changes in bitrate. In the GOP-based method, the structure and number of segments remain the same regardless of the number of computational nodes, and only the number of segments being processed simultaneously can differ. As a result, the number of nodes does not affect the bitrate, except for the case with a single node. In this method, the number of I-frames is the same in all cases. However, when segments are

processed in a distributed manner, they cannot reference frames of other GOPs, thus leading to increased bitrates compared to the single node case.

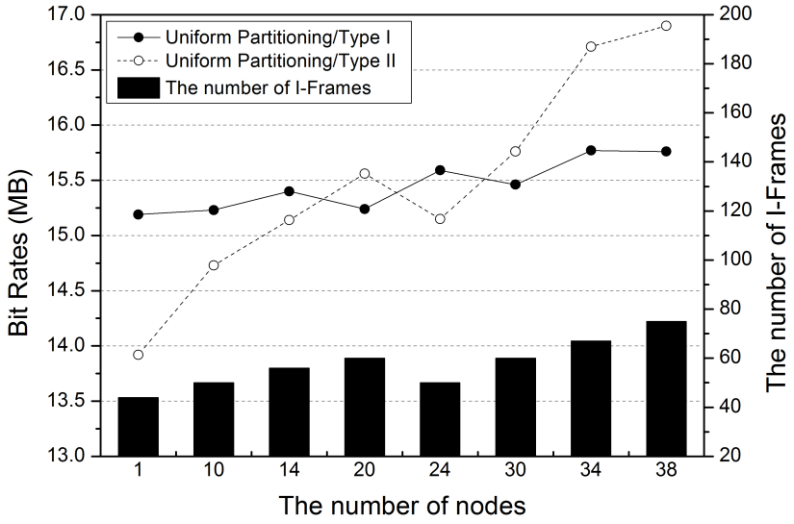


Figure 5. The Effect of the Number of I-frames on Bitrate

For uniform video partitioning, due to the increased number of I-frames and inability to reference frames belonging to other GOPs, bitrates tend to increase as the number of computational node increases. Interestingly, however, in some cases, the total bitrate decreased although the number of I-frames increased. For instance, in Figure 5, when encoding Type I video using 14 or 20 nodes, the number of I-frames generated were 56 and 60, giving respective bitrates of 15.4 MB and 15.24 MB. This result suggests that, depending on how the input video is partitioned, it is possible to achieve zero performance loss in terms of both encoding time and bitrate. A similar observation was made for the Type II video.

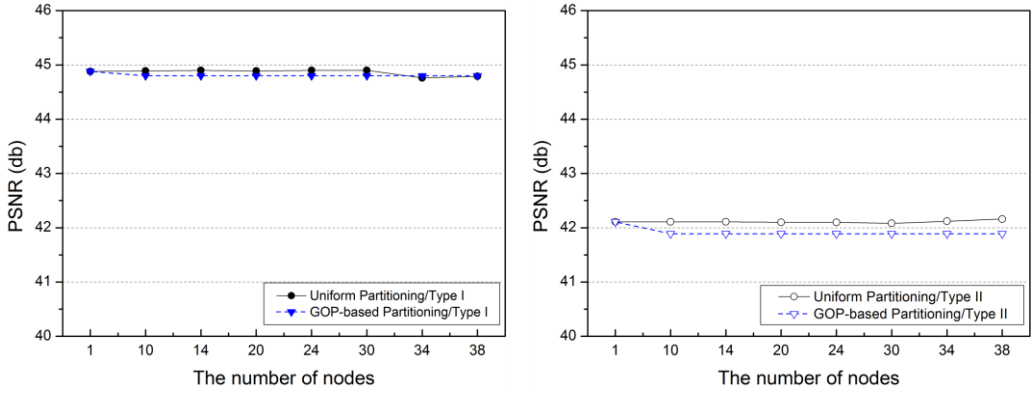


Figure 6. Changes in PSNR based on video Partitioning Methods

Finally, Figure 6 shows changes in PSNR. PSNRs for the Type II video were slightly higher than those for Type I. This is because the content of Type II video is relatively static and, as a result, its frames can be well coded. Overall, although there were some changes in PSNR, they were marginal. In summary, the experimental results show that video partitioning methods can affect various performance parameters of video encoding. In particular, the uniform video partitioning method exhibits some tendency in performance wise, whereas the bitrate generally (but not always) increases with the number of computational nodes. Therefore, one of our future research objectives is to

develop video partitioning algorithms to optimize the number of nodes necessary to achieve rapid HEVC encoding without significant increases in the bitrate.

6. Conclusion and Future Work

As users demand increasingly higher-quality, higher-resolution digital video content such as UHD, the computational complexity of associated video coding technologies also increases. Dedicated hardware solutions for video encoding can provide the best solutions but they involve higher costs and are difficult to extend. As a result, from the performance and cost perspectives, distributed video encoding based on cloud computing has gained significant attention in recent years. Distributed video encoding requires partitioning and allocating chunks of the input video to appropriate computational nodes for subsequent encoding. Therefore, the present study explored the effects of the basic video partitioning methods on various performance parameters, such as encoding time, bitrate, and PSNR.

The experimental results showed that cloud-based distributed encoding reduces the total encoding time significantly for HEVC, which was previously shown to involve 2–10 times greater computational complexity. In terms of bitrate, a greater number of segments generally results in higher bitrate of the encoded video, due to increased I-frames and/or limited use of reference frames, especially for Open-GOP structures. Interestingly, however, for uniform partitioning, a scenario with fewer segments does not always outperform one with more segments in terms of bitrate. This finding suggests that, depending on the approach to video partitioning, fast HEVC encoding can be achieved with higher number of I-frames yet no corresponding increases in bitrates. Our future research will also focus on developing various video partitioning algorithms, considering both encoding time and bitrate.

Acknowledgements

This work is supported by Kyonggi University Research Grant 2014.

References

[1] ITU-T and ISO/IEC JCT 1, Advanced Video Coding for Generic Audiovisual Services, ITU-T Recommendation H.264 and ISO/IEC 14496-10 (MPEG-4 AVC), **(2011).**
[2] ITU-T and ISO/IEC JTC 1, High Efficiency Video Coding (HEVC), ITU-T Recommendation H.265 and ISO/IEC 23008-2, **(2013)** April.
[3] J. Ohm, G. Sulliva, H. Schwarz, T. Tan, and T. Wiegand, "Comparison of the Coding Efficiency of Video Coding Standards – Including High Efficiency Coding (HEVC)", IEEE Trans. Circuits and Systems for Video Technology, vol. 22, no. 12, **(2012)** December, pp. 1669-1684.
[4] M. Pourazad, C. Doutre, M. Azimi, and P. Nasiopoulos, "HEVC: The New Gold Standard for Video Compression", IEEE Consumer Electronics Mag, vol. 1, no. 3, **(2012)** January, pp. 36-46
[5] F. Bossen, B. Bross, K. Suhring, and D. Flynn, "HEVC Complexity and Implementation Analysis", IEEE Trans. Circuits and Systems for Video Technology, vol. 22, no. 12, **(2012)** December, pp. 1685-1696.
[6] K. McCann, B. Bross, W. Han, I. Kim, K. Sugimoto, and G. Sullivan, "High Efficiency Video Coding (HEVC) Test Model 14 (HM 14) Encoder Description," JCT-VC, Doc. JCTVC-P1002, **(2014)** March.
[7] A. Lee, D. Jun, J. Kim, J. Choi, and J. Kim, "Efficient Inter Prediction Mode Decision Method for Fast Motion Estimation in High Efficiency Video Coding", ETRI Journal, vol. 36, no. 4, **(2014)**, pp. 528-536
[8] J. Xiong, H. Li, Q. Wu, and F. Meng, "A Fast HEVC Inter CU Selection Method Based on Pyramid Motion Divergence", IEEE Trans. Multimedia, vol. 16, no. 2, **(2014)**, pp. 559-564.
[9] R. Pereira, M. Azambuja, K. Breitman, and M. Endler, "An Architecture for Distributed High Performance Video Processing in the Cloud", Proceedings of 2010 IEEE 3rd International Conference on Cloud Computing, **(2010).**
[10] L. Zheng, L. Tian, and Y. Wu, "A Rate Control Scheme for Distributed High Performance Video Encoding in Cloud", Proceedings of 2011 International Conference on Cloud and Service Computing, **(2011).**
[11] F. Lao, X. Zhang, and Z. Guo, "Parallelizing Video Transcoding Using Map-Reduce-Based Cloud Computing", Proceedings of 2012 IEEE International Symposium on Circuits and Systems, **(2012).**

[12] A. Junzel, H. Kalva, and B. Furht, "A Study of Transcoding on Cloud Environments for Video Content Delivery", Proceedings of 2010 ACM Multimedia Workshop on Mobile Cloud Media Computing, **(2010).**
[13] M. Kim, Y. Cui, S. Han, and H. Lee, "Towards Efficient Design and Implementation of a Hadoop-based Distributed Video Transcoding System in Cloud Computing Environment", International Journal of Multimedia and Ubiquitous Computing, vol. 8, no. 3, **(2014)**, pp. 213-224.
[14] Z. Tian, J. Xue, W. Hu, T. Xu, and N. Zheng, "High Performance Cluster-based Transcoder", Proceedings of 2010 International Conference on Computer Application and System Modeling, **(2010).**
[15] Y. Sambe, S. Watanabe, D. Yu, T. Nakamura, and N. Wakamiya, "High-speed Distributed Video Transcoding for Multiple Rates and Formats", IEICE Trans. on Information and Systems, vol. E88-D, no. 8, **(2005)**, pp. 1923-1931.
[16] HEVC Test Model (HM 14.0), http://hevc.hhi.fraunhofer.de/.