A Practical Low Latency System for Cloud-Based VR Applications

Shuangfei Tian, Mingyi Yang(✉), and Wei Zhang

State Key Laboratory of ISN, Xidian University, Xi'an, China
myyang_96@stu.xidian.edu.cn

Abstract. With the development of multimedia technologies, VR services have quickly gained popularity at an accelerating speed. To reduce the high cost of purchasing high-performance VR terminals for end users and to enhance the user experience, recently, the concept of cloud-based VR was proposed which brings the cloud computing technologies to VR services. On-cloud GPU clusters and multi-core servers are expected to be used for simplifying VR terminals at the users' side. This idea, however, arises several challenges in deploying such cloud-based VR system for practical applications, among which the cloud-to-end latency is mainly concerned. In this paper, we designed a practical solution for bearing cloud-based VR applications. We aim at reducing the cloud-to-end latency to improve the experience of end users. In our system, a frame splitting technique was proposed to fulfill the goal. Specially designed algorithms including reference frame determination and rate control strategies were also included to limit the computational complexity and improve the coding efficiency while obtaining promising user experience. Experimental results showed that the proposed system can significantly reduce the cloud-to-end latency.

Keywords: Cloud-based VR · Cloud-to-end latency · H.264 coding scheme · Rate control

1 Introduction

Recent years have witnessed tremendous progress in the development of virtual reality (VR) technologies, which are now widely applied in many fields including education [6], entertainment [13], medicine [11] and gaming [9]. It is predicted that VR services will have a global user base of more than 275 million by 2025 [5]. A recent report foresaw that the VR business will grow into an $80 billion market by 2025 [1]. Currently, VR ecosystems is progressing in an accelerate pace in various aspects including the design of VR terminals (e.g., Head-Mounted Display (HMD)), coding and rendering schemes as well as transmission strategies.

This research was supported by the National Nature Science Foundation of China (Grant No. 61801364) and the Fundamental Research Funds for the Central Universities (Grant No. JB180105).

H. Gao et al. (Eds.): ChinaCom 2019, LNICST 313, pp. 73–81, 2020.
https://doi.org/10.1007/978-3-030-41117-6_7

Current VR terminals can be generally classified into three categories including mobile phone-based VR terminals, all-in-one VR terminals and PC-based VR terminals. The rather limited computing capability of the mobile phone-based and all-in-one VR terminals restrict their application to mainly VR videos services (i.e., 360-degree video streaming). In contrast, PC-based VR terminals are usually used to perform more demanding VR applications such as interactive VR gaming. However, most PC-based VR terminals are tethered to PCs using HDMI cables, which limits the movements range of end users and brings inconvenience. Moreover, to bring users with better sense of immersion and presence, visual scenes are expected to be rendered in higher resolution, e.g., 12K/24K with at 30/60 fps [8]. However, the capability of existing hardware fails to meet these challenging requirements. Additionally, mainstream PC-based VR manufacturers reported that the minimum configuration requirements for a basic VR experience will cost at least USD 1,500 including the purchase of a PC and an HMD [8]. This price will significantly arise if better visual experience is expected. The heavy cost of entry surely retard the popularity of VR applications among potential users. In this regard, it is of fundamental importance to design more portable, affordable and practical VR solutions for end users.

Very recently, the concept of cloud-based VR was proposed which brings the cloud computing technologies to VR services. The so-called VR cloudification aims at reducing the burden of logical computing and rendering process of VR terminals. The on-cloud GPU cluster and multi-core servers are used to compensate the deficiency of VR terminals. Figure 1 shows the basic system. The logic computing and rendering modules that have required high performance requirements for the terminal are placed in the cloud. VR terminals at the users' side only needs perform less demanding tasks such as sensor data transmission, video decoding and interaction control commands transmission. Such a change reduces the performance requirements of the VR terminal under the premise of ensuring the user experience, making it possible for VR to truly enter our daily life. Investigating the potential challenge as well as development practical cloud-based VR systems become a new research topic in both academia and industry.

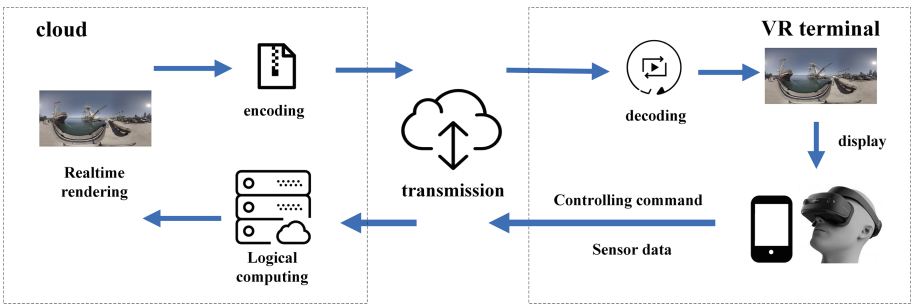

Fig. 1. An overview of the basic cloud-based VR system.

In September 2018, Huawei released the Cloud VR Solution White Paper which expressed its vision on the cloud-pipe-terminal Cloud VR [8]. Challenges of deploying such cloud-based VR system, e.g., the cloud-to-end latency, are discussed. Likewise, in [4], the authors discussed difficulties in improving the accessibility of interactive VR gaming experience by utilizing cloud techniques. They pointed out that cloud-to-end latency is the main issue to be solved before the successful deployment of cloud-based interactive VR gaming system. In [12], an implementation of an interactive VR game with the capability to move game servers across the world was demonstrated. However, end users reported a latency to different extent according to their distance to the server. Similarly, research in [7] investigated the challenges for enabling the cloud-based VR applications in wireless networks. They also analysed the challenging bitrate and latency requirements to enable wireless VR.

From the research above, we can see that solving the issue of cloud-to-end latency is the key factor in designing cloud-based VR system. In this paper, we designed a practical solution for bearing cloud-based VR application. More specifically, we aim at reducing the cloud-to-end latency to enhance the user experience. A frame splitting technique was first proposed to reduce the latency. Other algorithms including reference frame determination and rate control strategies were also included to limit the computational complexity and improve the coding efficiency while obtaining promising user experience. Experimental results showed that the proposed system can significantly reduce the cloud-to-end latency.

2 The Design of the Low Latency System

2.1 Overview of the Proposed System

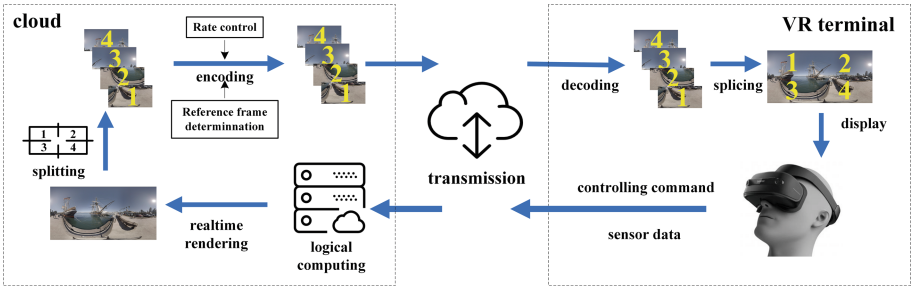

Fig. 2. An overview of the proposed system.

Figure 2 shows an overview of the proposed system. The entire VR rendering process was re-allocated where the logical computing, real-time rendering and encoding process are performed by the cloud server. In the original cloud-based

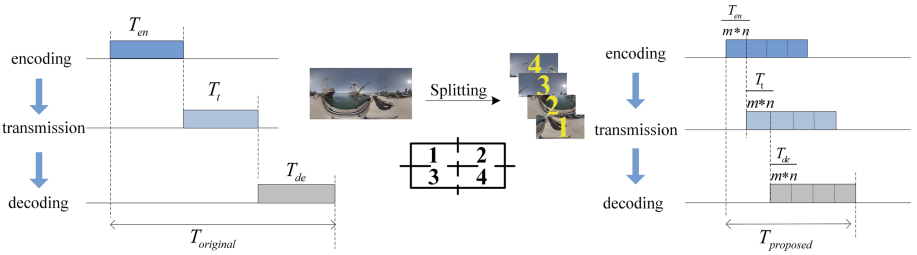

Fig. 3. The comparison of the cloud-to-end latency between no splitting and M × N splitting. M and N are set to be 2 for illustration.

VR system, the transmission of each frame should only begin after the codec has finished encoding that entire frame. Similarly, the decoding process only begin after the client receive the complete stream of one frame. Its corresponding cloud-to-end latency is denoted as $T_{original}$ in Fig. 3. To reduce the cloud-to-end latency, we proposed a frame splitting module in our system. As shown in Fig. 2, the cloud rendering platform first split each frame into M × N (e.g., 2 × 2 as illustrated) sub-frames. The obtained M × N sub-frames are then arranged to form a new temporal sequence in lower resolution. In this setup, once a part (e.g., a sub-frame) of the entire frame is encoded, it can be sent for transmission without waiting for the rest part to be encoded. Besides, as long as the client receive the stream of a sub-frame, the decoding process can be started. The cloud-to-end latency $T_{proposed}$ in our proposed system is shown in Fig. 3 which is clearly shorter. The reduction of cloud-to-end latency depends on the number of sub-frames divided.

When the sub-frame sequence are encoded, the temporal relativity between them is different from that of normal sequences, so a reference frame selection strategy is proposed. Moreover, as each frame is split into multiple sub-frames and each sub-frame is encoded separately, it is therefore important to make sure the sub-frames that belongs to the same original frame are consistent in their coding quality. To do so, a rate control module was further proposed in our system to keep that consistency. As current multimedia devices are compatible with H.264 coding scheme, our system thus used X264 codec as the basis.

2.2 Reference Frame Determination

After the frame splitting process, temporally adjacent sub-frames do not possess high relativity. Instead, this temporal relativity appears in every M × N sub-frames. Since the efficient prediction in temporal domain is a key technique for improving the coding efficiency, how to effectively use the temporal correlation is of fundamental importance.

The H.264/AVC video coding scheme [14] supports multi-reference frame in the temporal domain. More specifically, the encoder stores a number of reconstructed video frames as reference. Motion prediction is performed in these reference frames to find a more accurate match. However, X264 codec needs to

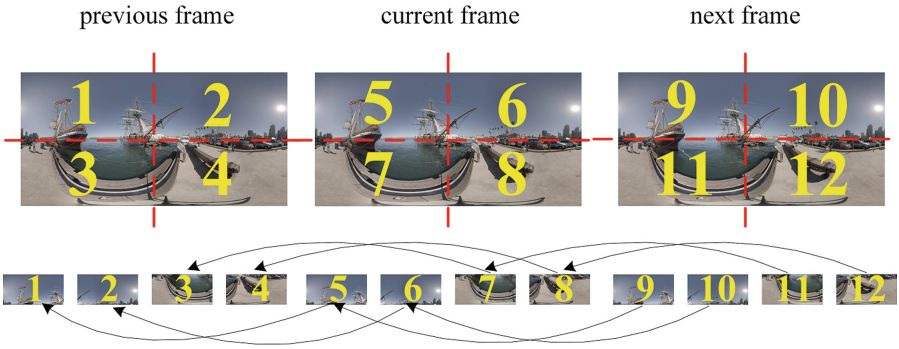

Fig. 4. Illustration of the prediction dependency of the sub-frames generated by the 2×2 splitting.

perform motion prediction in all reference frames when using multiple reference frames, which greatly increases the computational complexity compared to using a single reference. Especially, in the proposed system, it is no meaningful to perform motion prediction in those candidate reference sub-frames that have no temporal relativity with current sub-frame. As the sequence to be coded in our system has a fixed temporal structure, we hope to directly use single reference frame that is adjacent in temporal domain to control the computational complexity while ensuring sufficient coding efficiency. The expected prediction dependency between sub-frames just like Fig. 4 shows. With this in mind, we proposed an algorithm which can be achieved by setting the related syntax utilizing the H.264/AVC reference frame modification technique:

(1) Setting the reference frame list re-ordering flag:

$$ref\_pic\_list\_reordering\_flag\_10 = 1 \tag{1}$$

(2) Setting the type of modification process to be short-term reference frame modification where the re-ordered reference frame precedes the current frame:

$$reordering\_of\_pic\_nums\_idc = 0 \tag{2}$$

(3) Setting the distance between the current frame and the re-ordered frame to be m × n:

$$abs\_diff\_pic\_num\_minus1 = m * n \tag{3}$$

As the X264 codec supports no more than 16 reference frames, m × n must less than 16. That is to say, our splitting module thus allows several spatial partitions that generates less than 16 sub-frames.

By using the technique above, the reference frame list will be re-ordered. So that the right frame can be referred when just using single reference frame.

2.3 Rate Control for Quality Consistency

In our system, a frame will be divided into a number of sub-frames for encoding, and displayed after the client spliced them together. If the quality of the sub-frames that belong to the same original frame is significantly different, the user experience will be drastically reduced. To ensure the quality consistency among the splitted sub-frames, we designed a rate control algorithm as shown in Algorithm 1 based on the averaged bit rate (ABR) method [10] of X264 scheme.

Algorithm 1. Rate control for quality consistency

Input:

$SATD$: the sum of absolute transformed difference of the current sub-frame[10];

$cplxsum$: iterative quantities initialized according to the number of macroblocks[10];

$cplxcount$: iterative quantities initialized to be 0;

$qcompress$: the compression control parameter which equals to 0.6;

$rate\_factor$: the ratio of the summation of all coded bits to the complexity of current sub-frame;

$overflow$: the deviation between the total target bits and the actual generated bit;

$preQP$: quantification parameter of the previous sub-frame;

a, b, c : parameters which empirically set to be 12, 6 and 0.85 respectively;

Output: QP : quantification parameter;

1: $cplxsum \leftarrow cplxsum * 0.5 + SATD$
2: $cplxcount \leftarrow cplxcount * 0.5 + 1$
3: $blurred\_complexity \leftarrow cplxsum/cplxcount$
4: **if** current sub-frame is the first one in that frame **then**
5: $\quad qscale \leftarrow blurred\_complexity^{(1-qcompress)}$
6: $\quad qscale = qscale/rate\_factor$
7: $\quad qscale = qscale * overflow$
8: $\quad QP \leftarrow a + b \cdot log_2(qscale/c)$
9: **else**
10: $\quad QP \leftarrow preQP$
11: **end if**
12: $preQP \leftarrow QP$
13: **return** QP

In Algorithm 1, the step 4 to step 11 calculate the QPs of each sub-frame. The QPs of all sub-frames in one frame are set to be equal to that of the first sub-frame, To do so, we can ensure the quality consistency of the entire frame. However, the $blurred\_complexity$ of each sub-frame should always be calculated to ensure the correct calculation of $qscale$ for the following sub-frames. In step 5 the $qscale$ is firstly calculated according to $blurred\_complexity$, then the step 6 and 7 revise it according $rate\_factor$ and $overflow$.

3 Performance of the System

To benchmark the performance of the proposed system, we measured the coding efficiency as well as the cloud-to-end latency. The test sequences are four standard omnidirectional videos provided by Joint Video Exploration Team [3].

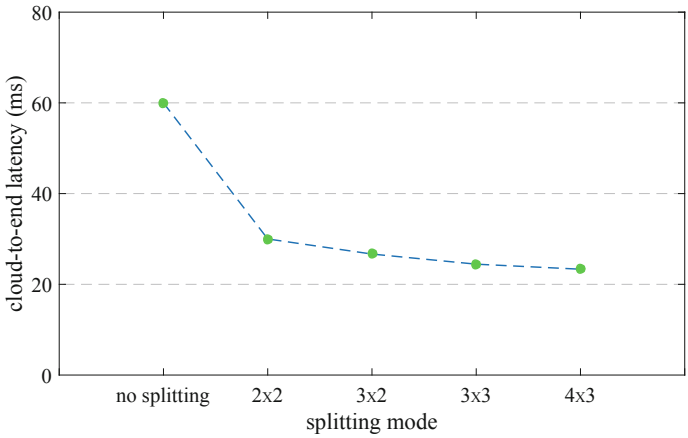

Fig. 5. Illustration of the cloud-to-end latency under different splitting mode.

3.1 Cloud-to-End Latency

The proposed system is designed to reduce the cloud-to-end latency by including the frame splitting module. As already shown in Fig. 3, when no splitting is used, the cloud-to-end latency can be calculated as:

$$T_{original} = T_{en} + T_t + T_{de} \tag{4}$$

where T_{en} is the encoding time of the entire frame, T_t is the transmission delay, T_{de} is the decoding time. In contrast, when a frame is divided into M × N subframes, the cloud-to-end latency can be calculated as:

$$T_{proposed} = \frac{T_{en}}{m * n} + \frac{T_t}{m * n} + T_{de} \tag{5}$$

To provide numeric evidence on to what extent can the proposed method reduce latency, we empirically set the encoding time, transmission delay and the decoding time for a single 4 K frame as 20 ms each. Note that the time of 20 ms was empirically set based on typical statistics in [8] to show the varying tendency of the cloud-to-end latency for different splitting mode. Real-world processing time in each step may be different according to specific methods used. Figure. 5 plots the cloud-to-end latency with different splitting mode. It shows that the latency reduces from 60 ms in no splitting mode to 23.3 ms in 4×3 mode. In the case of 4×3 splitting mode, the cloud-to-end latency is mainly occupied by the decoding time (i.e., 20 ms out of 23.3 ms) which is mainly determined by the client.

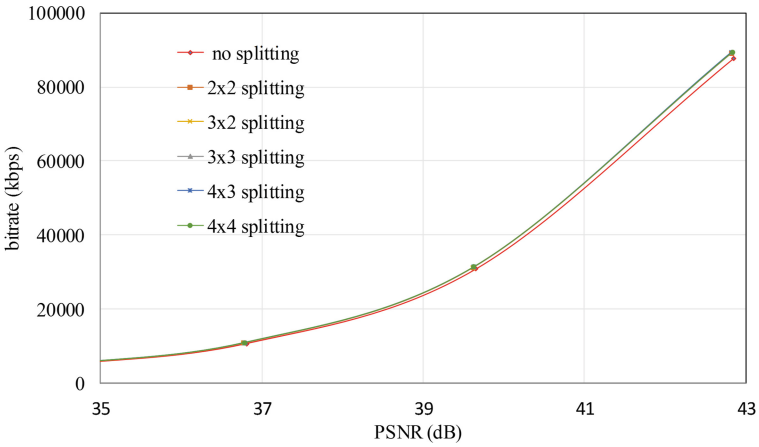

Fig. 6. Illustration of RD curves for encoding the sequence called PoleVault_le.

3.2 Coding Efficiency

Table 1 lists the performance of the proposed system in terms of Bjøntegaar Delta-rate (BD-rate) criterion [2] between no splitting mode and various possible partitions. Figure 6 further shows the RD curve for encoding the sequence called PoleVault_le. They showed that the coding efficiency of the proposed system experiences a slight drop since the spatial correlation of frames are affected due to the frame splitting. However, even at the largest 4×3 splitting mode, the BD-rates for different video content are controlled to be smaller than 4%, indicating that the coding efficiency of the proposed system can be relatively maintained.

Table 1. Performance in terms of the BD-rate of the proposed system with different $M \times N$ splitting mode

	2×2	3×2	3×3	4×3
AerialCity	1.42%	1.86%	3.13%	3.93%
Harbor	2.23%	2.36%	3.34%	3.89%
PoleVault_le	2.62%	2.75%	3.01%	3.15%
KiteFlite	2.57%	2.81%	2.94%	3.25%

4 Conclusion

This paper presents a practical cloud-based VR system to reduce the cloud-to-end latency for VR applications. A frame splitting technique was proposed to reduce the latency. To make sure the perceived quality of each sub-frame after

splitting is consistent, a rate control method was further proposed. Experimental results showed that the cloud-to-end latency can be significantly reduced by almost three times in our implementations. In the meanwhile, the video quality only experiences a slight drop in terms of BD-rate. Experimental results showed that the proposed system can significantly reduce the cloud-to-end latency.

References

1. Bellini, H.: The real deal with virtual and augmented reality (January 2016). http://www.goldmansachs.com/our-thinking/pages/virtual-and-augmented-reality.html
2. Bjontegaard, G.: Calculation of average PSNR differences between RD-curves. VCEG-M33 (2001)
3. Boyce, J., Alshina, E., Abbas, A., Ye, Y.: JVET common test conditions and evaluation procedures for 360 video. Joint Video Explor. Team ITU-T SG **16** (2017)
4. Chan, K., Ichikawa, K., Watashiba, Y., Iida, H.: Cloud-based VR gaming: our vision on improving the accessibility of VR gaming. In: Proceedings of the International Symposium on Ubiquitous Virtual Reality, pp. 24–25 (June 2017)
5. Espelien, J.: The future of consumer VR and its impact on video viewing, 2016–2025 (January 2016). http://tdgresearch.com/report/the-future-of-consumervr-and-its-impact-on-video-viewing-2016-2025/
6. Freina, L., Ott, M.: A literature review on immersive virtual reality in education: state of the art and perspectives. eLearning Softw. Educ. (1), 133–141 (2015)
7. Hou, X., Lu, Y., Dey, S.: Wireless VR/AR with edge/cloud computing. In: Proceedings of the 26th International Conference on Computer Communication and Networks, pp. 1–8 (July 2017)
8. Ltd., H.T.C.: Cloud VR solution white paper (September 2018). https://www.huawei.com/en/press-events/news/2018/9/cloud-vr-solution-white-paper
9. McMahan, R.P., Bowman, D.A., Zielinski, D.J., Brady, R.B.: Evaluating display fidelity and interaction fidelity in a virtual reality game. IEEE Trans. Visual. Comput. Graph. **18**(4), 626–633 (2012)
10. Merritt, L., Vanam, R.: X264: a high performance H. 264/AVC encoder (2006). http://neuron2.net/library/avc/overview_x264_v8_5.pdf
11. Moglia, A., Ferrari, V., Morelli, L., Ferrari, M., Mosca, F., Cuschieri, A.: A systematic review of virtual reality simulators for robot-assisted surgery. Eur. Urol. **69**(6), 1065–1080 (2016)
12. Schmoll, R., Pandi, S., Braun, P.J., Fitzek, F.H.P.: Demonstration of VR/AR offloading to mobile edge cloud for low latency 5G gaming application. In: Proceedings of the 15th IEEE Annual Consumer Communications Networking Conference, pp. 1–3 (January 2018)
13. Sreedhar, K.K., Aminlou, A., Hannuksela, M.M., Gabbouj, M.: Viewport-adaptive encoding and streaming of 360-degree video for virtual reality applications. In: Proceedings of the IEEE International Symposium on Multimedia, pp. 583–586 (December 2016)
14. Wiegand, T.: Draft ITU-T recommendation and final draft international standard of joint video specification (ITU-T Rec. H. 264— ISO/IEC 14496–10 AVC). JVT-G050 (2003)