Regular Paper

A Distributed Internet Live Broadcasting System for Multi-Viewpoint Videos

Satoru Matsumoto*, Tomoki Yoshihisa*, Tomoya Kawakami**, and Yuuichi Teranishi***

*Cybermediacenter, Osaka University, Japan
** Graduate School of Information Science, Nara Institute of Science and Technology, Japan
***National Institute of Information and Communications Technology, Japan
smatsumoto@cmc.osaka-u.ac.jp

Abstract—With the recent popularization of omnidirectional cameras, multi-viewpoint live videos are now often broadcast via the Internet. Multi-viewpoint live broadcasting services allow viewers to change their viewpoints arbitrarily. To reduce the computational load of video processes such as effect additions, various distributed Internet live broadcasting systems have been developed. These systems are designed for single-viewpoint live videos, in which the screen images (images to be watched by viewers) are the same for all viewers. However, in multi-viewpoint Internet live broadcasting services, the screen images differ according to the viewpoint selected by the viewer. Thus, one of the main research challenges for multi-viewpoint Internet live broadcasting is how to reduce the computational load of adding effects under different screen images. In this paper, we propose and develop a distributed multi-viewpoint Internet live broadcasting system. To distribute the computational load of video processes, our proposed system adopts ECA (event, condition, action) rules. For the systems using ECA rules, it is difficult to determine whether effects should be added on the server side or the player side. To determine this to reduce the computational load effectively, we classify ECA rules.

Keywords: Streaming Delivery, Internet Live Broadcasting, Multi-viewpoint Camera

1 INTRODUCTION

With the recent popularization of omnidirectional cameras, multi-viewpoint live videos are often broadcast through the Internet. In multi-viewpoint Internet live broadcasting services, viewers can arbitrarily change their viewpoints. For example, major live broadcasting services such as YouTube Live and Facebook provide 360° videos in which each user can select their desired viewpoint. In recent Internet live broadcasting services, viewers or broadcasters have been able to add video or audio effects to the broadcast videos. To reduce the computational load including them for adding such effects, a number of distributed Internet live broadcasting systems have been developed [1], [2].

These systems are designed for single-viewpoint live videos, and the screen images (images to be watched by viewers) are the same for all viewers. Therefore, screen images can be shared among processing servers, and the computational load can be reduced by exploiting distributed computing systems. However, in multi-viewpoint Internet live broadcasting services, the screen images differ according to the viewpoint selected by the user. Thus, screen images cannot be shared among processing servers. Here, one of the main research challenges for multi-viewpoint Internet live broadcasting systems is how to reduce the computational load required to add effects under different screen images.

In this paper, focusing on this challenge, we propose and develop a distributed multi-viewpoint Internet live broadcasting system. In our proposed system, video effects that can be shared among viewers are added by some distributed processing servers (i.e., on the server side). Video effects that cannot be shared among viewers are added by video players (i.e., on the player side). In such systems, it is difficult to determine whether it is better to add effects on the server side or player side. To determine this so as to effectively reduce the computational load, we use grouped rules. Moreover, we develop a distributed multi-viewpoint Internet live broadcasting system adopting our proposed rules system.

The remainder of this paper is organized as follows. In Section 2, we introduce some related work. We describe the design and the architecture of our proposed system in Section 3. Evaluation results are presented in Section 4 and discussed in Section 5. Finally, we conclude this paper in Section 6.

2 RELATED WORK

Some systems for distributing video processing loads have been proposed. Most of them fix load distribution procedures in advance. However, starting Internet live broadcasting is easy in recent years, and it is difficult to grasp which machines start Internet live broadcastings. Therefore, conventional systems establish load distributions at server side.

MediaPaaS encodes, re-encodes, and delivers video using a server machine provided by cloud computing services [2]. Different from MediaPaas, our proposed system establishes load distributions using PIAX [3], a P2P agent platform. The system has multiple servers to broadcast videos, and once a client (video recording terminal) connects to a server to broadcast its recorded video, one of the servers is randomly selected by the load distribution server. The loads caused by broadcasting videos are distributed among the servers. In [1], we confirmed that the video processing time for encoding and distributing videos can be reduced by distributing the processing load to some servers.

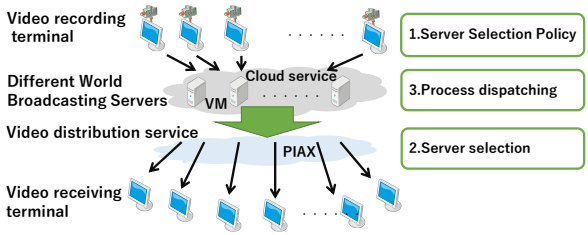


Figure 1: System architecture of the different world broadcasting system

An Internet live broadcasting system that allows the viewing of recently recorded videos (playback) was proposed in [4]. Several methods have been proposed for reducing the delay time for the distribution of videos in live Internet broadcasting. SmoothCache 2.0 [5], video data from other peers are cached and distributed among a P2P network. As a result, the communication load and delay times are reduced. Dai et al. also proposed a distributed video broadcasting system using P2P networks to reduce delay times [6]. In the HD method proposed in [7], communication traffic is reduced by simultaneously transmitting image data to a number of viewers using one-to-many broadcasting with one-to-one communication. Even in our proposed system, these delay reduction methods can be applied when delivering videos, but our current research considers the addition of video or audio effects.

Gibbon et al. proposed a system that performs video processing by transferring the data captured by a camera to a computer with high processing capabilities [8]. Ting et al. proposed a system that directly stores images captured by computers with low processing power in external storage devices, such as cloud storage [9]. However, these systems target stored video data and cannot be applied to live Internet broadcasting.

J. Bae et al. proposed a concept of blocks to classify processing flows into several patterns. A block is a minimal unit that specifies the behaviors represented in a process model [10]. A. Frömmgen et al. proposed a learning algorithm of complex nonlinear network nodes by genetic algorithm and ECA rules in [11]. As described in these papers, it is important to learn effective sequences to execute ECA rules. These are not focused on multi-viewpoint image processing. We propose a model focused on image processing with multi-viewpoint image processing.

3 DISTRIBUTED INTERNET LIVE BROADCASTING SYSTEM

In this section, we explain our previously developed cloud-based live broadcasting system using ECA (event, condition, action) rules. After that, we explain our proposed multi-viewpoint Internet live broadcasting system.

3.1 Different World Broadcasting System

3.1.1 Summary

In our previous research [1], we constructed a different-world broadcasting system using virtual machines (VMs)

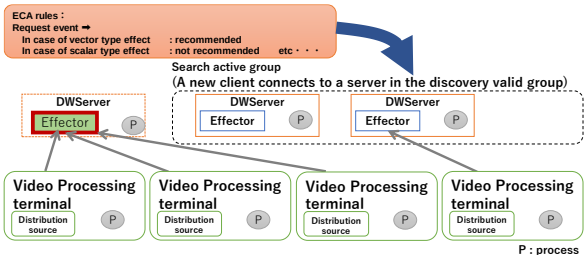


Figure 2: Load distribution mechanism using PIAX

provided by a cloud service. These machines work as the different world broadcasting servers that add video effects. In general, a number of VMs can easily be used in a cloud service. The use of multiple VMs as different world broadcasting servers enable a high-speed addition of effects by distributing the load among different world broadcasting servers. Therefore, we implemented a distributed live Internet broadcasting system using the cloud service and evaluated its performance. In our developed system, video effect additions are executed on the VMs provided by the cloud service.

Processing loads on different world broadcasting servers can be distributed by considering the load distribution when selecting a server. In conventional systems, load distribution is established by connecting processing servers via a load balancing mechanism such as a load balancer. In this method, when the load distribution mechanism needs to switch to another server while the video is being transmitted, the connection is interrupted. For this reason, it is difficult to switch servers while keeping smooth video plays. Therefore, in the different world broadcasting system, the load balancing mechanism selects a different world broadcasting server based on the requests.

3.1.2 System Architecture

The system architecture of the different world broadcasting system is shown in Fig. 1. There are three types of machine. The first is the client, which has cameras and records live videos. The second is the different world broadcasting servers, which execute processes for videos such as encoding, decoding, or video effect additions. The third type is the viewer, which plays the live videos. Each client selects a different world broadcasting server that executes the desired video effect, and transmits the video effect library and the recorded video to the different world broadcasting server. The different world broadcasting server is a VM of the cloud service that executes video processing on the video transmitted from the clients according to their requests. The video processed by the different world broadcasting server is delivered to the viewers via the video distributions service. In the system, viewers receive the processed video after selecting the server or channel of the video distributions service.

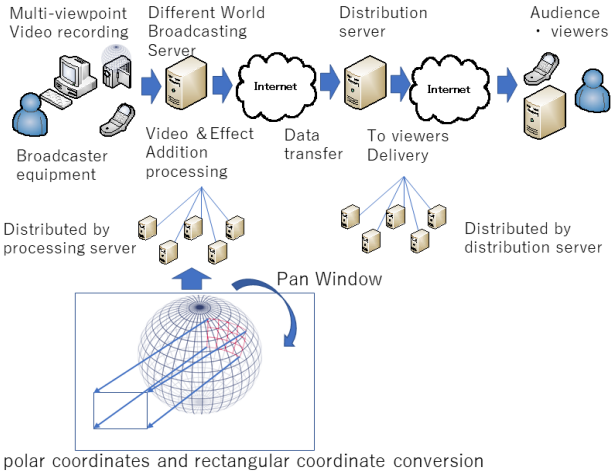


Figure 3: An overview of our designed distributed multi-viewpoint Internet live broadcasting system

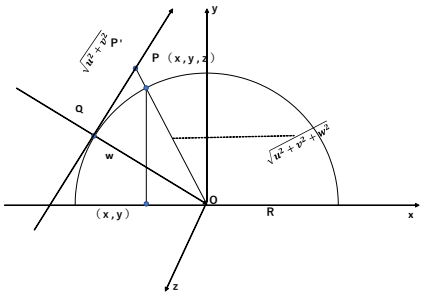
3.1.3 Load Distribution Mechanism

Figure 2 shows the load distribution mechanism of our developed system. The client software and the client side PIAX system are installed in the client. The different world broadcasting server software and the server side PIAX system is installed on the different world broadcasting servers. PIAX [3] is an open-source, Java-based platform middleware that enables efficient server resource searches using the resource search function of the overlay network. The PIAX systems used by the client and the different world broadcasting servers connect with each other via the overlay network. The client side PIAX system searches the overlay network according to the client software requests. The system selects a different world broadcasting server from the list, and then the client side PIAX system returns the IP address of the selected server and listens to the stated port number of the server software. The client software then establishes a connection with the different world broadcasting server and starts transmitting the video. New connections from the client are controlled based on the load state of the different world broadcasting server.

3.2 Extension of Different World Broadcasting System

Figure 3 shows an overview of our designed multi-viewpoint Internet live broadcasting system. As shown in the figure, our system converts the coordinates of videos from polar to rectangular when different world broadcasting servers execute processing. After that, the video images are delivered to viewers.

In this section, we first explain image conversion of multi-viewpoint videos and our design of ECA rules for multi-viewpoint videos. Then show some examples of ECA rules.



$$|\mathrm{OP'}|^2 = |\mathrm{QP'}|^2 + |\mathrm{QO}|^2 = (u^2 + v^2) + w^2$$

$$\mathrm{OP'}/\mathrm{OP} = \sqrt{u^2+v^2+w^2}\,/\,R$$

$$\mathrm{P'}\left(x',y',w'\right) = \frac{\sqrt{u^2+v^2+w^2}}{R}(x,y,z) \quad (1)$$

Figure 4: An image of the coordinate conversion

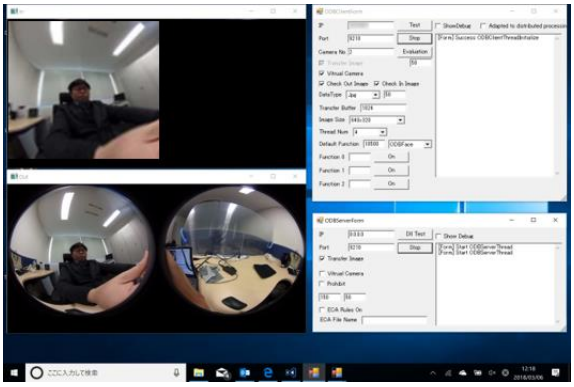


Figure 5: Server software and client software

3.2.1 Image Conversion of Multi-Viewpoint Videos

With omnidirectional cameras, it is not realistic to take dozens of omnidirectional images from a certain viewpoint and synthesize them on a computer to create a panoramic image. Instead, we create multi-viewpoint videos from panoramic images. The lower left part of Fig. 5 shows two panoramic images (front and back) for a multi-viewpoint video. These panoramic images were obtained from cameras using fisheye lenses. It is necessary to convert these images into a planar image. There are many methods that obtain wide images from car-mounted fisheye lenses and correct the distortion [12]. Figure 4 shows how to obtain a wide image from a panoramic image in our proposed system. As shown in this figure, the wide images are obtained by assuming an imaginary hemispherical border for the panoramic images. The converted wide image is shown in the upper left part of Fig. 5. The conversion transforms virtual hemispherical polar coordinates into rectangular coordinates using the equation (1). In our proposed system, the distributed processing of polar/rectangular coordinates is performed using a different world broadcasting server.

```
{
 "Rule A":{
        "eventname":"Set_effect",
        "condition":{
                    "name":"Num_Find_Object",
                    "object":"object1_haar.xml",
                    "Value":">=1"
        },
        "action":{
                    "name":"REQ_IP",
                    "IP_address":"192.168.0.5"
                }
        },
 "Rule B":{
        "eventname":"Set_effect",
        "condition":{
                    "name":" Spherical_coordinates_Convert",

        },
        "action":{
                    "name":"REQ_IP",
                    "IP_address":"192.168.0.6"
                }
        }
}
```

Figure 6: Examples of ECA rules

Table 1: Events in Communication

Event Name	Description
Receive_Data	Occurs when receives data.
Finish_Transmission	Occurs when data transmission finishes.
Computer_Request	Occurs when recommend server request.
Change_Server	Occurs when DWS server is busy

Table 2: Variables for Conditions in Communication

Variable Name	Description
Data[]	Received data
Transmission_Result	Result of transmission
Turn-around-avg	Turn around time average
T-around-avg-diff	Turn around time average previous differential

Table 3: Actions in Communication

Action Name	Description
Dispatch	Launch Dispatcher

3.2.2 Example of ECA Rules Set

In multi-viewpoint Internet live broadcasting services, the screen images differ according to the viewpoint selected by the user. Thus, the processes for adding effects are usually executed on the users' computers. On the other hand, general processes for Internet live broadcasting such as video encoding, video distribution are executed on the broadcaster's computer or the distribution servers. This means that processes for distributed multi-viewpoint

Internet live broadcasting systems have some types. We design three types for ECA rules. One is the effect type that is related to video effects. The viewers' computers are suitable for the execution of this type because they do not need to transmit video data to others. Other one is the communication type and the rules in this type is executed on the computers performing communications. The last one is the processing type. The DWB servers are suitable for the execution of this type because they execute these processes.

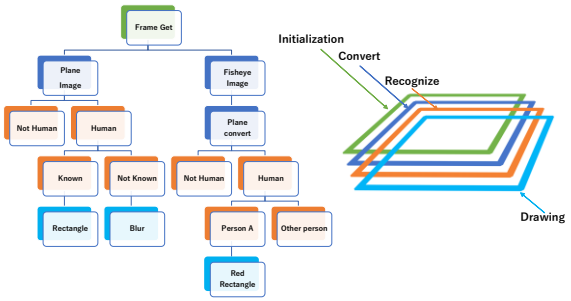


Figure 7: Examples of hierarchical ECA rules

3.2.3 Design of ECA Rules for Multi-Viewpoint Videos

Video effects have various procedures. For example, the face detection process is generally performed before the mosaic effect is applied to the detected face. The "Timer" or "Message" functions of the ECA rules in the proposed system can define such procedures. If the procedure is defined in order-dependent ECA rules, the system needs to execute the rules according to the sequence. Otherwise, if the ECA rules do not depend on the processing request, the system can execute the rules concurrently. This reduces the processing time compared with order-dependent ECA rules. In the current system, it is impossible to process ECA rules in parallel. The parallel processing of cloud computing services is left as a future task. Lists of events, conditions, and actions are described in our previous research [1]. We list some of them in Tables 1-3. Figure 6 shows an example of two ECA rules. In this example, the servers with IP addresses 192.168.0.5 and 6 are assigned as initial machines for the video processing requests from video recording terminals for the condition named "Num_Find_Object" and "Spherical_coordinates_Convert."

In cases where the processes of ECA rules have a sequence, the system should execute the processes in the order of the sequence. For example, Fig. 7 shows an example of the sequences of ECA rules. Some example sequences follow:

- 1. Is it a fisheye lens image? → Perform full spherical coordinate transformation → Human detection.
- 2. Are humans in the image → Who? → Match with a specific person → Blur is applied.
- 3. Are humans in the image → Is it a known person registered in the DB? → If it is an unregistered person, blur.

The ECA rules are classified into hierarchies of detection, conversion, inquiry, and pixel processing.

3.2.4 Implementation of Proposed System

We developed a distributed live Internet broadcasting system using Microsoft Azure as a cloud service. The different world broadcasting servers run on the VMs provided by

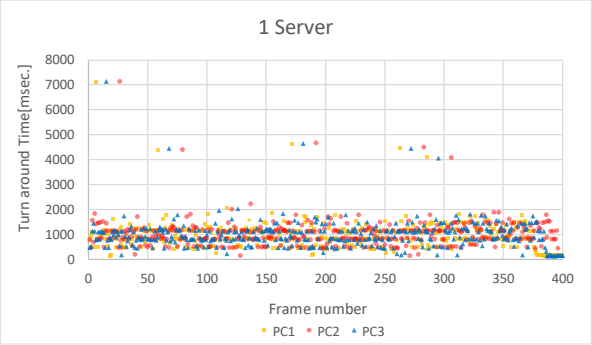


Figure 8: Turnaround times under one cloud server

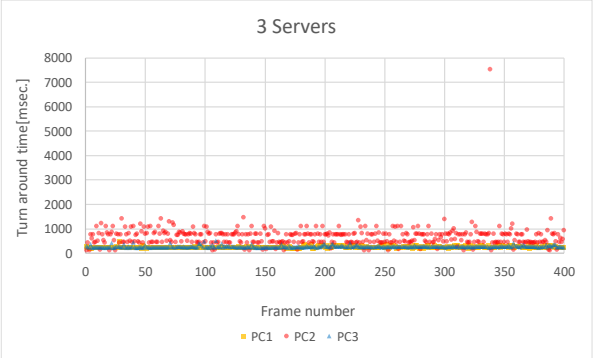


Figure 9: Turnaround times under three cloud servers

Table 4: Specifications of Microsoft Azure VMs

OS	Microsoft Windows Server 2016
Microsoft Azure Plan	Standalone Server Microsoft Corporation Virtual Machine x64-based PC
CPU	Intel E5-2697 v3 Equivalent 2.4GHz
Main memory	3.584GB

Table 5: Specifications of Client PCs

	Client PC1	Client PC2	Client PC3
OS	Microsoft Windows 10 Pro Version 1709,1511	Microsoft Windows 10 Pro Version 1709,1511	Microsoft Windows 10 Pro Version 1709,1511
CPU	Intel i7-7660U Equivalent 2.5GHz	Intel i5-6300U Equivalent 2.4GHz	Intel i3-4020Y Equivalent 1.5GHz
Main memory	8.00 GB	8.00 GB	4.00 GB

Azure. Each VM is logically connected through a virtual network, which is one of the services provided by Microsoft Azure. Figure 5 shows a screenshot of the server software and client software. When starting the process of adding video effects, it provides an interface of the different world broadcasting server software. Using the client software, we can visually check the result of applying the selected effects. The client software holds the IP address of different world broadcasting servers from which video processing can be requested. If the "Apply distributed processing" checkbox in the client software dialog box is selected, the client software requests the different world broadcasting server to execute the video processing specified by the pull-down menu of the initial IP address.

4 EXPERIMENTAL EVALUATION

We evaluated the performances of our implemented system.

4.1 Experimental System

In this evaluation, a different world broadcasting server was running on a VM provided by the Microsoft Azure service. Table 4 lists the specifications of the VM and OS. We used five different VMs for different world broadcasting servers. Open CV, parallelized by Intel's Parallel Computing Library TBB [13], was used as a library for executing video processing on different world broadcasting servers. The clients were PCs installed at Osaka University. Table 5 lists the specifications of the client PCs. We attached a full omnidirectional camera to only one PC. These PCs communicated with different world broadcasting servers via different home optical networks to avoid network congestion.

4.2 Evaluation Environment

We used a Theta S (RICHO Co., Ltd.) omnidirectional camera for evaluating our proposed system. Each video frame was encoded in JPEG format, transmitted, and received as a USB virtual camera. Image conversion and rule processing were realized by different world broadcasting. In the evaluation, we measured the time from the start of generating the original image data to the time that the processed image data were obtained. To confirm the efficiency of the proposed system, the video processing time, including the processing time of the ECA rule and the turnaround time, was measured as evaluation items.

This includes the following four items:

a) Preprocessing time during which the client receives data (same as the time from the end of reception of previous frame data to the start of the next frame data transmission).

b) Communication time, while different world broadcasting servers receive frame data.

c) Processing time on different world broadcasting servers.

d) Communication time during which the client receives frame data from a different world broadcasting server.

The video processing time is defined as the time from the start of the video processing, excluding the video data reception time, to the end of the processing.

To select an available different world broadcasting server, we used the PIAX overlay network described in subsection 3.1. When a different world broadcasting server overloads, the server sends a notification to the PIAX process on the server side and waits until the load has decreased. The turnaround time of the evaluation was measured in two cases. The first case is a concentrated case in which three clients request video processing from one of three different world broadcasting servers. The second case is a completely distributed scenario in which each of the client requests is sent to a different server.

As the video image processing for the evaluation experiments, face detections are executed on the processing servers after the coordinate conversions.

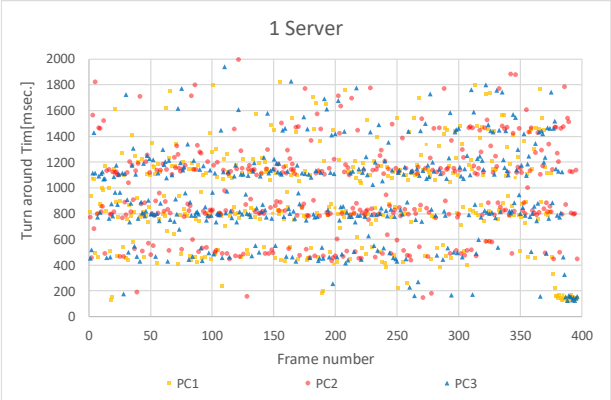


Figure 10: Turnaround time under one server

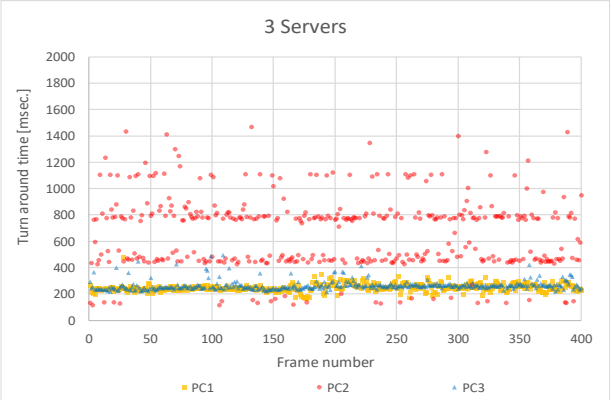


Figure 11: Turnaround time under three servers

4.3 Influence of the Number of Servers

Processes were assigned among the different world broadcasting servers based on ECA rules.

Figures 8 and 9 show the evaluation results of the turnaround time under the evaluation environment described in Section 4.1. The horizontal axis is the recorded frame number, and the vertical axis is the turnaround time. In Fig. 8, which shows the case where the load is concentrated on a single different world broadcasting server, the turnaround times gradually increase. Figure 9 shows the case where the image processing requests are distributed to three different world broadcasting servers. In this case, the turnaround time is less than 1500 ms, and the processing delay is around 7500 ms.

In the real environment of Microsoft Azure, the different world broadcasting server from which PC 2 requests image effect processing is a VM in the East Japan region. Therefore, there were variations in the communication route, and the turnaround time changes largely.

We also measured the turnaround time required to change the processing server to the recommended different world broadcasting server by PIAX. The average time required to process a query for determining the recommended different world broadcasting server was 16 ms.

As a result, we confirmed that the processing requests are allocated to the different world broadcasting servers based on the ECA rule, and the load is distributed. Moreover, we confirmed that the turnaround time might fluctuate, even for VMs with similar hardware performance, under the effects of communication delays.

4.4 Influence of Computational Load

We measured the turnaround times, changing the loads of DWB servers. To change the loads, we gradually increased the load caused by human face detection every one frame and measured the turnaround time. The results are shown in Figures 10 and 11.

The turnaround times were measured to determine whether the load is concentrated on one virtual server or not. The turnaround times were approximately 1000 ms in this experiment. We have measured the turnaround time for single-viewpoint videos in previous research [14]. The turnaround times were approximately 16 ms. Comparing with this result, the turnaround times for multi-viewpoint videos are longer because the data amount is larger.

5 DISCUSSION

5.1 Fluctuation of Turnaround Times

In our evaluation experiments, even when the calculation load was distributed among the three servers, video processing was sometimes concentrated on only one server. We used two networks for evaluation. (NTT's FLET'S Hikari and K-Opticom's eo light). When requests are concentrated on one different world broadcasting server, the turnaround time is relatively long. When requests from clients are concentrated on a single server, the processing load is distributed to the different world broadcasting server.

Moreover, the video processing involved detecting faces in the video using the specified effect described in the ECA rule. Results using the test rules are shown in Figures 10 and 11, which confirm the fluctuations in turnaround time among cloud computing service VMs. This is caused by actual server performance fluctuations due to differences in the cloud environment of the network distance. Such issues should be considered when the user configures the system.

5.2 Effectiveness of ECA Rules

In previous research, we implemented a distributed Internet live broadcasting system using a cloud service and evaluated its performance. In the installed system, the processing of additional effects is performed using the VM provided by the cloud service. By determining which processing should be allocated to the VM using the ECA rule, it is possible to flexibly change the computer that executes the processing. As a result of our previous evaluations, we confirmed that the turnaround time of the effect adding process could be reduced. As an ECA rule for load balancing to be given to client software, the effect selection made by the client software is set as an event, and a list of corresponding enquiries is set in advance as an IP address. As a result, the server selection is performed automatically and smoothly in the process of adding special video effects.

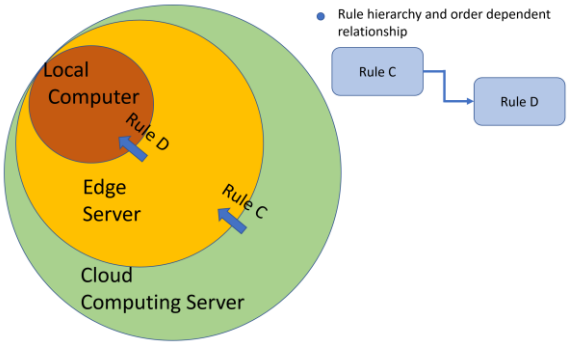


Figure 12: Image of the hierarchical rules

In cases where the processes of an ECA rule have sequences, the system should execute the processes in the order of the sequence. Otherwise, the system can execute them in parallel.

In this paper, we have proposed grouped three-stage rules. After the rules have been prepared, the location for their processing is selected to be either: (1) a local client, (2) edge computing, or (3) cloud computing. An image of the grouped rules is shown in Fig. 12, and the example rules are shown in Fig. 13. In this rule system, the different world broadcasting server that adds the video effects changes as the performance of the current server varies.

6 CONCLUSION

In this research, we have proposed and developed a multi-viewpoint distributed live Internet (different world) broadcasting system. One of the main research challenges for multi-viewpoint Internet live broadcasting systems is how to reduce the computational load required to add effects under different screen images. Our proposed system adopts ECA rules for executing video processes. In this research, we focused on which computer executes the rules, we classified the rules into three types. Each type has a suitable computer for its execution. By classifying ECA rules to these types, our proposed system establishes appropriate execution of rules for video processes.

In future work, we plan to exploit edge computing environments in which computers on the edge of the Internet can execute video processes. This could reduce the processing time because edge computers have short turnaround times.

ACKNOWLEDGMENT

This research was supported by a Grants-in-Aid for Scientific Research (C) numbered JP17K00146 and JP18K11316, and by I-O DATA Foundation.

REFERENCES

[1] S. Matsumoto, Y. Ishi, T. Yoshihisa, T. Kawakami, and Y. Teranishi, “Different Worlds Broadcasting: A Distributed Internet Live Broadcasting System with Video and Audio Effects,” in Proc. of IEEE International Conference on Advanced Information Networking and Applications (AINA), pp. 71-78 (2017).

```
{
“Rule C”:{
 “eventname”:” Computer request”,
    “condition”:{
      “name”:”turn-around”,
     “object”:”Piax-rq”
     “Value”:”>=20msec”
    },
    “action”:{
     “name”:”CHANGE_SERVER”,
     “IP_address”:”192.168.0.10”
    }
“Rule D”:{
“eventname”:” Computer request”,
    “condition”:{
      “name”:”turn-around-avg”,
     “object”:”t-around-avg-diff”
     “Value”:” >=1000msec”
    },
    “action”:{
     “name”:”CHANGE_SERVER”,
     “IP_address”:”127.0.0.1”
    }
}
```

Figure 13: Example of ECA rules

[2] S. Matsumoto, Y. Ishi, T. Yoshihisa, T. Kawakami, and Y. Teranishi, “A Design and Implementation of Distributed Internet Live Broadcasting Systems Enhanced by Cloud Computing Services,” in Proc. of International Workshop on Informatics (IWIN), pp. 111-118 (2017).

[3] M. Yoshida, T. Okuda, Y. Teranishi, K. Harumoto, and S. Shimojyo, “PIAX: A P2P Platform for Integration of Multi-overlay and Distributed Agent Mechanisms,” Transactions of Information Processing Society of Japan, Vol. 49, No. 1, pp. 402-413 (2008).

[4] Y. Gotoh, T. Yoshihisa, H. Taniguchi, and M. Kanazawa, “Brossom: a P2P Streaming System for Webcast,” Journal of Networking Technology, Vol. 2, No. 4, pp. 169-181 (2011).

[5] R. Roverso, R. Reale, S. El-Ansary, and S. Haridi, “Smooth-Cache 2.0: CDN-quality adaptive HTTP live streaming on peer-to-peer overlays,” in Proc. of ACM Multi-media Systems Conference (MMSys), pp. 61-72 (2015).

[6] J. Dai, Z. Chang, and G.S.H. Chan, “Delay optimization for multi-source multi-channel overlay live streaming,” in Proc. of the IEEE International Conference on Commu-nications (ICC), pp. 6959-6964 (2015).

[7] T. Yoshihisa and S. Nishio, “A division-based broadcasting method considering channel bandwidths for NVoD services,” IEEE Transactions on Broadcasting, Vol. 59, No. 1, pp. 62-71 (2013).

[8] D. Gibbon and L. Begaja, “Distributed processing for big data video analytics,” IEEE ComSoc MMTC E-Letter, Vol. 9, No. 3, pp. 29-31 (2014).

[9] W.-C. Ting, K.-H. Lu, C.-W. Lo, S.-H. Chang, and P.C. Liu, “Smart Video Hosting and Processing Platform for Internet-of-Things,” in Proc. of IEEE International Conference on Internet of Things (iThings), pp. 169-176 (2014).

[10] J. Bae, H. Bae, S. Kang, and Y. Kim, "Automatic Control of Workflow Processes Using ECA Rules," IEEE Transaction On Knowledge and Data Engineering, Vol. 16, No. 8, pp. 1010-1023 (2004).
[11] A. Frömmgen, R. Rehner, M.Lehn, and A. Buchmann, "Fossa: Using Genetic Programming to Learn ECA Rules for Adaptive Networking Applications," IEEE Conference on Local Computer Networks (LCN), pp.197-200 (2015).
[12] J. Jeong, H. Kim, B. Kim, and S. Cho, "Wide Rear Vehicle Recognition Using a Fisheye Lens Camera Image" IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), pp. 691-693 (2016).
[13] Thread Building Blocks, https://www.threadingbuildingblocks.org/, (referred October 1, 2017).
[14] S. Matsumoto, Y. Ishi, T. Yoshihisa, T. Kawakami, and Y. Teranishi, "A Distributed Internet Live Broadcasting System Enhanced by Cloud Computing Services," International Journal of Informatics Society (IJIS), Vol. 10, No. 1, pp. 21-29 (2018).

(Received April 14, 2019)

Satoru Matsumoto received his Diploma's degrees from Kyoto School of Computer Science, Japan, in 1990. He received his Master's degrees from Shinshu University, Japan, in 2004. From 1990 to 2004, he was a teacher in Kyoto School of Computer Science. From 2004 to 2007, he was Assistant Professor of The Kyoto College of Graduate Studies for informatics. From 2007 to 2010, he was Assistant Professor of Office of Society Academia Collabo-ration, Kyoto University. From 2010 to 2013, he was Assistant Professor of Research Institute for Economics & Business Administration, Kobe University. From 2015 to 2016, he was a specially appointed assistant professor of Cybermedia Center, Osaka University. From April 2016 to September 2016, he became a specially appointed researcher. Since November 2016, he became an assistant professor. His research interests include distributed processing systems, rule-based systems, and stream data processing. He is a member of IPSJ, IEICE, and IEEE

Tomoki Yoshihisa received the Bachelor's, Master's, and Doctor's degrees from Osaka University, Osaka, Japan, in 2002, 2003, 2005, respectively. Since 2005 to 2007, he was a research associate at Kyoto University. In January 2008, he joined the Cybermedia Center, Osaka University as an assistant professor and in March 2009, he became an associate professor. From April 2008 to August 2008, he was a visiting researcher at University of California, Irvine. His research interests include video-ondemand, broadcasting systems, and webcasts. He is a member of the IPSJ, IEICE, and IEEE.

Tomoya Kawakami received his B.E. degree from Kinki University in 2005 and his M.I. and Ph.D. degrees from Osaka University in 2007 and 2013, respectively. From 2007 to March 2013 and from July 2014 to March 2015, he was a specially appointed researcher at Osaka University. From April 2013 to June 2014, he was a Ph.D. researcher at Kobe University. Since April 2015, he has been a assistant professor at Nara Institute of Science and Technology. His research interests include distributed processing systems, rule-based systems, and stream data processing. He is a member of the IPSJ and IEEE.

Yuuichi Teranishi received his M.E. and Ph.D. degrees from Osaka University, Japan, in 1995 and 2004, respectively. From 1995 to 2004, he was engaged Nippon Telegraph and Tele-phone Corporation (NTT). From 2005 to 2007, he was a Lecturer of Cybermedia Center, Osaka University. From 2007 to 2011, He was an associate professor of Graduate School of Information Science and Technology, Osaka University. Since August 2011, He has been a research man-ager and project manager of National Institute of Information and Communications Technology (NICT). He received IPSJ Best Paper Award in 2011. His research interests include technologies for distributed network systems and applications. He is a member of IPSJ, IEICE, and IEEE.