Regular Paper

A Distributed Internet Live Broadcasting System Enhanced by Cloud Computing Services

Satoru Matsumoto[*,†], Yoshimasa Ishi[*], Tomoki Yoshihisa[*],
Tomoya Kawakami[*,**], 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
[†]smtsumoto@cmc.osaka-u.ac.jp

Abstract -Recently, live Internet broadcasting services such as USTREAM and TwitCasting have become popular. In such broadcasting, broadcasters record videos using the cameras on their laptops or smart phones (clients). The clients send the video data to the broadcast servers of live Internet broadcasting services, and the servers distribute the videos to the viewers. We can use high processing power computers provided by cloud computing services to improve the qualities of the videos or to show additional information. The processing power of the computers provided by cloud computing services often changes. Thus, by flexibly selecting the most useful computers, and flexibly determining the processing from the desired effects, clients can reduce the limitations on the types of effects. In this paper, we propose a 'different-world broadcasting system', a live Internet broadcasting system, enhanced by cloud computing services. In the proposed system, the policy for using cloud computing services, (i.e., the selection of the desired machine and effect specifications,) is described by ECA (Event, Condition, Action) rules, to enable flexible policy change. In this research, we implemented and evaluated the different-world broadcasting system enhanced by cloud computing services.

Keywords: Streaming Delivery, Multimedia, Internet Live Broadcasting, Video on Demand, Different Worlds Broadcasting

1 INTRODUCTION

In recent years, real-time video distribution (live Internet broadcasting) via the Internet, such as USTREAM and TwitCasting, has become popular. In live Internet broadcasting, broadcasters or viewers often add video effects to enhance visual experiences. By using high processing power clients, the broadcasters can add various types of effects to further enhance the visual experiences and/or show additional information. Meanwhile, high processing power computers, provided by cloud computing services such as Microsoft Azure, Amazon Web Services, and Google Cloud Platforms, have recently become available. For using these high power computers to add such effects, clients select the desired computer, send their recorded video data to the selected computer, and specify the desired effects (types, parameters, etc.)

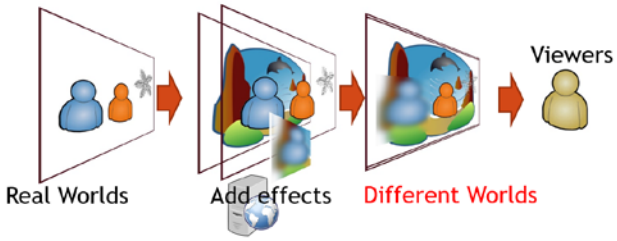

Figure 1: Different worlds broadcasting

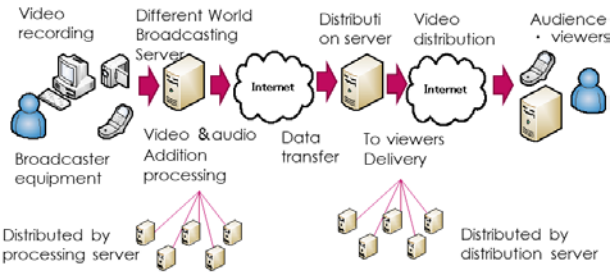

Figure 2: Overview of load distribution of effect addition processing

Though some distributed processing systems for live Internet broadcasting have been developed [1],[2], they do not focus on cloud computing services. Our research group has developed a distributed live Internet broadcasting system, called the different-world broadcasting system [1], an image of the system is shown in Fig. 1. In different-world broadcasting, effects such as blurring are added by a camera to videos recording the real-world, and videos are distributed to viewers. In the different-world broadcasting system, the different-world broadcasting server adds the video effect. Therefore, the processing load caused by the addition of the video effect, which has conventionally been done by the clients, is here distributed (Fig. 2). This system assumes that a different-world broadcasting server is a computer owned by a distributor, or a virtual computer provided by a cloud.

In our previous research [1], we developed a different-world broadcasting system using a computer owned by a distributor as the different-world broadcasting server. In this case, when many viewers request adding effects in a short

period of time, and when effects with a large load are requested, it takes long time to add the effects since the computer processes all of these requests. This causes delay and destabilization of the delivery frame rate. For this reason, the available effects in this system are limited, to light-load effects, in order to keep the delivery frame rate. This limitation decreases the quality of the user's different-world experience.

In this research, we assume the systems that use a cloud service virtual computer as the different-world broadcasting server that adds video effects. In general, a number of virtual machines are easily available in cloud services. Therefore, the use of multiple virtual machines as different-world broadcasting servers improve the speed to add effects, while distributing the load among different-world broadcasting servers and maintaining a stable delivery frame rate. In our previous research, we assumed the systems that use only a single computer. In this research, we implement a distributed live Internet broadcasting system using cloud service, and evaluate its performance. In the implemented system, effect addition is processed using the virtual computer provided by cloud services. When delegating processing to virtual computers, the performance of the processing computer is flexibly controlled by ECA rules. The evaluation results confirmed that the turnaround time of the effect addition can be reduced by using these rules.

The rest of the paper is organized as follows. We discuss related research in Section 2. In Section 3, we explain the design and the implementation of the proposed different-world broadcasting system using cloud service, and in Section 4, we evaluate its performance. We discuss the evaluation in Section 5, and conclude the research in Section 6.

2 RELATED RESEARCH

Various methods have been proposed for distributing the processing load. However, most of them require the construction of load distribution systems beforehand. In live Internet broadcasting, however, it is easy to commence live broadcasting, and the number of video distribution terminals is typically unspecified. Therefore, it is difficult to construct a load distribution system for a large number of video distribution terminals in advance. Several studies have been done on distributed video processing systems that use computers other than video recording terminals.

In MediaPaaS, designed as a cloud computing service in the PaaS (Platform as a Service) type for live broadcasting, it is possible to encode, re-encode, and distribute the video using the computer on the service provider side [2]. Different from MediaPaas, our proposed system implements, a load balancing system by using PIAX [9], a P2P agent platform.

In our proposed system, when the video recording terminal connects to a different-world broadcasting server, it interrupts the connection and distributes the load by randomly selecting the connection destination from some of different-world broadcasting servers. It has been experimentally confirmed that distributing the processing load over a number of different-world broadcasting servers and video recording terminals reduces video processing time. In this research, in order to be able to use more different-world broadcasting servers, we implement the proposed system, and evaluate the video processing time, using the virtual computers provided by a commercial cloud service.

The system implemented in [3] enables the viewing of recently recorded video (chasing playback) for live broadcasting performed by a P2P network. In this video distribution system proposed in this paper, broadcasters select adding video effects, one of the features of the system proposed in this research.

Several methods have been proposed for reducing the delay time of video distribution for live Internet broadcasting. In live broadcasting with SmoothCache 2.0 [4], it is possible to reduce the communication load and delay time of the video recording terminal, by caching video data from other peers and distributing it from the cached peers using the P2P network. Dai et al. Proposes a P2P network distribution route determination method which minimizes the delay time in [5]. In the HD method proposed, communication is reduced by simultaneously transmitting video data to multiple viewing terminals using one-to-many broadcast distribution combined with one-to-one communication in [6]. In the proposed system, these delay reduction methods can be applied when distributing videos, but our research differs in its design of the video processing system.

studies on systems that perform video processing on stored video data have been discussed. Gibbon et al. proposed a system that performs video processing by transferring video data shot by a camera to a computer with high processing power. Ting et al. proposed a system that stores images captured by low processing power computers (e.g., smartphones) directly in an external storage device such as cloud storage, without storing them in them [8]. These systems, however, target stored video data, and cannot be applied for live Internet broadcasting.

3 SYSTEM DESIGN AND IMPLEMENTATION

We applied a load distribution mechanism to the different-world broadcasting servers and video recording terminals of the different-world broadcasting system.

In this section, we explain the system design, the load distribution mechanism, the design of the description method used for dynamically allocating and executing the effect addition processing, and the implementation of these components.

3.1 Design Overview

In the different-world broadcasting system enhanced by cloud services in order to distribute the load of virtual machines, it is necessary to describe the processing to be assigned to each virtual machine. The types of processing can be explicitly described and allocated when the types do not change frequently. When they frequently change, the system dynamically allocates them according to the specific type of processing. In this research, processing is allocated using ECA rules, so that the allocations can be flexibly inscribed.

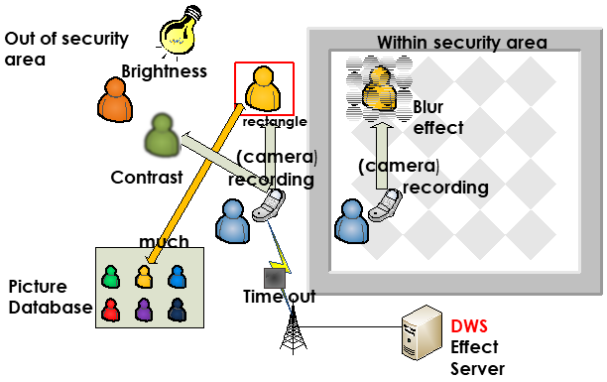

Figure 3: An image of video processes on different-world broadcasting systems

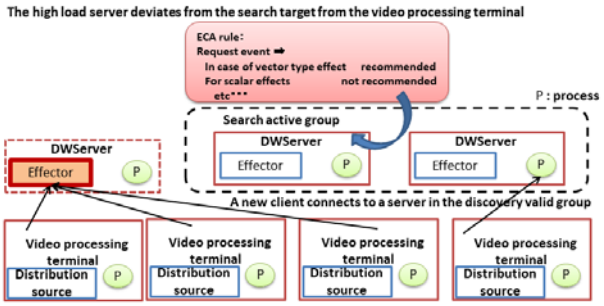

Figure 4: Overview of ECA rules application

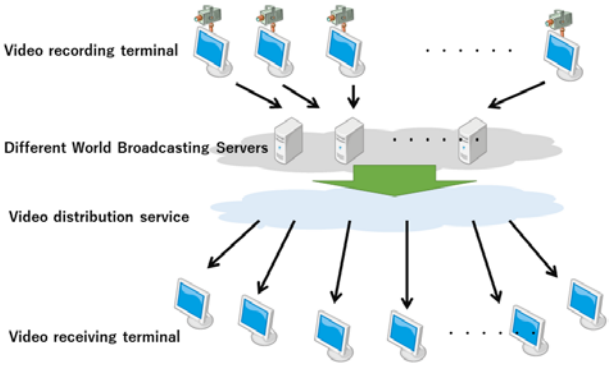

Figure 5: An image of video distribution

In our previous system [1], in order to reduce the operation load on broadcasters, event driven processing based on ECA rules is automatically executed. At the time of recording, each rule is automatically invoked by its specified event, and enlists different-world broadcasting servers to add visual and auditory effects. For example, to protect portrait rights, face authentication automatically detects specific persons in a database and adds mosaics effects on these faces when the individuals enter a given area. Also, the system can automatically adjust the brightness of the video or adjust the contrast according to the brightness of the image (Fig. 3). In our previous research, ECA rules were applied on the different-world broadcasting server side, mainly for the purpose of reducing the operation load on the broadcaster, as described above. In this research, we design a load balancing rule based on ECA rules, so as to avoid new video processing requests to different-world broadcasting servers in a heavy load state. The ECA rules are distributed from the ECA rule distribution server to each server (Fig. 4).

3.2 Overview of Load Balancing Mechanism

Figure 5 shows the environment assumed in this research. In our assumed environments, there are three types of terminals or servers: video recording terminals, different-world broadcasting servers, and video receiving terminals. The video recording terminal selects a different-world broadcasting server that can execute desired video effects, and transmits video effect libraries and the recorded videos. The different-world broadcasting server operated on the cloud service virtual machine executes the video processing to the video sent from the video recording terminal, according to the requests from the terminal. The video processed by the different-world broadcasting server is delivered to the video receiving terminals via the video distribution service. The video receiving terminal receives the processed video after selecting the server or the channel of the video distribution service.

In this research, focusing on how the video recording terminal selects a different-world broadcasting server, we consider load distribution for different-world broadcasting servers. In our previously proposed system, load distribution is achieved by connecting the video recording terminal via the load distribution mechanism, when connecting to the different-world broadcasting server. In this method, the load distribution mechanism selects a different-world broadcasting server that less video recording terminals use when it is necessary to switch to another different-world broadcasting server while the video is being transmitted. For example when switching to a different-world broadcasting server more suitable for a given scene or effect, the video recording terminal terminates the existing connection, which cannot be resumed without formal reconnection. For this reason, it is difficult to smoothly switch between different-world broadcasting servers while continuing the video distribution.

In addition, even when the load distribution mechanism changes the different-world broadcasting server according to the requests from video recording terminals, the mechanism relays the communication data as a simple byte stream. This means that the data structure (e.g., the boundary of the video frame) is ignored, and the video data may be damaged when the mechanism switches the servers. Therefore, in this research, the load distribution mechanism serves only to select the different-world broadcasting server, based on the requests of video recording terminals, and the video recording terminals connects directly to one of the different-world broadcasting servers as a result of these requests. In this way,

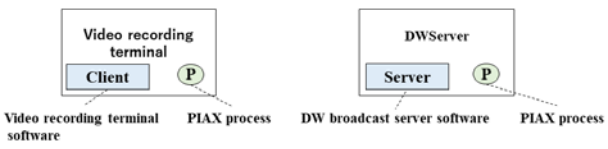

Figure 6: Internal systems of video recording terminals and different-world broadcasting servers

video recording terminals can switch different-world broadcasting servers at any time, even when the boundaries of the scene or video frames come.

3.3 System Architecture

Figure 6 shows the internal configurations of the video recording terminal and different-world broadcasting server. Video recording terminal software and a client-side PIAX system are installed to the video recording terminal. Different-world broadcasting server software and a server-side PIAX system are installed to the different-world broadcasting server. PIAX [9] is a Java-based platform middleware that enables serverless and efficient resource searches by utilizing the search function of the overlay network. PIAX is provided as open source software. The PIAX system in the video recording terminal and the different-world broadcasting server can be connected via their overlay network.

The client-side PIAX system on the video recording terminal searches the overlay network in accordance with the request of the video recording terminal software. The system selects a different-world broadcasting server from a host of different-world broadcasting servers, and returns the IP address of the selected server and the listen port number of the server software. Based on the response, the video recording terminal software establishes a connection with the different-world broadcasting server software and starts sending the video. JSON-RPC is used directly over TCP for communication between the video recording terminal software and client-side PIAX system.

The server-side PIAX system on the different-world broadcasting server determines its search state based on the instructions from the different-world broadcasting server software. If the different-world broadcasting server is regarded as in a high-load state itself, it connects with the control port of the server-side PIAX system on localhost, and sends a ‘leave’ request command for the search, which prevents the different-world broadcasting server from receiving further connection from the video recording terminal. If the load on the different-world broadcasting server lessens, and the server gets a margin to accept a new connection, the broadcasting server reconnects with the control port of the server-side PIAX system on localhost, and sends a ‘release leave’ request command. This command, the server-side PIAX system again enters a state of search acceptance from the video recording terminal.

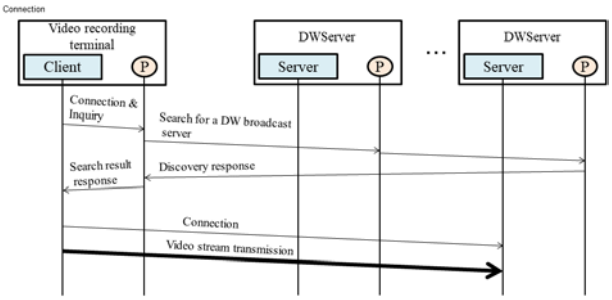

Figure 7: Data flow and timing chart of communication procedure


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


Figure 8: An example ECA rules

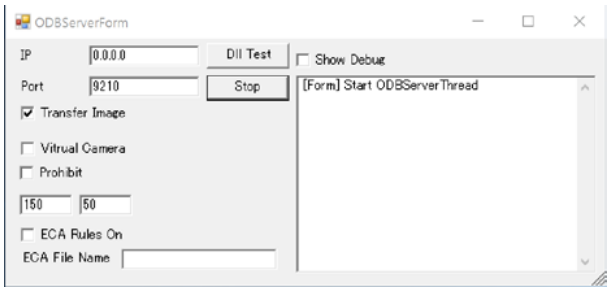

Figure 9: A screen shot of different-world broadcasting server software

With the above operation, overload of the different-world broadcasting server software can be avoided since the new connection from the video recording terminal is controlled based on the load state of the different-world broadcasting server. Figure 7 shows the data flow and timing chart of the communication procedure.

3.4 Designing ECA Rules

The lists of events, conditions, and actions are explained in our previous paper [1]. In addition to the list, we add one event and one action to enhance the system for using cloud computing services. “Set_effect” events occur when new video effects are used in the system. “Req_IP” actions initially set the IP address of the different-world broadcasting server that adds the video effect designated in the condition part. In case when the designated server is on heavy loads, the address is automatically changed by PIAX. Figure 8 shows a concrete ECA rules set. In the example, the video recording terminal initially requires the video processing named "Num_Find_Object" or "Bluer" to the server of that IP address is 192.168.0.5 or 6.

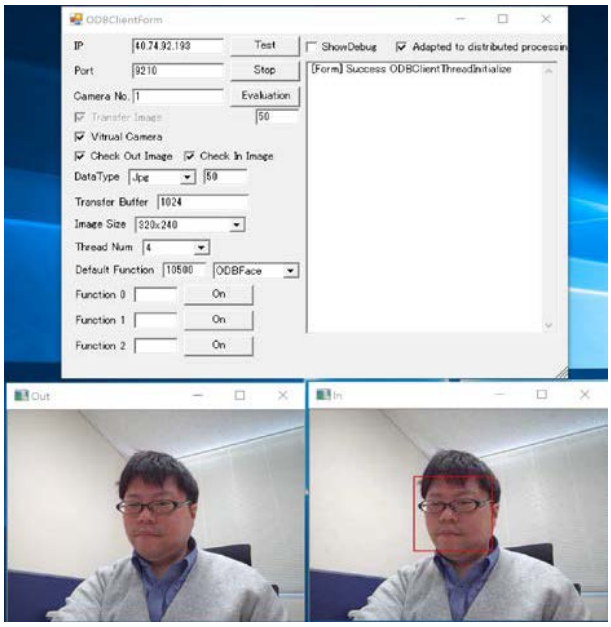

Figure 10: A screen shot of different-world broadcasting client software

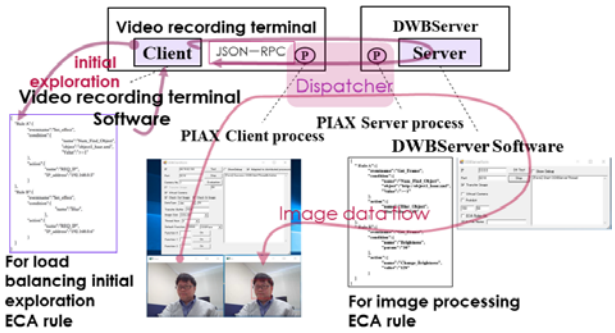

Figure 11: Processing flows of our implemented system

3.5 Implementation

We used Microsoft Azure as a cloud service, to deploy the different-world broadcasting system. The different-world broadcasting server works on virtual machines provided by the Azure service. Each virtual machine is logically connected by a Virtual Network (VNet), one of the services provided by Microsoft Azure. Figure 9 shows the user interface of the different-world broadcasting server software when initiating the video effect addition process. Figure 10 shows the user interface of the video recording terminal software that distributes the video on the video recording terminal. It can visually recognize the result of applying the selected effects.

If the 'ECA Rules On' check box in the different-world broadcasting server dialog box is checked, the image effect is automatically added, based on the different-world broadcasting server ECA rule without the operations of the user.

The IP address of the different-world broadcasting server for requesting the video processing is held in the video recording terminal software. The terminal software uses the IP address to request the video processing designated in the pull-down menu to the different-world broadcasting server if the 'Apply Distributed Processing' check box in the video recording terminal software dialog box is checked, the initial video processing request is sent to the different-world broadcasting server via the load distribution mechanism, based on the ECA rules held by the client software. The processing request is sent to a different-world broadcasting server with a low processing load. In this case, the ECA rule inscribes the initial IP address to request its execution to the load distribution mechanism. Figure 11 shows the processing flows of our implemented system in this research.

4 EVALUATION

We evaluate the effectiveness of the proposed method using an implementation system built on the virtual machines provided by the Microsoft Azure service. The evaluation method, environment, and results are described below.

4.1 Evaluation Method

To evaluate the efficiency of the proposed system, we focused on the video processing time and turnaround time, including the processing time of the ECA rule, as evaluation indices. We made a comparative evaluation of the turnaround time when (a) the video effect processing requests were concentrated on the video effect processing server without employing the ECA rule; and (b) the video effect processing requests were not concentrated, and the ECA rule was employed. For the evaluation, we assigned multiple computers with the same performance parameters on the cloud service. For the selection of available different-broadcasting servers, we used the PIAX overlay network explained in the previous section. When a different-world broadcasting server is overloaded, the server sent a notification to the server-side PIAX process, and wait until the load becomes low. We measured the turnaround time in two cases. one is the concentrated case, in which four video recording terminals made requests to one out of five different-world broadcasting servers. This corresponds to the situation (a). The other is the non-concentrated case, in which each of the four video recording terminals made requests to a different-world broadcasting server service. This corresponds to the situation (b).

Since the main focus of the evaluation is processing load distribution, live broadcasting software is not used in this evaluation. In the evaluation, face detection video processing, which detects the face of a person in a given video, is employed, with the video effect inscribed in the ECA rule. For evaluation under different computational loads, we gradually increased the load caused by human face detection and measured the turnaround time. The time taken to transmit and receive frame data was defined as the turnaround time. This includes the following four items.

Table Ⅰ: Specifications of virtual machine (different world broadcasting server)

Component	Performance
OS	Microsoft Windows Server 2016 Datacenter
Microsoft Azure (Virtual server service plan)	Standalone Server Microsoft Corporation Virtual Machine x64-based PC
CPU	Intel E5-2697 v3 Equivalent 2.4 GHz
Main memory	3,584 MB

Table Ⅱ: Specifications of video recording terminals

Component		Performance
OS	Client 1～4	Microsoft Windows 10 Pro Build 14393
CPU	Client1	Intel i7-6500U Equivalent 2.4 GHz
	Client2	Intel i7 2677M 1.8 GHz
	Client3	Intel i7 4650U 1.70 GHz
	Client4	Intel i3-4020Y 1.50 GHz
Main memory	Client1	8,118 MB
	Client2	3,999 MB
	Client3	8,097 MB
	Client4	3,999 MB

- The preprocessing time during which the client obtains the video data (equal to the time from the end of the previous frame data reception to the beginning of the next frame data transmission).
- The communication time while the different-world broadcasting server receives the frame data.
- The processing time on the different-world broadcasting server.
- The communication time during which the client receives the frame data from the different-world broadcasting server. The video processing time is defined as the time from the start of the video processing except for the image data reception time to the end of processing.

4.2 Evaluation Environment

In the evaluation, the different-world broadcasting server worked on the virtual machines provided by the Microsoft Azure service. Table 1 shows the specifications and OS of the virtual machines. We used five different virtual machines for the different-world broadcasting servers. Open CV, parallelized by Intel's parallel computing library TBB [10], was used as the library for executing video processing on the different-world broadcasting server. The video recording terminal is a PC installed in Osaka University. Table 2 shows the specifications of the video recording terminal PCs. 4 PCs were used, and the video recording terminal software were started two processes at a time, thereby operating the video recording terminal software of a total of four processes. These PCs accessed different world broadcasting servers via different home optical line networks to avoid networks (FLET'S Hikari by NTT, eo HIKARI by K-Opticom) congestion.

4.3 Evaluation Results

Figures 12 and 13 show the evaluation results for the turnaround time under the evaluation environment described in Section 4.2. The horizontal axis shows the recorded frame number, and the vertical axis the turnaround time in milliseconds. In Fig. 14 where the load is concentrated on a single different-world broadcasting server, the turnaround time increases in a nearly linear manner.

In Fig. 15, where the video effect processing request was distributed among four different-world broadcasting servers, the turnaround time increased in the latter half, since the calculation load increased. There were no sharp increases in the turnaround time compared with the result in Fig. 14. In addition, though the virtual machines of the Azure service shared the same performance parameters, the turnaround times fluctuated due to the communication delay between the video recording terminal and the different-world broadcasting servers, or other reasons. Over the first half of the frame numbers, the turnaround time in Fig. 15 gradually increased compared with the result in Fig. 14. This indicates that the concentration of video processing is suppressed. We also measured the turnaround time required to inquire the recommended different-world broadcasting server. The average value for a single inquiry, over 50 trials, was 16.28 msec.

The results revealed that the processing requests were allocated among different-world broadcasting servers based on the ECA rules, and the load could be distributed. In addition, we confirmed that, the turnaround times could fluctuate even if the hardware performance of the virtual machines was equal due to the influence of communication delay, etc.

5 DISSCUSSION

5.1 Evaluation Results

In the evaluation, the video processing sometimes concentrated on one server, even when the computational load was distributed to four servers. Moreover, the processing time increased when face was detected because we used actual video images and so it is difficult to grasp the overall pattern. Therefore, we draw the complimentary curves in the figures.

We used two networks for the evaluation. (FLET'S HIKARI by NTT, and eo HIKARI by K-Opticom). In case that the requests from video recording terminals were going to be concentrated on a single different-world broadcasting server, processing load was distributed to each different-world broadcasting server. Compared to the access time, the turnaround time was relatively long.

In the evaluation, video processing was detecting the face of a person in the video, with the designated effect inscribed in the ECA rule. As explained in the previous subsection, we gradually increased the load caused by human face detection and measured the turnaround time. The time needed to transmit and receive the frame data was compared by the turnaround time. The results are shown in Figs. 14 and 15.

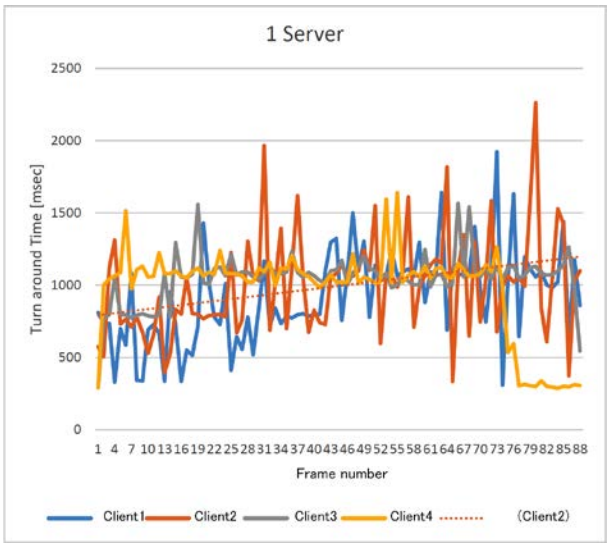

Figure 12: Turnaround times under one different-world broadcasting server

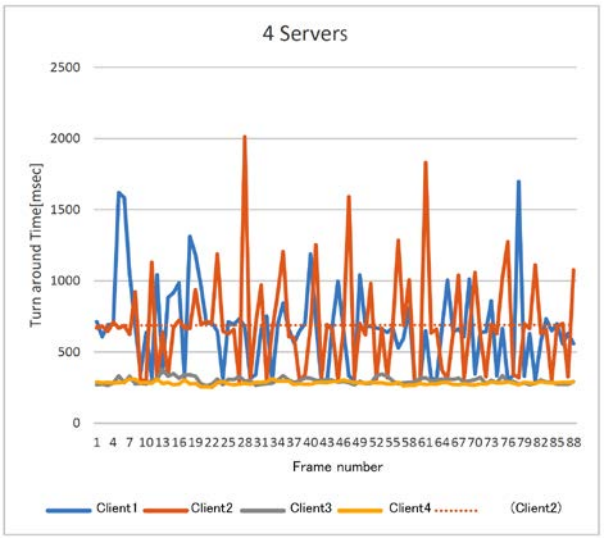

Figure 13: Turnaround times under four different-world broadcasting servers

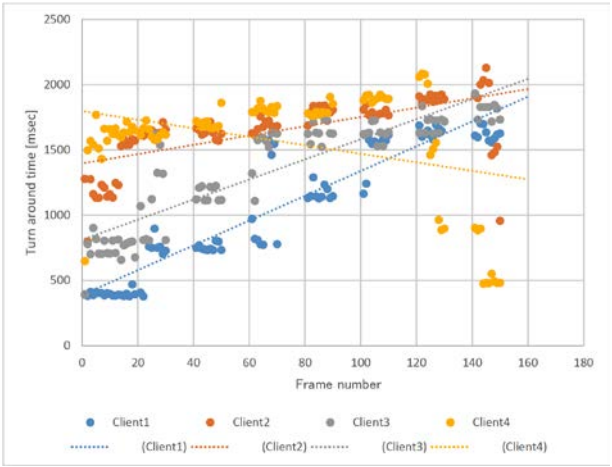

Figure 14: Turnaround times under one external different-world broadcasting server

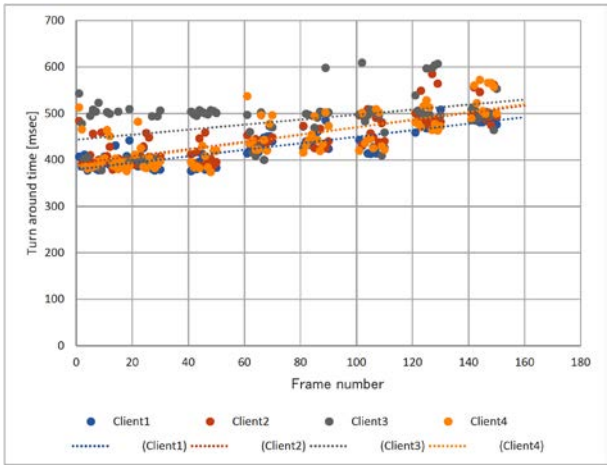

Figure 15: Turnaround times under four external different-world broadcasting servers

Fluctuations of the turnaround time were confirmed among the virtual machines of the cloud computing services. This arised by the factors such as fluctuations in the performance of the real servers in the cloud and differences in the network distances.

It is better for the users to consider these issues when they configure the system. Also, further improvement is required for ECA rule definitions.

5.2 ECA Rules Processing

Some video effects have sequences. For example, face detection processes are generally executed before the mosaic effects on the detected faces. `Timer' or `Message' functions of the ECA rules in our proposed system can define such sequences. When the sequences are defined in the ECA rules and there are order dependencies in the ECA rules, the system should execute the rules along with the order. Otherwise, when the ECA rules are order independent, the system can execute the rules in parallel and the processing time can be reduced compared with the ECA rules with order dependencies. Current system cannot process ECA rules in parallel and their parallel processing on cloud computing services is our future work.

6 CONCLUSION

In this research, we implemented and evaluated a distributed live Internet (different-world) broadcasting system using cloud services. By inscribing the processing assignment in ECA rules, it was possible to flexibly assign the virtual machine performance for video processing. In the implemented system, the PIAX platform was used for searching and communicating with the virtual machines, enabling continuous allocation of the video processing while live Internet

broadcasting, even if the number of virtual machines changes. The system evaluation confirmed that the video processing turnaround time could be reduced by using our proposed system.

In the future, we plan to allocate processes with consideration of the hardware performance of different-world broadcasting servers, and distribute light weight processes on video recording terminals.

ACKNOWLEDGMENT

This research was supported by JSPS KAKENHI Grant Number JP15H02702 and JP17K00146, and part of this research is the result of NICT & Osaka University joint research "research and development of advanced network platform technology for large scale distributed computing".

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 31st IEEE International Conference on Advanced Information Networking and Applications (AINA 2017), pp. 71-78 (2017).

[2] B. Cheng, "MediaPaaS: A cloud-based media processing platform for elastic live broadcasting," Proceedings of the 7th IEEE International Conference on Cloud Computing (CLOUD 2014), pp. 713-720 (2014).

[3] 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).

[4] 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," Proceedings of the 6th ACM Multi-media Systems Conference (MMSys 2015), pp. 61-72 (2015).

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

[6] 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).

[7] 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).

[8] 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," Proceedings of the 2014 IEEE International Conference on Internet of Things (iThings 2014), pp. 169-176 (2014).

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

[10] Thread Building Blocks, https://www.threadingbuildingblocks.org/, (referred October 1, 2017).

(Received October 20, 2017)

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

Yoshimasa Ishi received his B.E. degrees from Kyoto Institute of Technology, Japan, in 2004 and his M.I. degrees from Osaka University, Japan, in 2006, respectively. From 2006 to 2008, and from 2012 to 2015, he was a specially appointed researcher of Cybermedia Center, Osaka University. From 2008 to 2012, he was a specially appointed researcher of Graduate School of Information Science and Technology, Osaka University. Since January 2017, he has been a specially appointed researcher of Institute for Datability Science, Osaka University. His research interests include technologies for distributed network systems and its development. He is a member of the IPSJ.J.

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.