P2P LOAD DISTRIBUTION OF CONTENT DISTRIBUTION IN EDGE COMPUTING

Takuya Fukuda, Noriko Matsumoto and Norihiko Yoshida
Department of Information and Computer Sciences, Saitama University
255 Shimo-Ohkubo, Saitama 338-8570, Japan

ABSTRACT

In recent years, the number of Internet of Things (IoT) devices/sensors and mobile devices has been rapidly increasing. Considering the increase of generated data and network traffic by them, problems such as cloud computing resource and network performance shortage will occur in the future. There are also problems such as transmission delay due to the cloud and the physical distance between them. To support these problems, a new computing paradigm named "edge computing" has been introduced. Edge computing is an architecture or concept that extends the traditional cloud to the edge of the network and physically distributes cloud-based facility close to the end user. The cloud-based facility is called the "cloudlet." In this paper, we focused on content distribution by edge computing. There were inefficiencies in the content distribution, so it is found that the load would concentrate on the cloud. In order to solve this problem, we propose a load distribution method using peer-to-peer (P2P) between cloud and edge. We evaluate the method through simulation experiments.

KEYWORDS

Edge Computing, Content Distribution, Peer-to-Peer, Internet of Things

1. INTRODUCTION

The cloud is one of services used by IT infrastructure such as computing resources and communication devices. Conventionally, when we used the IT infrastructure, the user had to estimate in advance the required IT infrastructure and procure it in advance. However, in fact, there was a problem that surplus and shortage occurred in the prepared IT infrastructure. On the other hand, in the cloud, users can reduce such problems by enlarging or shrinking the IT infrastructure as necessary.

In recent years, not only desktop PCs and notebook PCs, but also Internet of Things (IoT) which connects various "things" such as automobiles, household electric appliances and medical equipments to the Internet attract attention. Automatic adjustment of household electrical appliances and medical devices are being carried out by collecting a large amount of data from things (IoT device) connected to the Internet and analyzing and utilizing the data. Also, with the spread of mobile devices such as smartphones and tablets, contents are getting diverse such as movies and dynamic contents. Today, cloud computing is widely used as an IT infrastructure for operating these services. However, considering the increase in traffic, the burden on the bandwidth, and content processing load and capacity increase due to the popularization of IoT and diversification of content, it can be predicted that the operation in the cloud will be limited in the future. In addition, real-time and latency-sensitive computation service requests to be responded by a distant Cloud datacenters often have problems such as large communication delay, network apparel, and poor quality of service.

To solve these problems, a new paradigm called "edge computing" attracts attention [Garcia et al, 2015]. Edge computing is a new architecture that extends cloud computing to the edge of the network and physically distributes it close to the end user. Edge computing solves the transmission delay caused by the physical distance between the device and the cloud by performing processable data and services on the edge and it aims to utilize more data and extract value. The processing platform placed on the edge is called "cloudlet"[Satyanarayanan et al, 2009]. In addition, since some processing is performed at the edge part, the load of the cloud is distributed, and it is possible to solve the lack of computing resources and insufficient network performance due to load concentration on the cloud.

There are several methods of using edge computing, one of which is “content distribution” that supports content distribution. We found problems with the content distribution. Many cloudlets request data required for content distribution to the cloud when a huge number of users request a specific application provided from the edge. The cloud sends the requested data for each request from the cloudlet. This is an inefficient distribution that the cloud transmits the same data.

In this paper, we plan to utilize the peer-to-peer (P2P) technology in the cloud and edge in order to make such distribution more efficient and realize load distribution. We built a simulator which makes edge computing on a single computer, and we verified the effectiveness of our proposal through some experiments. The results confirmed that our proposal improves content distribution in edge computing and realizes load distribution on cloud and edge.

This paper is organized as follows: Initially, we explain edge computing in Section 2. we summarize some related works in Section 3, and we describe our proposed method in Section 4. In Section 5, we explain simulation of our proposed method. Subsequently, we evaluate our method through some simulation experiments in Section 5. Finally, Section 6 contains some concluding remarks and future work.

2. EDGE COMPUTING

Edge computing is a new architecture that extends cloud computing to the edge of the network and physically distributes cloud-based computing, storage and networking facilities close to the end user.

Currently, user applications that access the public cloud do so through the core network to reach the cloud data center. With the introduction of cloud-based computing, storage and networking capacity at the edge of the network, edge computing can provide computing services and storage services. The cloud-based facility with such capabilities is called the “cloudlet.” Figure 1 illustrates the cloudlets concept within the hierarchical infrastructure of the edge computing.

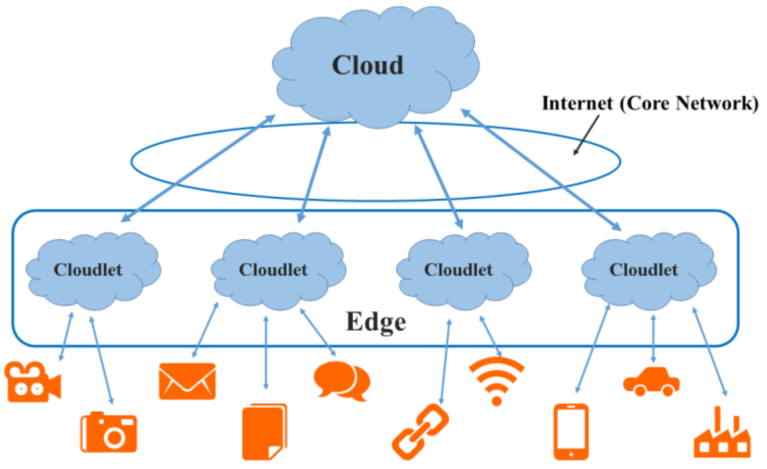

Figure 1. Edge Computing

Some typical usages [Steven , 2016] of edge computing are described below.

1) Content Distribution: When distributing content from the cloud to the user, the edge holds the cache of the content. As a result, problems such as load concentration on the cloud and compression of network bandwidth around the cloud are avoided, and responsiveness and convenience are improved for the user. Specifically, the edge caches the dynamic object generation code and application data in its own storage. Users are requesting contents directly by accessing the cloud, but in fact the user is accessing the edge distributing contents to users. Therefore, when a request for content once distributed comes to the edge again, it is not necessary to inquire to the cloud each time. In this way, by distributing the content from the edge rather than directly receiving the content distribution from the cloud, the communication range can be completed only between the user and the edge. Also, instead of the long distance cloud, the short distance edge processes, so the user can acquire the content at high speed.

2) Data Aggregation: The edge functions as an intermediate processing platform between the cloud and the end user. By handling a large amount of data sent from the end user in advance by the edge instead of the cloud, processing at the edge reduces the processing load of the cloud. In addition, since the application is executed at the edge closer to the end user, it is possible to respond quickly to the situation of the device and the surrounding environment change.
3) On-premise Applications: Real-time latency-sensitive application provision is realized by running applications on edges closer to the end user. By providing the service from the edge, it is possible to solve the transmission delay due to the physical distance from the cloud. In recent years, services with real-time nature such as medical care, AR and games are increasing, so it is considered that provision of services by On-premise Applications is important.

3. RELATED WORK

There are many studies combining cloud computing and P2P technology [Zhang et al, 2012][Wu et al, 2014] and cloud-based content distribution methods [Sahoo et al, 2016]. In this section we will summarize works related specifically to our proposed method: PROOFS (P2P method used in FCAN), FCAN, Virtual Machine Deployment for Inter-Cloud.

3.1 Proofs

P2P Randomized Overlays to Obviate Flash-crowds Symptoms (PROOFS) [Stavrou et al, 2004] is a P2P method devised for scalable distribution under flash-crowds. Flash-crowds is that a large number of unexpected and unpredictable user access concentrate on specific contents, delayed provision of contents, unresponsive state.

PROOFS consists of two protocols. The first forms and maintains a network overlay, which is an ongoing, randomized process. The second performs a series of randomized, scoped searches for contents atop the overlay formed by the first protocol.

Details of the two protocols are described below.

1. Nodes continually perform what is called a shuffle operation. The shuffle is an exchange of a subset of neighbors between a pair of clients and can be initiated by any client. The client C1 that initiates a shuffle chooses a subset of neighbors of size c that is no greater than its current number of neighbors. It selects one neighbor, C2 from this subset and contacts that neighbor to participate in the shuffle. C1 sends the subset of neighbors it selected with C2 removed from the subset and C1 added. If C2 accepts C1's shuffle, it selects a subset of neighbors from its list of neighbors and forwards this subset to C1. Upon receiving each other's subsets of neighbors, C1 and C2 update their respective neighbor sets by including the set of neighbors sent to them.
2. The node creates a search packet for searching the desired contents. Then, the search packet is transmitted to the node randomly selected from the neighbor list. The search packet has parameters of Object, TTL and fanout.

Shuffle makes the neighbor list of all clients (nodes) participating in the overlay essentially random. Once a random state is reached, scoped searches for objects can be performed atop the overlay. Via results and simulation, PROOFS claimed to be nature robust to the overlay partitioning, peer dynamic joining/leaving and limiting participations in the system.

3.2 FCAN

The Flash Crowds Alleviation Network (FCAN) [Atajanov et al, 2007] is a load distribution method of content distribution used in a cache proxy deployed between a server and a client. Considering the short duration and unpredictability of ash crowds, FCAN invokes the overlay only when a server is overwhelmed by a large amount of requests, and reorganizes the overlay when necessary. This overlay is made using PROOFS. While FCAN is running, cache proxies that store contents distribute them to clients instead of servers.

A member server and member cache proxies, both of which comprise FCAN, do little more than what normal ones do. When detecting the predictor of Flash-crowds (increase in traffic load), the member server detects the predictor of Flash-crowds. All subsequent client requests are routed to this overlay by DNS-based redirection. If the subset of proxies is not large enough to handle the amount of requests, new proxies are invited, and the overlay is enlarged. When the ash crowd declines, some proxies leave, so that the overlay shrinks and is eventually released.

3.3 Virtual Machine Deployment for Inter-Cloud

Conventional studies combining cloud and P2P connect P2P network with node as device to cloud.On the other hand, this study is a P2P network with nodes as cloudlets. This related study [Maeno et al, 2015] is a study to deploy a VM on the cloud to speed up. When providing a service from the cloud, it is necessary to deploy a VM that provides the service. Especially when the same service is provided from each cloud, a similar VM is deployed in each cloud. In such a case, it takes time to deploy a VM in which applications necessary for providing the service are built in advance or to transfer/register the disk image of the VM to each cloud as necessary. The method in this related study shortens the VM deploying time by eliminating such a process when deploying a VM in clouds.

This method uses a configuration management tool to speed up a deploying time of a VM. A configuration management tool (i.e., Chef, Puppet, Ansible and others) automatically configures environments such as middleware and applications, and changes the state of the computer according to the contents described in the configuration file.

Specific design is done by the following procedure.

1. Set up a disk image with a minimum configuration OS installed in clouds.
2. Activate a VM on clouds by using the disk image.
3. The configuration management tool prepares necessary software and contents for a service, and sets the VM in a state that can provide a service.

By such a procedure, the time for transferring and registering a disk image created in advance is shortened. Therefore, high-speed VM deploying on clouds is realized.

4. PROPOSED METHOD

There are many content distribution methods in cloud computing environments have been devised, but few studies have considered cloudlet (edge). Our method is based on cloudlet. For content distribution in edge computing, there is inefficient content distribution that cloud transmits data for each request to each cloudlet that has requested the same data. We propose to apply FCAN to content distribution in edge computing.

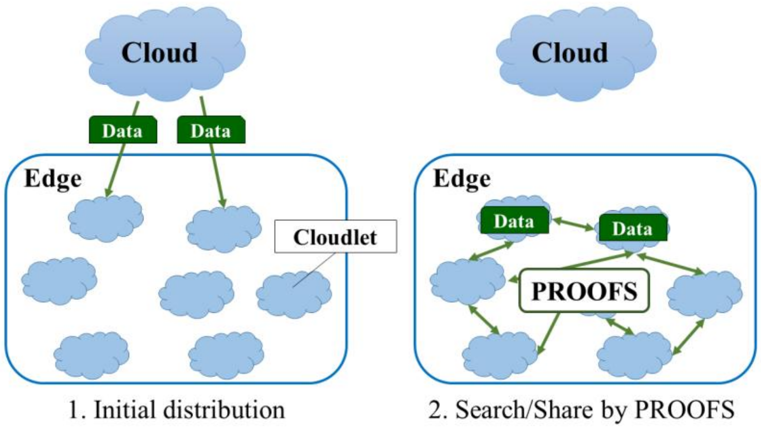

Figure 2. Proposed Method

The reason for applying FCAN is that the network topology of FCAN and Edge Computing are very similar, namely “server = cloud”, “cache proxy = cloudlet” and “client = device (end user.” In addition, because FCAN is a past studies of our laboratory, we fully understand the content and effect of FCAN's proposed method. Specifically, we devised using the P2P method PROOFS used in FCAN between cloudlets that make up the edge. The procedure of the proposed (Figure 2) method is as follows.

1) Initial distribution
Cloud transmits contents to several cloudlets. After transmission is complete, the cloud notifies managed cloudlets to start "Search/Share by PROOFS."
2) Search/Share by PROOFS
Each cloudlet uses the overlay formed between cloudlets by PROOFS and searches and shares data using P2P.

According to the above procedure, data required for content distribution is shared among cloudlets by FROOFS, and the cloudlet distributes contents to clients. In addition, PROOFS requires a bootstrap server that manages the neighbor list of clients participating in the overlay. The bootstrap server only manages the transmission of the neighbor list and participation/leaving to the overlay. This process does not perform particularly expensive processing. Therefore, in this study, we decided that the cloud is responsible for the bootstrap server.

Our proposed method differs from FCAN in terms of the type of transmitted data. FCAN sends/receives static data which is unchanged by the user such as CSS file, image file, HTML file, and so on. On the other hand, our proposed method sends/receives dynamic data which is changed by users such as VM, process and script file. In addition, if dynamic content requires a VM, it is necessary to send VM disk images and deploy VM. We assumed dynamic content as a VM and decided to use the proposed method for Virtual Machine Deployment for Inter-Cloud. Therefore, our proposed method improves the efficiency of content distribution and distributes the load between cloud and edge by using PROOFS between the cloudlets that compose the edge.

5. SIMULATION

In order to confirm the effectiveness of our proposed method, we simulated the proposed method using a program created in Java. In the simulation, a virtual edge computing environment was deployed. Several parameters need to be defined so that PROOFS is used in simulation. Therefore, prior to the evaluation of the proposed method, we conducted experiments on parameter selection and selected parameter values.

5.1 Edge Computing Environment

Since our proposed method is used between cloud and edge, we generated cloud and several cloudlets in simulation. The parameters used for the simulation of edge computing environment are shown in Table 1. Parameter values are set from related study and empirical rules, so there are few implementations of edge computing.

Table 1. Edge Computing Environment

Parameter	Cache Proxy
Number of Clouds	1
Number of Cloudlets	20
Cloud bandwidth	1000 [KB/s]
Cloudlet bandwidth	1000 [KB/s]
Data capacity	300 [MB]
VM deployment time	300 [Cycle]

5.2 Parameter Selection of PROOFS

It is necessary to determine the following parameter values in order to use PROOFS.

1. neighborhood size: A size of neighbor list of each cloudlet.
2. subset size: A size of subset used for shuffle operation.
3. Fanout: A value that indicates to how many neighbors a cloudlet should forward a search packet.
4. shuffle operation frequency: A value that indicates to how many search packets is received should be executed shuffle operation.

Based on the parameters described in the PROOFS and FCAN papers and some of experiments (for the experiment environment, use the environment shown in Table 1) we performed, the parameter values were determined. The parameter values are shown in Table 2.

Table 2. Parameter for PROOFS

Parameter	Cache Proxy
neighborhood size	5
subset size	2 (neighborhood size / 2)
Fanout	2 (neighborhood size / 2)
shuffle operation frequency	20 / number of initial distributions (number of cloudlets / number of initial distributions)

5.3 Simulation Process

Our proposed method consists of two steps ("Initial distribution" and "Search/Share by PROOFS") and VM deploying of content distribution. Actually, these operations are executed in 5 steps described below in simulation.

1) Initial distribution
   Cloud transmits contents to several cloudlets. After transmission is complete, cloud notifies cloudlets managed by it to start "Search/Share by PROOFS."
2) Content distribution
   When the cloudlet has a requested content, it is transmitted to the destination using P2P.
3) Content Search
   When the cloudlet does not possess the content notified from cloud, the cloudlet searches the content.
4) Shuffle operation
   Each cloudlet performs shuffle operation each time a certain number of search packets pass.
5) VM deployment
   Each cloudlet deploys a VM for content distribution as soon as content is received.

These 5 steps are continued until all the cloudlets deploy VM and become content distribution possible.

6. EVALUATION

We implemented a simulator in Java and built a virtual edge computing environment on a single computer. The simulation environment is as shown in Table 1, 2 shown in Section 4. Assuming a situation where the cloud is under high load, cloud and cloudlet have the same bandwidth. In experiment based on the simulation, we examined the effectiveness of the proposed method. The following two evaluation criteria were used.

1) Total simulation time (= initial distribution + search/share by PROOFS)
   Total simulation time indicates the number of cycles until all the cloudlets become available for content distribution.
2) Cloud allocated bandwidth
   Cloud allocated bandwidth indicates the bandwidth that the cloud can allocate to each receiver.

6.1 Total Simulation Time

The results of the experiment are shown in Figure 3. When the number of initial distributions is 20, it is the result that the proposed method is not used. In short, the cloud distributes contents to all cloudlets. When the number of initial distributions is 2 to 16, the total simulation time decreased. When initial distribution number is small, it takes much time to searching and sharing by PROOFS. Because the number of initial distributions is small, content requests are increased to each of the cloudlets having the contents, and the content distribution to the cloudlets requesting the contents is delayed. On the other hand, when the initial distribution number is large, it takes much time to initial distribution. Because the number of initial distributions is large, the cloud has to distribute content to many cloudlets. Consequently, it took a lot of time for initial distribution. Therefore, efficient content delivery can be realized by setting the appropriate number of initial distributions and using the proposed method.

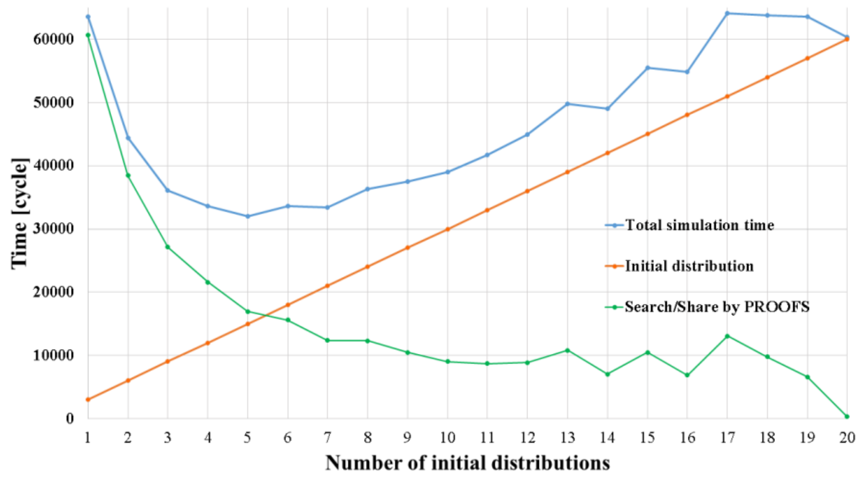

Figure 3. Total Simulation Time

6.2 Cloud Allocated Bandwidth

The results of the experiment are shown in Figure 4. The result using the proposed method is when the number of initial distributions is 5, which is the most efficient in the total simulation time. Also, normal is when the proposed method is not used. Cloud allocated bandwidth was improved by the proposed method. When the proposed method is used, cloud allocated bandwidth returns to 1000 KB/s at an early stage about 15000 cycles. This is because content distribution to each cloudlet in the initial distribution was completed earlier. On the other hand, when the proposed method is not used (normal), it takes much time to return to 1000 KB / s compared to using the proposed method. This has been loading the cloud's bandwidth for a long time because the cloud has to distribute content to many cloudlets at initial distribution. Therefore, by using the proposed method, load distribution can be realized between the cloud and the edge.

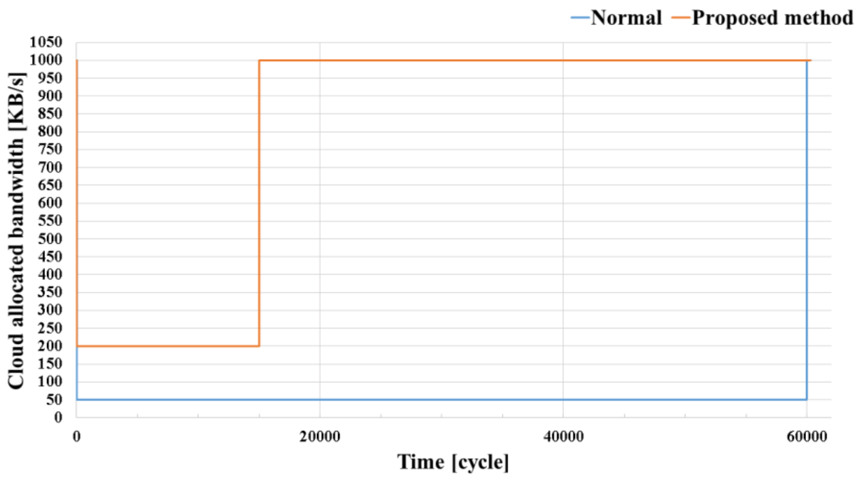

Figure 4. Cloud Allocated Bandwidth

7. CONCLUSION AND FUTURE WORK

In recent years, mobile/IoT devices and data are rapidly increasing. Considering such a situation, when data processing and content distribution are operated in the cloud as in the past, problems such as lack of computing resources and lack of network performance may occur due to load concentration on the cloud. In order to solve these problems, edge computing is attracting attention.

In this study, we focused on study on focusing on content distribution, which is one of utilizations of edge computing. There was inefficiency in content distribution in edge computing, and this caused a problem that the load was concentrated on the cloud. The purpose of this study was to improve the content distribution in edge computing and realize load distribution between cloud and edge. For the purpose we proposed using P2P technology between cloud and edge. we simulated the proposed method using a program created in Java and conducted some experiments. As a result of the experiment, we confirmed that our proposed method realizes efficiency of content delivery in edge computing, and the load is dispersed between cloud and edge.

Issues to address in further studies are as follows.

1) Data shortage in edge computing environment
There are few implementations of edge computing. Therefore, the simulated edge computing environment in this study was set with reference to related study and current edge computing concept. To depend on the edge computing that will be realized in the future, it is necessary to review the edge computing environment simulated in this study.

2) Timing of use of the proposed method
We have experimented on the assumption that the load is concentrated on the cloud. Our proposed method is a symptomatic therapy method used when the load is concentrated on the cloud. At least, by using our proposed method at normal time, it will reduce the pressure on the bandwidth around the cloud. As a future task, we need to consider how much load is concentrated on the cloud to use the proposed method.

3) VM deploying method suitable for edge computing
We simulated VM deploying using Virtual Machine Deployment for Inter-Cloud. The study is not a study based on edge computing. It may be impossible to use the study depending on the edge computing to be realized in the future. Therefore, while paying attention to future edge computing related technology, it is necessary to review the VM deploying technology to be used.

REFERENCES

Garcia Lopez P., et al, 2015. Edge-centric computing: Vision and challenges, *ACM SIGCOMM Computer Communication Review*, Vol. 45, Issue 5, October 2015, pp. 37-42.

Steven C., February 8, 2016., *The Drivers and Benefits of Edge Computing*, Retrieved Augast 23 , 2017, from http://it-resource.schneider-electric.com/h/i/206109108-wp-226-the-drivers-and-benefits-of-edge-computing

Satyanarayanan M., et al, 2009. The Case for VM-Based Cloudlets in Mobile Computing, *IEEE Pervasive Computing*, Vol. 8, No. 4, pp. 14-23.

Zhang, Q., et al, 2012. A Novel Scalable Architecture of Cloud Storage System for Small Files Based on P2P, *2012 IEEE International Conference on Cluster Computing Workshops*. Beijing, China, pp. 41-47.

Wu, J., et al, 2014. Chordmr: A p2p-based job management scheme in cloud, *Journal of Networks* , Vol. 9, No. 3, pp. 541-548.

Sahoo, J., et al, 2016. A Survey on Replica Server Placement Algorithms for Content Delivery Networks, *IEEE Communications Surveys & Tutorials* , Vol. 19, Issue. 2, pp. 1002-1026.

Stavrou A., et al, 2004. A lightweight, robust P2P system to handle flash crowds, *IEEE Journal on Selected Areas in Communications*, Vol. 22, No. 1, pp. 6-17.

Atajanov M., Shimokawa T. and Yoshida N., 2007. Autonomic Multi-Server Distribution in Flash Crowds Alleviation Network. *Proceedings of IFIP 3rd International Symposium on Network Centric Ubiquitous Systems*. Lecture Notes in Computer Science, Springer, pp. 309-320.

Maeno H., Kamiya Y. and Shimokawa T., 2015. Study about High-Performance Virtual Machine Deploy on Inter-Cloud Autoscale. *Internet Conference 2015*. Kobe, Japan, pp. 79-84 (*in Japanese*).