Vol.2019-CSEC-85 No.15
Vol.2019-IOT-45 No.15
2019/5/24

Fault tolerance evaluation of wide area distributed application based on exhaustive FIT scenario generation

Shinnosuke Miura[1,a) Hiroki Kashiwazaki[2,1,b)]

Abstract: A wide area distributed application is affected by network faults due to natural disasters because the servers on which the application operates are distributed geographically in a wide area. Fault Injection Testing (FIT) is a method for verifying fault tolerance of widely distributed applications. In this paper, by limiting network failures to line disconnections, all FIT scenarios are generated and exhaustive evaluation of fault tolerance is performed. Authors evaluate the visualization method of performance data obtained from this evaluation and the reduction of the fault tolerance evaluation cost by the proposed method.

1. Introduction

Information and communication services are essential to people's lives[*1]. There are various information communication services such as e-mail, map services, and services for deposit and management of data, and users rely on them without knowing it. One of the reasons for this is the rapid spread of electronic devices such as smartphones.

Cloud computing is one of the typical information communication services. Cloud computing is a form of provision of computers that can use computers without being aware of the location and number via the Internet. Cloud computing is a concept advocated by Eric Schmidt, who was then CEO of Google in 2006, and has since advanced rapidly into research and development and commercial deployment. At present, Amazon Web Services (AWS) provided by Amazon, Microsoft Azure provided by Microsoft, Google Cloud Platform (GCP) provided by Google, and IBM Cloud provided by IBM etc. are representative. Known as a cloud computing service. Cloud computing services are still rapidly spreading and the market is expanding. In fact, according to the domestic public cloud service market forecast announced by IDC Japan in October 2018, the domestic public loud service market in 2018 is expected to increase 27.4 % over the previous year to 666.3 billion yen.

In addition, the market size in 2022 is estimated to be 1.46.5 trillion yen, which is 2.8 times that in 2017. Cloud computing services are classified into Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS), etc. according to the service level. SaaS is a form of cloud computing that provides software. PaaS provides language processing systems, libraries, middleware, etc. as a basis for operating software. IaaS provides computing infrastructure such as CPU, memory, disk and network. Furthermore, in recent years, a wide variety of cloud computing services such as Machine Learning as a Service (MLaaS), which provides machine learning services, and Desktop as a Service (DaaS), which provides personal desktop environments, are becoming widespread. is there.

These information and communication services are built on a wide area distributed system composed of computer resources of multiple geographically dispersed sites for the purpose of load distribution and improvement of fault tolerance. By distributing geographically, robustness can be secured against failure at a single site.

However, on the other hand, the information communication service constructed as a wide area distributed system is vulnerable to the failure of the network connecting multiple points, and various factors can be considered as the network failure. For example, packet loss due to a network device failure, disconnection of a network cable, or human error caused by an incorrect operation are examples. Also, especially in Japan where natural disasters occur frequently, network failures due to disasters are also conceivable. In fact, in the case of large-scale disasters represented by the Great Hanshin-Awaji Earthquake (1995) and the Great East Japan Earthquake (2011), communication path interruption was a threat.

From the viewpoint of information communication ser-

[1] Osaka University, Ibaraki, Osaka, 567–0047, Japan
[2] National Institute of Informatics, Chiyoda, Tokyo 101–8430, Japan
[a)] miura.shinnosuke@ais.cmc.osaka-u.ac.jp
[b)] reo_kashiwazaki@nii.ac.jp

[*1] Information and Communications in Japan WHITE PAPER 2018
`http://www.soumu.go.jp/johotsusintokei/whitepaper/eng/WP2018/2018-index.html`

vices, it is desirable to be able to provide services with the same level of performance as in normal times, from the viewpoint of information communication services, considering the necessity of information communication services built as a widely distributed system today. However, it is difficult to present the same level of performance at the time of failure. Even in consideration of such current situation, the information communication service provider, which has become indispensable to our lives, needs to show the service user the performance index of the provided information service in the steady state and in the occurrence of a failure. There is. It is also important to show the fault tolerance performance of the provided network from the viewpoint of the network provider that provides the network. In other words, it is necessary for the information communication service provider to grasp the performance of the information communication service that can be provided at the minimum when communication can not be performed not only during steady state but also during non-steady state due to a fault or the like. is there.

2. Proposed Approach

In this paper, an information communication service provided by a group of computers connected via a network is defined as a wide-area distributed service. The sites are connected by a route control device (router), and by operating this route control device, it is possible to generate failures between the sites. Routers include not only appliance products with physical enclosures, but also software routers installed on computers using x86 processors, and virtual routers that can be installed as virtual machines (VMs).

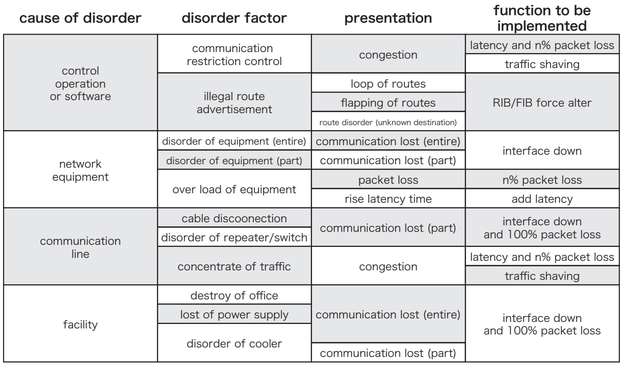
Connect to the console of the Network Operation System (NOS) that operates the router, and execute the NOS command using the Command Line Interface (CLI) to connect the routers between the bases. Failure can occur. However, NOS commands may require interactive input. This interactive input requirement is a barrier when trying to realize programmatic automation.

With the spread of cloud computing[2], NOS also implements similar functions when it becomes possible to manipulate VM deployment and configuration changes using an application programming interface (API). In 2008, Cisco Systems in the United States released the API of its integrated router, Cisco ISR series, in 2008[3]. Vyatta, implemented as a software router, has implemented API operations from Ver. 6.2 in 2011. In this study, we evaluate the fault tolerance and automate this evaluation by generating an intentional failure in the network connecting the bases using the API provided by NOS.

2.1 Classification of network failures

In this study, it is assumed that network failure caused by natural disaster is generated. There are various factors in the network failure caused by natural disaster, from the failure caused by natural disaster to the failure caused by equipment failure etc. In addition, it is also necessary to examine the influence range of the failure pattern, the presence or absence of spatial change, and the temporal transition.

At the time of disaster based on the contents of the disaster event shown in the Ministry of Internal Affairs and Communications “Study Group on the Ways to Secure Communications in Large-scale Disasters and Other Emergency Situations”[4] and “Information Network Safety and Reliability Standards”[5] published by the Ministry. Focus on faults for communication equipment etc. and classify control applied to network equipment for each event (Table2.1).



cause of disorder	disorder factor	presentation	function to be implemented
control operation or software	communication restriction control	congestion	latency and n% packet loss
			traffic shaving
	illegal route advertisement	loop of routes	RIB/FIB force alter
		flapping of routes	
		route disorder (unknown destination)	
network equipment	disorder of equipment (entire)	communication lost (entire)	interface down
	disorder of equipment (part)	communication lost (part)	
	over load of equipment	packet loss	n% packet loss
		rise latency time	add latency
communication line	cable discoonection	communication lost (part)	interface down and 100% packet loss
	disorder of repeater/switch		
	concentrate of traffic	congestion	latency and n% packet loss
			traffic shaving
facility	destroy of office	communication lost (entire)	interface down and 100% packet loss
	lost of power supply		
	disorder of cooler	communication lost (part)	

Table 1 Classification of network failures

Network failures are caused not only by communication link failures but also by control, operation, software, network devices, and equipment environments. In control, operation and software, communication regulation control represented by quality of service and access control list is assumed. In network devices, overloading due to illegal route propagation, total or partial device failure, memory shortage and CPU resource shortage is assumed. In the communication line, cable disconnection, failure of repeaters and switches, and concentration of traffic in the data link layer are assumed. In the facility environment, network equipment and circuit malfunction due to damage to the office building, power loss, and air conditioning failure are assumed. However, the phenomena resulting from these assumptions can be summarized in the following four points.

(1) Increased delay
(2) n% packet loss ($0 < n \leq 100$)
(3) Deactivate network interface card (NIC)
(4) Change of routing control table

Therefore, we set the four types of obstacles implemented in this research.

2.2 Proposed system

The figure 1 shows a schematic diagram of the system for

[2] The Internet White Paper [Japanese] http://www.impressrd.jp/news/180209/NP

[3] https://tech.nikkeibp.co.jp/it/article/NEWS/20080528/304536/

[4] http://www.soumu.go.jp/main_sosiki/kenkyu/saigai

[5] http://www.soumu.go.jp/menu_seisaku/ictseisaku/net_anzen/anshin

realizing fault tolerance verification by intentional failure occurrence and its automation proposed in this research. The proposed system consists of fault pattern generator, FIT controller, bench marker, and visualizer. The fault pattern generator generates fault patterns based on the topology information. The FIT controller updates topology data according to the fault, using the fault pattern as input. The benchmark performs the benchmark on the wide area distributed service in conjunction with the FIT controller. After that, the bench marker sends the benchmark result to the visualizer. The visualizer receives the measurement results and visualizes the data. The following describes each component.

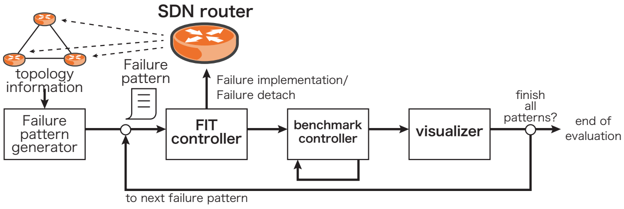


Fig. 1 A diagram of proposed SDN-FIT system

2.2.1 Failure pattern generator

The fault pattern generator generates fault patterns according to the number of circuits in the topology from the topology information of the wide area distributed service. Bases supporting wide-area distributed services to be subjected to fault tolerance verification are connected by a routing controller that can be operated by API. The identifiers are given to each base, and the NICs at both ends of the circuit connecting the bases are given identifiers in the NOS of each router. From the above information, the topology of the base supporting wide area distributed service can be expressed by nesting of hash and array. Yet Another Markup Language (YAML) is a format that represents structured data, and the topology can be described using YAML. For example, a network consisting of three bases in the figure 1 can be expressed as shown in listing1. In this topology data, base A is connected to B by eth0 and to C by eth1; base B is connected to A by eth0 and A by eth1; and base C is connected to A by eth0 and B by eth1. To indicate that

Listing 1 3nodes_network

```
- A:
- [[eth0, B], [eth1, C]]
- B:
- [[eth0, A], [eth1, C]]
- C:
- [[eth0, A], [eth1, B]]
```

At the same time, this topology data shows the circuit between bases. In the example of Listing 1, the line a connecting eth0 of base A to eth0 of base B, the line b connecting eth1 of base A to eth0 of base C, and the line connecting eth1 of base B to eth1 of base C Indicates that there are 3 lines of c. When the number of lines is m, the fault pattern generator searches for combinations of fault patterns that generate all n ($0 < n \leq m$) double faults in each line. One failure pattern is represented by an array composed of the failure type identifier, the identifier of the router that generates the failure, and the identifiers of one or more NICs that cause the failure in the router. Listing 2 shows the case where the line a and the line b are interrupted due to the deactivation of the NIC.

Listing 2 shutdown

```
- [shutdown, A, eth0, eth1]
```

2.2.2 FIT controller

The FIT controller uses the fault patterns created by the fault pattern generator to update probabilistic data in accordance with each fault. The implementer of fault tolerance verification provides the FIT controller with router information of the base supporting the wide area distributed service to be verified. The FIT controller uses the API for the router to obtain NIC information of each router and the IP address assigned to that NIC. It is determined that NICs in the same IP address range at different sites are connected, and topology data is created.

The FIT controller provides the created topology data to the fault pattern generator, and the fault pattern generator returns all fault patterns to the FIT controller. The FIT controller sequentially processes the obtained fault pattern data. As described in Section 2.2.1, one failure pattern consists of an identifier of the failure type, an identifier of the router that causes the failure, and an identifier of one or more NICs that cause the failure in that router. The FIT controller reads this array, and uses the API to control the NIC specified as the router and the instruction corresponding to the identifier of the failure type.

After the control that implements the fault condition ends normally, the FIT controller applies processing to the benchmark to measure the performance in the event of a fault. When the execution of the benchmark ends normally, the FIT controller controls the specified NIC of the router using the API and cancels the failure status. When the release of the fault condition ends normally, the FIT controller applies processing to the visualizer to visualize the performance measurement results obtained by the bench marker. Execute these processes for all failure patterns, and repeat them until finished.

2.2.3 Benchmarker

Benchmarker performs object storage benchmarking. The benchmarker then sends the benchmark results to the visualizer. In benchmarker, benchmark software is implemented according to the wide area distributed service to be verified. According to an instruction from the FIT controller, benchmarker executes the specified benchmark software based on the specified arguments.

Those who perform fault tolerance verification install benchmark software according to the items they want to investigate. For example, if the wide area distributed service is a Web service and you want to verify its response performance, the fault tolerant verifier uses Apache Bench[*6]. If wide area distributed services are POSIX compliant storage, fio[*7] or IOZONE[*8] may be used as benchmark software.

2.2.4 Visualizer

The visualizer receives measurement results from the bench marker and visualizes the result data. The measurement results obtained by the bench marker are placed in a local storage area in the computer where the visualizer is deployed, or placed in a place that can be obtained by remote access. When the visualizer receives an instruction from the FIT controller, it reads the specified file and visualizes the data according to the specified drawing method. The visualizer shows the location of the visualized file. This enables the verifier to view the visualized data.

3. Implementation

We deploy a wide area distributed service in a real environment and implement SDN-FIT system to verify the fault tolerance of this wide area distributed service.

3.1 Implementation of evaluation environment

We will explain the elemental technology of the environment that was built when conducting this evaluation.

3.1.1 Distcloud

Distcloud is a wide-area distributed virtualization platform under Regional InterCloud Subcommittee (RICC) of the Internet Technology 16th Committee (ITRC) of the Japan Society for the Promotion of Science and Technology. It is constructed by connecting computers distributed by geographically dispersed universities, research organizations, and cloud operators by broadband networks (Figure 2). Wide-area distributed virtualization infrastructure is realized by deploying scale-out distributed storage. Focusing on live migration as a disaster recovery method, we implement storage technology with little degradation of I/O performance before and after wide-area live migration [1], [2].

Distcloud's bases are connected by SINET[*9], an academic information network provided by the National Institute of Informatics. It uses L2VPN / VPLS service[*10] that allows Ethernet frames to be exchanged between LANs at remote sites.

Virtual Private LAN Service (VPLS) [3] is a technology that can transfer Ethernet frames using Multi-Protocol Label Switching (MPLS) defined in RFC3031 [4]. Because a virtual Ethernet LAN can be constructed for each network created in each network, a protocol to be used does not depend on IP, and a network with L2 connectivity can be constructed (Figure 3).

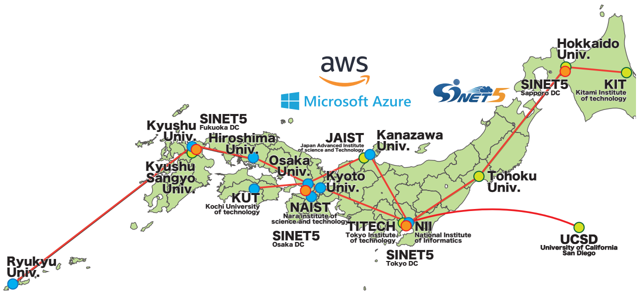

aws
Microsoft Azure
SINET5
Hokkaido Univ.
SINET5 Sapporo DC
KIT Kitami Institute of technology
Kyushu Univ.
SINET5 Fukuoka DC
Hiroshima Univ.
JAIST Japan Advanced Institute of science and Technology
Kanazawa Univ.
Kyushu Sangyo Univ.
Osaka Univ.
Kyoto Univ.
Tohoku Univ.
KUT Kochi University of technology
NAIST Nara institute of science and technology
TITECH Tokyo Institute of technology
NII National Institute of Informatics
SINET5 Osaka DC
SINET5 Tokyo DC
UCSD University of California San Diego
Ryukyu Univ.

Fig. 2 Distcloud

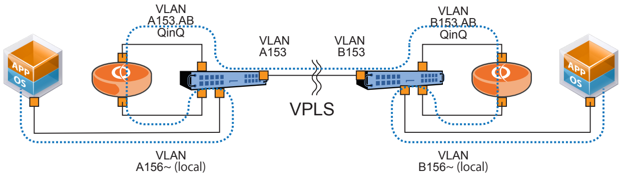


Fig. 3 Inter-communication among sites with VPLS

Distcloud uses SINET VPLS and prepares L2 networks called ID 0153 and ID 0154 respectively. An ID 0153 is a network used for communication of services and applications, and an ID 0154 is a network for management of devices constituting the base. In addition, an L3VPN network called ID 0155 is prepared separately. As for ID 0155, / 24 IPv4 addresses are assigned in advance for each site.

A base connected to Distcloud needs to prepare a VLAN to connect with the ID 0153, ID 0154 and ID 0155 in the LAN in the base. A base connected to Distcloud prepares computer resources and connects this with the above-mentioned VPLS. Two L2VPN / VPLS connectivity by VPLS provided by SINET, one IPv4 network by L3VPN, three VLANs in the site, and computer resources connected to it are the environment provided by Distcloud.

3.1.2 VyOS

VyOS[*11] is a network OS developed by open source. It is developed based on Debian GNU / Linux. Originally from Vyatta mentioned in section 2, it was forked from version 6.6 R1 of Vyatta Core, which is the free version of Vyatta. In addition to being installed on a physical computer and used as a software router, it may also be installed as a VM in a virtual environment and used as a virtual router. Like a general NOS, it has a unified CLI like a hardware router.

In order to cause communication failure due to FIT proposed in this research between bases, it is necessary to configure an independent network at each base that configures Distcloud, and it is necessary to perform routing control with the deployed router at the base . The NICs connecting between the bases are independent of the networks owned by each base, and the two connected bases need to belong to the same network. In Distcloud, only the aforementioned network with ID 0153 is provided as a service network.

Although it is conceivable to newly secure an independent

[*6] https://httpd.apache.org/docs/2.4/programs/ab.html
[*7] https://github.com/axboe/fio
[*8] http://www.iozone.org/
[*9] https://www.sinet.ad.jp/
[*10] https://www.sinet.ad.jp/connect_serice/service/l2vpn
[*11] https://vyos.io/

VLAN for connection between sites as L2VPN / VPLS, it is necessary to apply for the number of lines connecting between sites and to apply L2VPN / VPLS. This method becomes impractical if the number of connected lines increases. Therefore, by using IEEE802.1ad (Q-in-Q)*12 in a router deployed at each site, networks of different VLANs can be configured across different sites on the ID0153 network.

VyOS is a network OS that can communicate with Q-in-Q and can realize all the failure implementations described in Section 2.2.1 on its own. As VyOS is developed based on Debian GNU / Linux as mentioned above, it can be used by specifying the tc command of Linux*13 as `traffic-policy` of VyOS. For these reasons, it is used for verification experiments of this study.

3.1.3 CLOUDIAN HYPERSTORE

CLOUDIAN HYPERSTORE*14 is an object storage product that is fully compatible with the Amazon S3*15 API marketed by CLOUDIAN.

Object storage is a computer data storage that manages data as an object as opposed to file systems that manage data as a file hierarchy and other storage architectures such as block storage that manages data as blocks specified by sectors and tracks Refers to the architecture.

Each object contains data, metadata, and a unique identifier. Object storage can be implemented at multiple levels, including object storage device level, system level, and interface level, in which case object storage is an interface directly programmable by the application, multiple instances of physical hardware It provides data management functions including namespaces that can span and replication of data.

CLOUDIAN HYPERSTORE has a function to manage data at bucket level, and control parameters can be defined at bucket level. The bucket policy is a parameter that determines the number of copies of data. In this evaluation experiment, when the client uploads a file (PUT operation), when three copies are created at all bases. The policy is to return an acknowledgment (ACK). CLOUDIAN HYPERSTORE is a wide area distributed service that is also used in the back end of the video sharing site "Nico Nico Douga"*16 of Dwango Co., Ltd*17*18.

3.1.4 COSBench

COSBench is an object storage benchmark tool developed by Qing Zheng et al. Object storage has different indexes (workloads) to keep the performance of the access system in a proper state for each service that utilizes it. However, in 2013, when the use of object storage started to increase worldwide, there was no workload for object storage. COSBench was designed and implemented to address this problem [5].

The development of COSBench, which has been developed by Intel, aims at preparing both object storage system performance comparison and system optimization, and is a scalable implementation to cope with the scale of the system. At COSBench, there are two types of drivers: a driver that loads object storage, and a controller that instructs to load the driver. If the load details such as read / write (R: W) ratio are described in the XML file that describes the workload and registered in the controller by the web console or the command for CLI, it will be queued on the controller. The load test is performed sequentially.

3.2 Construction of wide area distributed system environment

In this study, CLOUDIAN HYPERSTORE and its environment for verification are constructed using three Distcloud bases (Osaka University, Tohoku University, Ryukyu University) (Figure 4). The x86 server installed at Osaka University has a CPU of 28 physical cores, a main memory of 256GB, a 3.6TB SSD array is connected, and an exclusive 10 Gbps leased line is connected to the campus network. Tohoku University has 28 physical cores of CPU, 128GB of main memory and 2.2TB of disk array connected, and is connected to the campus network via a shared 10 Gbps line. SINET 5 connects Osaka University and Tohoku University at 100 Gbps, and Osaka University and Ryukyu University, and Tohoku University and Ryukyu University at 40 Gbps.

Install Ubuntu*19 18.04 LTS, an operating system based on Debian GNU / Linux, on the x86 server at each site. In order to run VM on this Linux, we build the environment of KVM*20 which is a virtualization module that makes Linux kernel function as a virtual hypervisor.

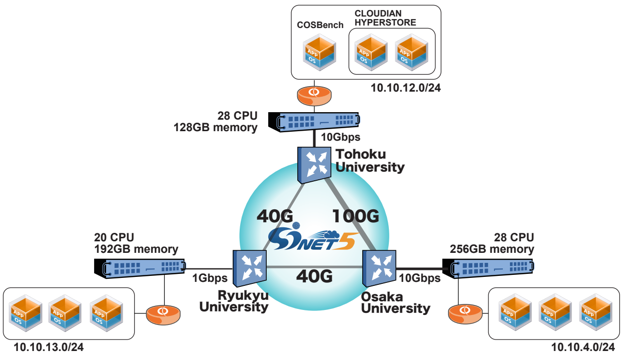


Fig. 4 A diagram of the wide-area distributed system

Then we created the following four VMs on Linux installed on the x86 server at each site.

- CLOUDIAN HYPERSTORE 2VMs
- CentOS7 for COSBench 1VM
- VyOS 1VM

The VMs performance of CLOUDIAN HYPERSTORE, COSBench, and VyOS are shown in Table 2, 3 and 4 , respectively.

[12] http://www.ieee802.org/1/pages/802.1ad.html
[13] https://www.linux.org/docs/man8/tc.html
[14] https://cloudian.com/jp/products/
[15] https://aws.amazon.com/jp/s3/
[16] https://www.nicovideo.jp/
[17] http://dwango.co.jp/
[18] https://news.mynavi.jp/kikaku/20190204-754979/
[19] https://www.ubuntu.com/
[20] http://www.linux-kvm.org/page/Main_Page

OS/Version	CentOS Linux release 7.4.1708
RAM [MiB]	32768
number of vCPUs	8

Table 2 Performance of VM for Cloudian Hyperstore

OS/Version	CentOS Linux release 7.6.1810
RAM [MiB]	4096
number of vCPUs	2

Table 3 Performance of VM for COSBench

OS/Version	VyOS 1.1.8
RAM [MiB]	512
number of vCPUs	1

Table 4 Performance of VM for VyOS

The VM belongs to an independent network for each location, and assigns an IPv4 address that does not overlap with the networks of other locations. A unique VLAN is assigned to this network in the site, and VMs for CLOUDIAN HYPERSTORE at each site, VMs for COSBench, and one NIC of VyOS are connected to the bridge interface of this VLAN.

The VyOS at each site has an NIC for configuring a backbone network connected to the VyOS at the other two sites. As described in Section 3.1.2, NICs connected to each backbone network need to belong to independent VLANs, so select VLANs that do not overlap with VLANs at all sites. The two NICs connected to the backbone network are connected via a unique L2 network created on the L2 network of ID 0153 by Q-in-Q.

In VyOS at each site, OSPF [6] is operated as an Interior Gateway Protocol [7], the cost with the adjacent site is set to 10, dead-interval to 40 seconds, hello-interval to 10 seconds, and retransmit-interval to 5 seconds. In this way, VMs belonging to the networks at each site can communicate with each other. Also, by setting verb — disabled — for the interconnected NICs, that NIC can be deactivated and communication disconnection can occur. When the NIC becomes inactive and communication interruption occurs, OSPF recalculates the path in the topology where communication interruption occurred, and the path is changed by sending Link State Update packet. The inactive state of the NIC can be released by the `delete` command.

3.3 Implementation of a proposed system

In this research, five programs were created to implement the FIT controller, the benchmarker, and the visualizer among the proposed systems described in Section 2.2. In the FIT controller, in this paper, in order to simplify the evaluation of CLOUDIAN HYPERSTORE, we implemented the deactivation of the network interface among the four faults shown in Section 2.1. The outline of each script is as follows.

3.3.1 network failure implementation script

The network failure implementation script is one of the scripts that configure the FIT controller, and causes a failure in the network connecting between bases. In VyOS, it is separated into operation mode and configuration mode, and it connects to the VyOS console and switches to configuration mode by entering `configure` at the prompt. Here, in order to deactivate eth0, you need to input a command as follows.

```
 # set interface ethernet eth0 disable
```

The network failure implementation script is an implementation of this series of processing using VyOS cli-shell-api[*21]. The network failure implementation script must first initialize the environment. Use a command to acquire environment variables required for initialization.

```
# /bin/cli-shell-api getSessionEnv
```

The `getSessionEnv` command outputs a series of operations specific to the session by giving a process identifier as an argument. The initialization is completed by executing as follows after initialization.

```
# /bin/cli-shell-api setupSession
```

After initialization is complete, the following command can deactivate the NIC with the identifier specified by the `NIC identifier`.

```
# /opt/vyatta/sbin/my_set ethernet \\
  [NIC identifier] disable
```

The program that executes this series of processing receives the identifier of NIC as an argument. When multiple `NIC identifiers` are specified, the specified NICs are sequentially deactivated.

3.3.2 network failure deactivation script

The network failure deactivation script is one of the programs that configure the FIT controller, and is a script that deactivates the failure that has occurred in the network connecting the bases. As described in Section 3.3.1 Network Failure Implementation Script, after using cli-shell-api of VyOS and initializing by executing getSessionEnv command and / bin / cli-shell-api setupSession, it is used as an argument. Execute the following command for the specified NIC identifier to release the inactive status.

```
# /opt/vyatta/sbin/my_delete ethernet \\
[NIC identifier] disable
```

The network failure deactivation script can receive multiple arguments in the same way as the network failure implementation script, and when multiple `NIC identifiers` are specified, the inactive state of the specified NIC is released sequentially.

3.3.3 Benchmark execution script

The benchmark execution script is a program that configures benchmarks, and is a program for performing comprehensive benchmarks that is specialized for COSBench. The

[*21] https://wiki.vyos.net/wiki/Cli-shell-api

benchmark execution script uses the program `cli.sh` that executes the load test installed on the controller of COSBench. Benchmark can be performed by giving an XML file name that contains information necessary for the workload as an argument when executing this program. Since benchmarking is performed repeatedly changing the workload, the benchmark execution script generates an XML file describing all the workloads in advance.

Workloads executed with `cli.sh` are given a workload identifier, and the benchmark output of different workloads can be distinguished by this identifier. The program of the visualizer described in Section 3.3.4 also has a function to save this identifier as data in order to align the data using this identifier.

3.3.4 Data formatting script

The data shaping script is one of the programs that make up the visualizer, and is a program that organizes the data output by the benchmark execution script. The bandwidth information to be evaluated is extracted from the data of multiple workloads output by the benchmark execution script, sorted, and compared with the steady-state benchmark result performed before the fault tolerance evaluation is performed, Normalize the implemented daily steady state benchmark and output it. The output data is shaped as a format that can be read by gnuplot[22] used in the visualization script in Section 3.3.5.

3.3.5 Visualization script

The visualization script is one of the programs that compose the visualizer, and the data output from the data shaping script is output using gnuplot. gnuplot is graph utility software based on command line operation that runs on various operating systems such as various UNIX operating systems and Windows. The visualization script can change the width of the x-axis and the y-axis according to the arguments given, and is an implementation that enables adaptive visualization according to the range of the benchmark.

4. Evaluations

We evaluate the disaster tolerance of CLOUDIAN HYPERSTORE quantitatively using CLOUDIAN HYPERSTORE, which is constructed in Section 3.2, and the SDN-FIT system implemented in Section 3.3, and visualize the evaluation results.

The network consisting of three bases constructed in Section 3.2 is connected by three circuits, and the failure pattern of a single failure of the network is the following three patterns.

(1) Osaka-Tohoku
(2) Ryukyu-Osaka
(3) Tohoku-Ryukyu

When a double failure occurs, the connectivity of the two bases is maintained but the split-brain state is isolated from the other one. As described in Section 3.1.3, the CLOUDIAN HYPERSTORE bucket used in this evaluation has a policy to return an ACK when all 4 sites have been replicated in all 3 sites. Unable to complete the PUT process, all COSBench workloads fail. Because triple failure also causes all workloads to fail for the same reason, this evaluation does not evaluate double failure and triple failure.

In this evaluation, the minimum value of the file size is set to 64 KB by using `benchmark_exec` command, and the evaluation is performed with nine file sizes up to 16 MB by doubling each time. Also, the workload read:write (RW) ratio is increased by 10 % from 0 % to 100 %. After each failure pattern is implemented, this workload is executed, and the non-steady-state benchmark results are normalized using the previously measured steady-state benchmark results. The average transfer rates in steady state are shown in Figure 5 and 6 respectively.

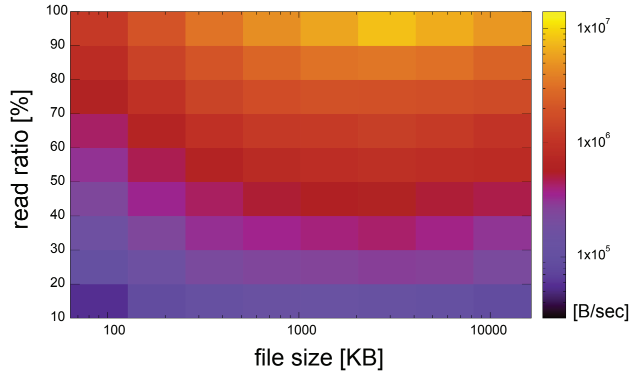


Fig. 5 Averaged bandwidth under steady state (Read)

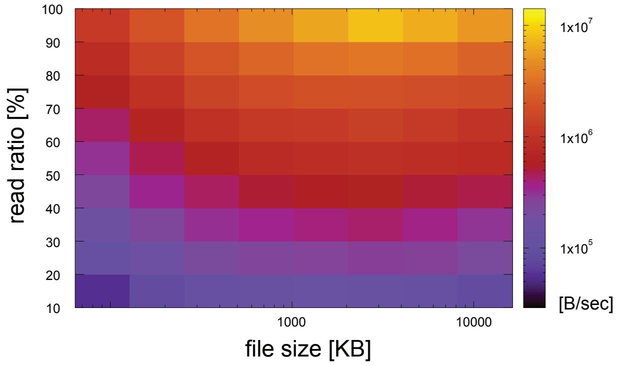


Fig. 6 Averaged bandwidth under steady state (Write)

The horizontal axis is the log axis of the file size, and the vertical axis is the RW ratio, which is visualized as a three-dimensional color map. This can be visualized as a two-dimensional heat map by displaying it in the gaze direction of a vector parallel to the Z axis. With this heat map, it is possible to grasp the relative quality deterioration in the unsteady state for each failure pattern.

As described in Section 3.2, when you deactivate the VyOS interface connected to the backbone network connecting each location, OSPF detects it and re-routes the routing table in the topology where one of the backbone networks

[22] `http://www.gnuplot.info/`

is lost. Calculate and send Link State Update packets to VyOS at other sites, and the routing control tables of VyOS at all sites are updated. In the steady state, all locations can reach all other locations with one hop.

Among them, the one with the largest increase rate of the delay time is at the time of failure occurrence between Osaka University and Tohoku University of failure 1, and the delay time is 14.81 ms at steady state, and it is 56.71 ms which is 3.8 times. The average transfer rate of normalized Read at failure 1 is shown in Figure 7, and the average transfer rate of Write is shown in Figure 8. The average transfer rate of Read is 8MB, and the RW ratio shows a worst value of 0.42 at 60%, which indicates that only 42% performance can be obtained compared to the average transfer rate at steady state. Moreover, the average transfer rate of Write is less than 0.83 in the whole area, and the worst value is 0.56 in 10% of RW ratio of 4 MB.

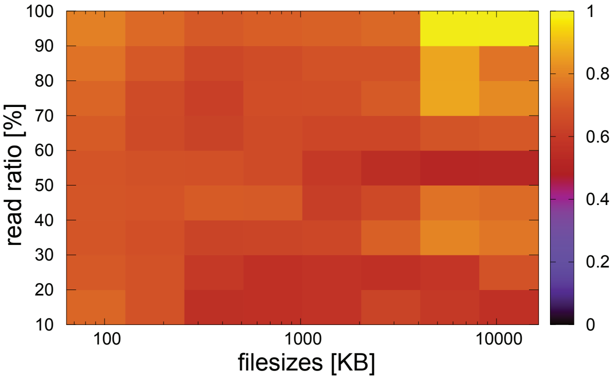


Fig. 7 standardized averaged bandwidth between Osaka-Tohoku (Read)

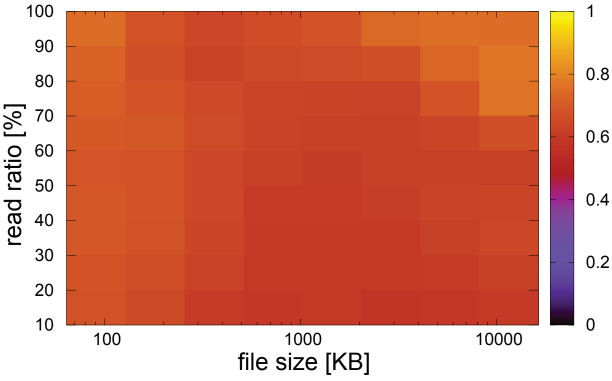


Fig. 8 standardized averaged bandwidth between Osaka-Tohoku (Write)

5. Conclusion

In this research, we proposed a system that supports automation of fault tolerance evaluation of wide area distributed service for the purpose of quantifying fault tolerance evaluation of information communication service constructed as wide area distributed system and reducing cost required for evaluation. . This system consists of fault pattern generator, FIT controller, bench marker and visualizer.

In order to evaluate the effectiveness of this proposal, we constructed a wide area distributed service on the wide area distributed platform "Distcloud", and performed fault tolerance verification by implementing the proposed method for this service. Perform comprehensive benchmarks based on failure patterns that are automatically generated by providing router information at multiple locations, and compare the steady-state and non-steady-state performances to reduce the performance against steady-state The heat map was output and visualized.

In order to evaluate the cost reduction by automation of the fault tolerance evaluation of this proposal, the time required for failure occurrence was measured and compared between the proposed method and the manual case. In single failure, double failure, and triple failure, it was confirmed that the proposed system finished processing in less than 20% of the time required for manual failure implementation. It is also found that the probability of performing incorrect fault implementation for a given fault pattern occurs with a probability of 5% or more in the manual case. From this, it is shown that the proposed system reduces human restraint time and realizes accurate fault implementation.

Acknowledgments Part of this work was carried out under the Cooperative Research Project Program of the Research Institute of Electrical Communication, Tohoku University. The research was supported by ROIS NII Open Collaborative Research 19FA08, JSPS KAKENHI Grant Number JP19K20256.

References

[1] Nakagawa, I., Ichikawa, K., Kondo, T., Kitaguchi, Y., Kashiwazaki, H. and Shimojo, S.: Transpacific Live Migration with Wide Area Distributed Storage, *2014 IEEE 38th Annual Computer Software and Applications Conference*, pp. 486–492 (online), DOI: 10.1109/COMPSAC.2014.71 (2014).

[2] Nakagawa, I., Kashiwazaki, H., Shimojo, S., Ichikawa, K., Kondo, T., Kitaguchi, Y., Kikuchi, Y., Yokoyama, S. and Abe, S.: A Design and Implementation of Global Distributed POSIX File System on the Top of Multiple Independent Cloud Services, *2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, pp. 867–872 (online), DOI: 10.1109/IIAI-AAI.2016.75 (2016).

[3] Rekhter, Y. and Kompella, K.: Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling, RFC 4761 (2007).

[4] Viswanathan, A., Rosen, E. C. and Callon, R.: Multiprotocol Label Switching Architecture, RFC 3031 (2001).

[5] Zheng, Q., Chen, H., Wang, Y., Zhang, J. and Duan, J.: COSBench: Cloud Object Storage Benchmark, *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*, ICPE '13, New York, NY, USA, ACM, pp. 199–210 (online), DOI: 10.1145/2479871.2479900 (2013).

[6] Ferguson, D., Lindem, A. and Moy, J.: OSPF for IPv6, RFC 5340 (2008).

[7] Faucheur, F. L., Merckx, P., Telkamp, T., Uppili, R. and Vedrenne, A.: Use of Interior Gateway Protocol (IGP) Metric as a second MPLS Traffic Engineering (TE) Metric, RFC 3785 (2004).