Received 16 April 2020; accepted 11 May 2020. Date of publication 19 May 2020; date of current version 8 June 2020.

*Digital Object Identifier 10.1109/OJCOMS.2020.2994995*

# Experiment and Availability Analytical Model of Cloud Computing System Based on Backup Resource Sharing and Probabilistic Protection Guarantee

**TAKEHIRO SATO [1] (Member, IEEE), FUJUN HE [1] (Student Member, IEEE), EIJI OKI [1] (Fellow, IEEE), TAKASHI KURIMOTO [2] (Member, IEEE), AND SHIGEO URUSHIDANI [2] (Member, IEEE)**

[1]Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan

[2]Information Systems Architecture Science Research Division, National Institute of Informatics, Tokyo 101-8430, Japan

CORRESPONDING AUTHOR: T. SATO (e-mail: takehiro.sato@i.kyoto-u.ac.jp)

This work was supported in part by Japan Society for the Promotion of Science KAKENHI, Japan, under Grant 18H03230 and Grant 19K14980, and in part by the Research Fund of National Institute of Informatics.

**ABSTRACT** A probabilistic protection guarantee enables a cloud provider to improve the availability of their cloud computing system in a cost-efficient manner. A backup resource allocation strategy based on the probabilistic protection guarantee reduces the total amount of required backup computation resources by allowing multiple virtual machines to share the same backup computation resources of a physical machine. There have been no experimental studies that investigate the impact of applying the probabilistic protection guarantee to a cloud computing framework in real use. This paper presents an experiment of failure recovery on the cloud computing system in which the backup computation resources are shared by multiple virtual machines. We implement a prototype cloud system by using the OpenStack framework to demonstrate the failure recovery scenario according to the backup resource allocation strategy. We develop an availability analytical model for the backup resource allocation strategy. Based on the analytical model, we present case studies which derive the availability of cloud system by using the measurement results of the experiment.

**INDEX TERMS** Availability, cloud computing, failure recovery, probabilistic protection guarantee, shared protection.

## I. INTRODUCTION

CLOUD computing, which provides resource and services to customers through the Internet, has already become an essential technology in our lives. Cloud services are generally categorized into three types; infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS) [1]. An IaaS provider provides customers with virtual machines (VMs) that have specific computation performance according to the customers' requests [2], [3]. VMs are launched in physical machines (PMs) deployed in a datacenter of the IaaS provider by occupying computation resources, such as CPU cores, memory, and storage volume.

A failure of PM due to a hardware or software malfunction causes an interruption of service provided by using VMs, which leads to a huge financial loss of both customers and the IaaS provider. Since the IaaS enables customers to rearrange the number and performance of VMs flexibly while reducing the cost of in-house operations, personal as well as business usage has increased [4], [5]. Therefore, the IaaS provider must provide VMs for customers while supporting a certain level of availability as defined in the Service Level Agreement (SLA) contracted between the provider and the customers [6].

The deployment of dedicated backup PM to every primary PM improves the availability of cloud computing system in a

simple manner. In this method, each of backup PMs provides the same set of computation resources and works as a mirror of the corresponding primary PM. The backup PM runs the same workloads with the primary PM at the same time. When the primary PM fails, the backup PM takes the place of the primary PM without any service interruption. This backup method covers every random failure scenario and guarantees 100% protection unless the backup PM itself fails. However, this method raises the implementation cost of PMs since the PMs need to be duplicated.

The work in [7] presented a backup resource allocation strategy which provides a probabilistic protection guarantee in the cloud computing system. This strategy allocates computation resources of backup PMs to VMs in primary PMs so that the probability that the protection fails due to insufficient backup resources is not greater than a certain probability. The computation resources of backup PMs can be shared by multiple VMs in different primary PMs as long as the constraint of probabilistic protection is satisfied.

The backup resource allocation strategy based on a probabilistic protection guarantee enables an IaaS provider to reduce the amount of required backup computation resources, which leads to saving costs for improving availability. However, several aspects that occur in the real-world cloud system such as PM locations and failure recovery time have not been considered in the strategy. These aspects are needed to be incorporated with the strategy to adopt the strategy to cloud computing systems in real use. In addition, this backup resource allocation strategy does not directly guarantee the availability, which the IaaS provider supports. Different from the dedicated protection which creates a complete mirror of primary PM in the corresponding backup PM, the protection based on this strategy requires a certain period of time to recover from the failure; VMs need to be relaunched in backup PMs by using snapshots of failed VMs. To judge whether the required availability can be supported, we should take into account the failure recovery time and calculate the availability which is achieved by applying the backup resource allocation strategy.

This paper presents an experiment of failure recovery on the cloud computing system which is based on the backup resource allocation strategy providing a probabilistic protection guarantee. Our experiment has significance in verifying the impact of the real-world cloud characteristics on the backup strategy based on probabilistic protection guarantee. A prototype cloud system demonstrates the sharing of backup computation resources. The prototype cloud system is based on OpenStack [9], which is an open-source software framework to create and manage a cloud computing system. We examine the failure recovery scenario according to the backup resource allocation strategy by using the prototype cloud system. The failure recovery scenario is tested in two environments, which differ in the geographic location of primary and backup PMs, and under two cases about the prefetch of VM snapshots. The required time for recovery from failure and creating a snapshot of VM which is used for failure recovery are measured. Measurement results obtained through such an experimental approach help IaaS providers to derive the availability of their cloud systems. We develop an availability analytical model for the backup resource allocation strategy. We present case studies of availability analysis based on the developed model and the measurement results of the experiment.

A part of this paper was presented in [8], which mainly focused on the implementation of prototype cloud system and the results of failure recovery tests. The extensions to the work in [8] are mainly described as follows. We describe the detailed configurations and results of the failure recovery tests. We conduct an additional test to measure the required time for creating a VM snapshot under the existence of storage access since it can lead to degradation of the availability of cloud system. We develop an availability analytical model, which incorporates the real-world cloud characteristics with the theoretical model of the probabilistic protection guarantee, for the backup resource allocation strategy. We analyze the availability of cloud system as case studies according to the measurement results of experiment. We extensively survey existing researches related to our work.

The rest of the paper is organized as follows. Section II briefly describes the backup resource allocation strategy presented in [7]. Section III introduces the implementation of the prototype cloud system. Section IV shows the experiment results performed on the prototype cloud system. Section V presents the availability analytical model for the backup resource allocation strategy and provides case studies. Section VI describes the related works. Section VII concludes this paper.

## II. BACKUP RESOURCE ALLOCATION STRATEGY

We introduce the backup resource allocation strategy based on backup resource sharing and probabilistic protection guarantee [7] in this section. This strategy allocates computation resources of backup PMs to VMs running in primary PMs, where PMs are separated into primary ones and backup ones. The allocation of single type of computation resource is considered to simplify the discussion. According to the description in [7], we call the amount of computation resource in PMs "capacity" in the rest of this section.

Fig. 1 shows an example of capacity allocation in a cloud provider, which consists of three primary PMs and two backup PMs. The capacity of primary PMs is exclusively used to launch VMs. The amount of capacity allocated to each VM in each primary PM is determined according to the performance of VM that the customer requests. On the other hand, the capacity of backup PMs is exclusively used for the failure recovery purpose. When a failure occurs in a primary PM, the backup capacity is used to relaunch VMs which initially run on the failed PM. As shown in Fig. 1, VMs in different primary PMs can share the capacity of the same backup PM. This means that the protection can fail due to the shortage of backup capacity when multiple primary PMs fail simultaneously. For example, when only primary PM #1

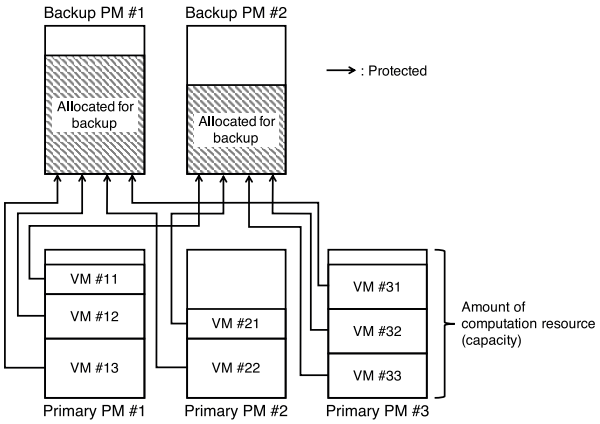



**FIGURE 1.** Example of computation resource allocation in primary and backup PMs.

fails in Fig. 1, the recovery of VMs succeeds; VM #11 is relaunched on backup PM #2, and VMs #12 and #13 are relaunched on backup PM #1. However, when primary PMs #1 and #2 fail simultaneously, several VMs cannot be recovered by using the allocated backup capacity.

The backup resource allocation strategy provides a probabilistic protection guarantee; it is guaranteed that the probability that the protection provided by a backup PM fails does not exceed a certain probability. The strategy determines the allocation of backup capacities to VMs in primary PMs under the condition that the primary PMs fail in a probabilistic manner and the backup PMs never fail.

The work in [7] provides a mixed integer linear programming (MILP) problem for the strategy to obtain the solution in which the total backup capacity required to protect the primary PMs is minimized. To deal with the uncertainty in the backup capacity required for the protection, a robust optimization technique [11] is used in the formulation of MILP problem. A heuristic approach based on simulated annealing is adopted for the case where the MILP problem cannot be solved in a practical time. A numerical evaluation in [7] shows that the backup capacity required for the protection is reduced by adopting the probabilistic protection guarantee.

## III. IMPLEMENTATION OF PROTOTYPE CLOUD SYSTEM

The prototype cloud system is implemented to examine the failure recovery scenario based on the backup resource allocation strategy in Section II. Fig. 2 shows the schematic view of the prototype cloud system, which has the minimum configuration to test the failure recovery scenario. Three PMs are separated into two primary PMs and one backup PM, each of which has a capability to configure up to one VM. The prototype cloud system assumes the use case that a customer runs an application which continuously generates output data in each VM. The generated data are successively stored in a virtual storage volume, which is provided in a shared storage system and attached to each VM. The controller manages the PMs and the shared storage system via a management network.

Based on the assumption in [7], we consider only a failure in the primary PMs. The common failure recovery scenario

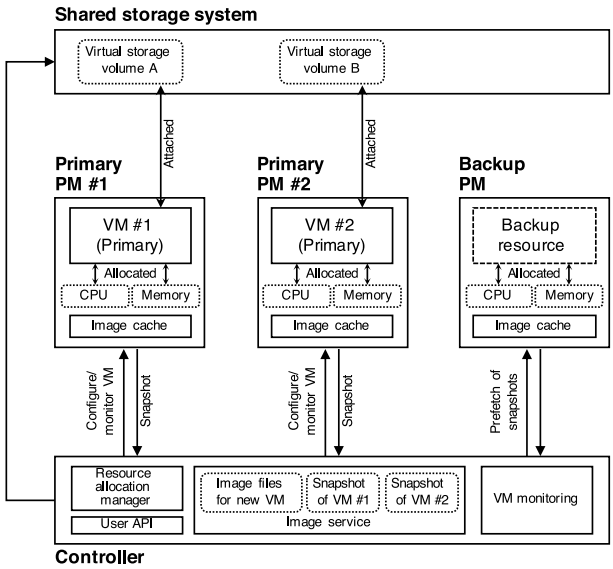



**FIGURE 2.** Schematic view of prototype cloud system.

based on checkpointing [10] is applied in the prototype cloud system. During the normal operation, snapshots of running VMs in the primary PMs are periodically created and stored as image files in the controller. When a failure occurs in the primary PM, the controller recovers the VM in the failed PM by relaunching the VM in the corresponding backup PM by using the latest snapshots. The controller reattaches the virtual storage volume to the relaunched VM. Then the VM restarts its application from the point when the latest snapshot was created. Note that we use the common failure recovery scenario since we consider it is meaningful to make the verification based on a scenario that has been widely applied in cloud systems.

The prototype cloud system is implemented based on OpenStack Newton version [12]. An Ubuntu 16.04.4 server is used to implement each component of the system, namely, the PMs, the shared storage system, and the controller. Primary PM #1 and the backup PM are implemented by using servers, each of which equips Intel Xeon E5-2683 v4 CPU with 128 GB memory and 1 TB disk storage. Primary PM #2 is implemented by using a server, which equips Intel Xeon L3406 CPU with 8 GB memory and 250 GB disk storage.

### A. CONTROLLER

The controller offers four major functions, each of which are described as follows.

First, the user API provides an interface for customers to request the configuration of VMs. A customer selects the required performance of VM from among several options, which are called *flavors*, and an operating system for VM. The flavors differ in the number of CPU cores, the amount of memory, and the amount of root disk. Note that, in this prototype cloud system, the root disk is reserved in the internal storage of the primary PM only for the purpose of running an operating system and an application; a virtual storage

volume for storing output data is separately reserved in the shared storage volume.

Second, the resource allocation manager controls computation resources in the primary and backup PMs. When a customer requests the configuration of a VM, this manager reserves CPU cores and memory in a primary PM according to the selected flavor. This manager computes the allocation of computation resources of backup PMs to existing VMs and the VM to be launched, and then reserves CPU cores and memory in backup PMs based on the computation result.

Third, the image service stores image files, which are used for launching VMs in the primary or backup PMs. This image service, named *glance*, is one of the OpenStack core services. When a new VM is launched according to the configuration request, an image file of operating system that a customer specifies is copied to the primary PM. During the operation of VMs, snapshots of running VMs are periodically created and stored as image files. When a primary PM failure happens, image files which were created as snapshots of failed VMs are used to relaunch the VMs in the corresponding backup PMs.

Finally, the VM monitoring function watches running VMs in primary PMs to detect a failure. This function obtains the status of each VM periodically by using existing capabilities of the OpenStack framework. When the function detects simultaneous failures of VMs in one primary PM, the function presumes that the primary PM is failed and triggers the failure recovery process.

### B. PHYSICAL MACHINES

The primary and backup PMs equip computation resources such as CPU cores and memory to host VMs. The OpenStack compute service (*nova*) is installed to these PMs to manage VMs. In this prototype system, we set availability zones of PMs in order to launch a VM in a specific primary or backup PM.

During the process of launching a VM, the PM receives a copy of image file from the controller to launch a requested VM. The adequate amount of computation resources in the PM are reserved based on a flavor that a customer selected.

A PM implemented based on the OpenStack compute service equips an image cache to store image files which are once used to launch VMs. The PM uses a cached image file when it configures the same type of VM again. This mechanism enables the quick launch of VMs since there is no need to copy the image file from the controller again. The reduction of the VM relaunching time on a backup PM improves the availability that a cloud provider supports for customers. To achieve this, we have developed an optional function that periodically copies the latest snapshot of running VMs to the image cache of corresponding backup PM. This function is referred to as a prefetch function hereafter.

### C. SHARED STORAGE SYSTEM

The shared storage system provides a virtual storage volume for each VM. The OpenStack block storage service (*cinder*) is used to implement the shared storage system. As mentioned in the begging part of Section III, this prototype system assumes the use case that output data of an application in each VM are stored in a virtual storage volume. A virtual storage volume is configured and attached to a new VM by the controller. In the case of a PM failure, the controller reattaches virtual storage volumes to VMs relaunched in backup PMs.

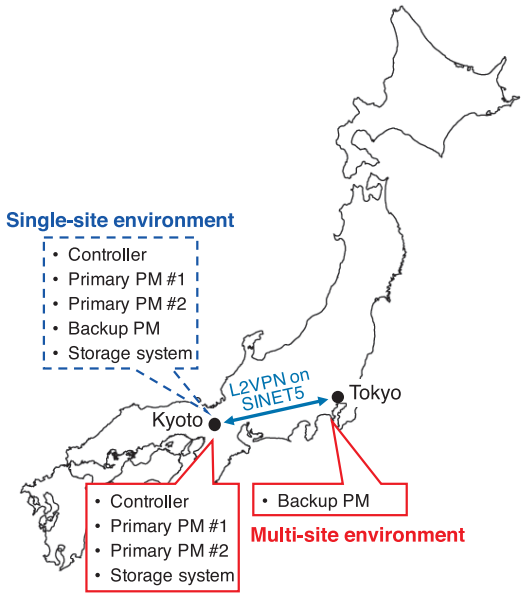



**FIGURE 3.** Geographic location of components in experiment.

## IV. EXPERIMENT

We conduct an experiment of VM failure recovery and VM snapshot creation by using the implemented prototype cloud system.

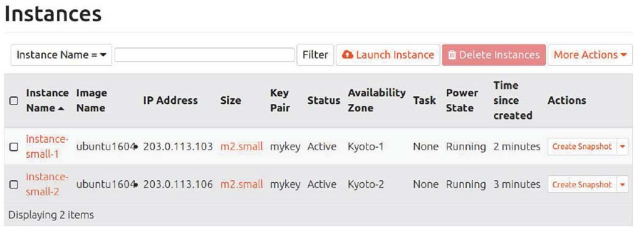
### A. FAILURE RECOVERY

At first, the failure recovery scenario based on the backup resource allocation strategy in Section II is examined. We define the failure recovery time as the time required for a failed VM to become available in the backup PM. The failure recovery time is measured under the condition that there is no storage access from each VM to the corresponding virtual storage volume.

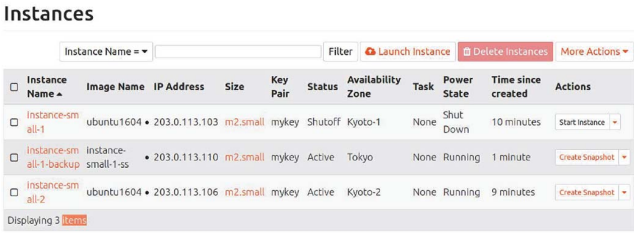
#### 1) ENVIRONMENT

We examine the failure recovery scenario in two environments, namely a single-site environment and a multi-site environment, in consideration of various types of deployments in the real cloud computing systems. Fig. 3 shows the deployment of the prototype cloud system components in the two environments, which differ in the geographic location of the backup PM.

The single-site environment deploys all components of the prototype cloud system on the same server rack in the Kyoto site. The management network is based on Gigabit Ethernet

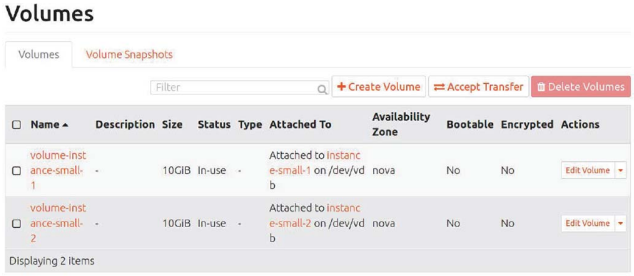



(a) Before failure recovery.

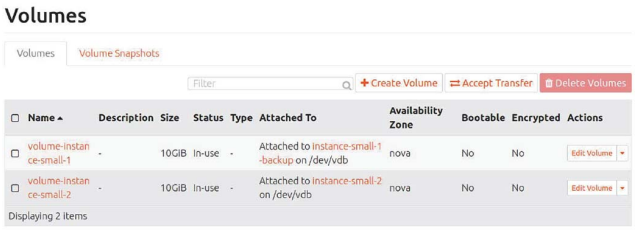



(b) After failure recovery.

**FIGURE 4.** Information of VM instances captured from OpenStack dashboard.




(a) Before failure recovery.




(b) After failure recovery.

**FIGURE 5.** Information of virtual storage volumes captured from OpenStack dashboard.

(1000BASE-T). Measured round trip times (RTT) between each pair of components are less than one millisecond.

The multi-site environment assumes the distributed cloud. The difference from the single-site environment is that the backup PM is deployed in the Tokyo site. The Kyoto site and the Tokyo site are connected by the Science Information Network 5 (SINET5) [13], which is a Japanese academic backbone network provided by the National Institute of Informatics [14]. We configure a layer-2 virtual private network (L2VPN) on SINET5 to extend the management network from Kyoto to Tokyo. The total optical fiber cable length in SINET 5 between Kyoto and Tokyo is 691 kilometers. The average RTT between the controller in Kyoto and the backup PM in Tokyo is 8.6 milliseconds.

A flavor with one CPU core, 1 GB memory, and 10 GB root disk is used in the experiment. An Ubuntu 16.04.4 desktop image (1.5 GB), a CentOS 7 image (4.2 GB), and a CirrOS 0.3.4 image (12.7 MB) are stored in the controller. One of the three images is selected and used in the experiment. The size of a virtual storage volume which is attached to each VM is set to 10 GB. A failure of primary PM is simulated by shutting down a VM running on the PM. We examined both cases that the prefetch function of VM snapshots is enabled and disabled.

The detail of the failure recovery scenario tested in this experiment is as follows.

(i) A single VM is launched in each primary PM. Virtual storage volumes are configured in the shared storage system, and each of them is attached to each VM.
(ii) Computation resource of the backup PM is shared by the two VMs.
(iii) A snapshot of each VM is periodically created and stored as image files in the controller. If the prefetch function is enabled, the snapshots are copied to the backup PM.
(iv) In parallel with (iii), the controller watches the two VMs to detect a failure.
(v) When the controller detects a failure of a primary PM, the controller launches a VM in the backup PM by using the latest snapshot of the VM in the failed PM. The same flavor is selected to launch the VM.
(vi) The controller detaches the virtual storage volume from a failed VM and reattaches the volume to the VM launched in the backup PM.

### 2) RESULTS

The failure recovery scenario described in Section IV-A1 is performed successfully in both single- and multi-site environments. Figs. 4 and 5 are screenshots of VM instance information and virtual storage volume information, respectively, captured from the OpenStack dashboard (*horizon*) installed on the controller. Figs. 4(a) and 5(a) are captured before the failure in primary PM #1. Figs. 4(b) and 5(b) are captured after the failure in primary PM #1. Here, primary PM #1, primary PM #2, and the backup PM belong to different availability zones, named *Kyoto-1*, *Kyoto-2*, and *Tokyo*, respectively.

Fig. 4(a) shows that in the initial state, VM instances named *instance-small-1* and *instance-small-2* are configured on the two primary PMs in *Kyoto-1* and *Kyoto-2* zones, respectively. An Ubuntu desktop image is used to configure the VMs. As shown in Fig. 5(a), 10 GB virtual storage volumes named *volume-instance-small-1* and *volume-instance-small-2* are attached to these VM instances. When a failure happens in primary PM #1 (*Kyoto-1* zone), the backup VM instance named *instance-small-1-backup* is configured on the backup PM in *Tokyo* zone as shown in Fig. 4(b). The image

**TABLE 1.** Failure recovery time in single-site environment.

| | Without prefetch | | | With prefetch | | |
|---|---|---|---|---|---|---|
| | Min. | Avg. | Max. | Min. | Avg. | Max. |
| Launching backup VM | 34.1 s | 34.9 s | 35.8 s | 15.7 s | 16.4 s | 17.3 s |
| Reattaching virtual storage volume | 15.2 s | 15.4 s | 15.5 s | 15.3 s | 15.6 s | 15.7 s |
| **Total failure recovery time** | **49.7 s** | **50.3 s** | **51.3 s** | **31.1 s** | **31.9 s** | **32.9 s** |

**TABLE 2.** Failure recovery time in multi-site environment.

| | Without prefetch | | | With prefetch | | |
|---|---|---|---|---|---|---|
| | Min. | Avg. | Max. | Min. | Avg. | Max. |
| Launching backup VM | 36.6 s | 37.5 s | 38.0 s | 19.7 s | 20.5 s | 21.5 s |
| Reattaching virtual storage volume | 15.4 s | 15.6 s | 15.7 s | 15.5 s | 15.7 s | 15.8 s |
| **Total failure recovery time** | **52.0 s** | **53.1 s** | **53.5 s** | **35.3 s** | **36.2 s** | **37.4 s** |

file named *instance-small-1-ss* is the snapshot of *instance-small-1* generated before the failure. As shown in Fig. 5(b), virtual storage volume *volume-instance-small-1* is reattached to *instance-small-1-backup*.

We measure the required time for performing the failure recovery scenario in the single- and multi-site environments. In both environments, we measure the failure recovery time in two policies, whether the prefetch function of VM snapshots is enabled or not. We use an Ubuntu desktop image to generate VMs in this measurement. We measure the failure recovery time from a failure in primary PM #1 five times. Specifically, we measure the required times for two operation sections in the failure recovery scenario: 1) from a failure detection to launching a backup VM and 2) the reattachment of the virtual storage volume to the backup VM. Note that we experimentally observe that the failure recovery time from a failure in primary PM #2 is basically the same as in the case of primary PM #1.

Tables 1 and 2 show the measurement results of failure recovery time in the single- and multi-site environments, respectively. The minimum, average, and maximum times obtained in the five measurements are shown in each table. Compared with the single-site environment, the recovery time becomes longer in the multi-site environment. This is because the larger RTT and the disturbance created by other traffic in SINET5 increase the transmission time between the controller to the backup PM. We can also observe that the prefetch function reduces the required time to launch a VM in the backup PM in both environments.

Fig. 6 shows the failure recovery time when different image files of operating systems are used in the prototype cloud system. This measurement result is obtained in the multi-site environment. When the prefetch function is disabled, the failure recovery time increases linearly to the size of the image file. On the other hand, when the prefetch function is enabled, the increasing rate of the failure recovery time is moderated. The prefetch function enables the recovery of failed VM within basically the same time regardless of the image file size.

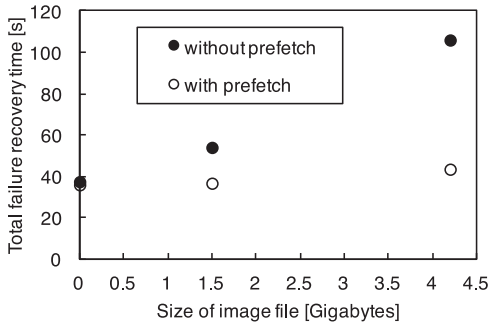



**FIGURE 6.** Failure recovery time when using different image files.

**TABLE 3.** VM snapshot creation time.

| | Min. | Avg. | Max. |
|---|---|---|---|
| Without storage access | 33.6 s | 39.0 s | 42.0 s |
| With continuous storage access | 43.4 s | 46.4 s | 49.8 s |

### B. VM SNAPSHOT CREATION TIME

Next, we examine the effect of storage access on the required time for creating a snapshot of running VM. The storage access from the VM to the virtual storage volume interrupts the process of creating a VM snapshot. This means that the storage access extends the VM snapshot creation time, which can lead to degradation of the availability of cloud system. In this section, we experimentally verify the effect of storage access by using the prototype cloud system. The creation time of VM snapshot, which is obtained through this test, can be used to determine the interval of VM snapshot creation; it determines the lower limit of interval of creating the VM snapshot.

#### 1) ENVIRONMENT

We measure the VM snapshot creation time when the access to the virtual storage volume exists and does not exist. In this measurement, we use a part of the prototype cloud system shown in Fig. 2: primary PM #1, the shared storage system, and the controller. All components are deployed on the same rack. A VM is configured on primary PM #1 by using the same flavor as the experiment in Section IV-A and an Ubuntu 16.04.4 desktop image. A 10 GB virtual storage volume is configured in the shared storage system and attached to the VM. Storage access from the VM to the virtual storage volume is realized by using Linux *dd* command with the direct I/O option; during the measurement, the VM continuously write zeros to the virtual storage volume. We measure the VM snapshot creation time ten times in both cases where the continuous storage access exists and does not exist.

#### 2) RESULTS

Table 3 shows the measured VM snapshot creation time with and without the continuous access to the virtual storage volume. It can be observed that the required time for VM snapshot creation increases when there is storage access. The extended time caused by continuous storage access is 7.4 seconds on average. The interval of creating VM snapshot

in the prototype cloud system should be set to a sufficiently larger value compared to the result in Table 3, i.e., 49.8 seconds.

## V. AVAILABILITY ANALYTICAL MODEL

We present a model to analyze the availability of the cloud system based on backup resource sharing and probabilistic protection guarantee. Specifically, we consider the availability of a service provided by using a VM in the cloud system; the service is available if the corresponding VM is running in either primary or backup PMs and an application in the VM generates output data. In this availability analysis, we incorporate the real-world cloud characteristics, such as the failure recovery time and the VM snapshot creation time, with the theoretical model of the probabilistic protection guarantee. The failure recovery following the procedure described in Section IV-A1 is referred to as "the prompt failure recovery" hereafter.

Generally, the availability of a system is defined as (1) by using Mean Time Between Failures (MTBF) and Mean Time To Repair (MTTR).

$$\text{Availability} = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}} \tag{1}$$

In this model, we assume that each VM runs an application which generates output data by using past data generated before. The output data are generated at a constant rate and stored in a virtual storage volume. When a VM in a failed primary PM is relaunched in a backup PM, since the snapshot used for relaunching the VM was created at a certain time point of the past, the application in the relaunched VM needs to regenerate the data which were once generated in the primary PM after the snapshot was created. We refer to the difference between the time point when the latest snapshot was created and the time point when the primary PM failed as *time gap* hereafter. We consider that MTTR of the service includes this data regeneration time, which has the same duration with the *time gap*, in addition to the failure recovery time of VM.

Let $W$ and $B$ be a set of primary PMs and a set of backup PMs, respectively. Let $L_k$ be a set of primary PMs $i \in W$, each of which is partially or totally protected by backup PM $k \in B$. When the probabilistic protection guarantee is applied, for backup PM $k \in B$, the probability that the recovery of failed primary PMs in $L_k$ does not succeed is not greater than a certain probability, namely $\epsilon$. Let $p_i$ denote the failure probability of primary PM $i \in W$. The probability that all primary PMs in $L_k$ work without any failure is $\prod_{i\in L_k}(1-p_i)$. The probability that one or more primary PMs in $L_k$ fail is $1-\prod_{i\in L_k}(1-p_i)$. According to the condition of probabilistic protection guarantee described above, the probability that the prompt failure recovery does not succeed by backup PM $k \in B$ is not greater than $\epsilon$. The probability that the prompt failure recovery is performed by backup PM $k \in B$ is not less than $1 - \epsilon - \prod_{i\in L_k}(1 - p_i)$. Let $T_\text{P}$ and $T_\text{R}$ represent the failure recovery times when the prompt failure recovery succeeds and does not succeed, respectively. $T_\text{P}$ consists of the times for launching a VM in a backup PM and reattaching a virtual storage volume to the VM. $T_\text{R}$ consists of the time required for manually setting up backup resources and $T_\text{P}$. In this model, we assume that the failure recoveries of multiple VMs can be processed in parallel; $T_\text{P}$ and $T_\text{R}$ are constant regardless of the number of VMs that need to be relaunched in a backup PM. The time interval between failure events of a primary PM is assumed to be sufficiently longer than $T_\text{R}$. Since $T_\text{P} < T_\text{R}$, we can derive the lower bound of availability by considering that the probability that the prompt failure recovery succeeds is at least $1 - \epsilon - \prod_{i\in L_k}(1 - p_i)$ and the probability that it does not succeed is at most $\epsilon$.

### A. MODEL FOR NON-PREFETCH POLICY

At first, we consider the case where the prefetch of VM snapshot is not executed. Let $\tau_\text{c}$ and $\tau_\text{i}$ represent the required times for the creation of VM snapshot and the interval of creating VM snapshots, respectively. We assume that $\tau_\text{i}$ is larger than $\tau_\text{c}$ so that multiple snapshots of the same VM are not created in parallel.

Fig. 7(a) shows the operations of the primary PM, the controller, and the backup PM in the timeline before the failure of primary PM happens. This figure is illustrated by focusing on one VM in the primary PM. A snapshot of the VM created at some point, namely $t$, becomes available in the controller at time $t + \tau_\text{c}$.

Fig. 7(b) shows the operation of each component when the failure of primary PM happens and the prompt failure recovery succeeds. The latest snapshot of the failed VM is copied from the controller to the backup PM just after the primary PM fails. After the prompt failure recovery, which takes $T_\text{P}$, the regeneration of data is executed. The required time for data regeneration, which has the same duration with the *time gap*, varies from $\tau_\text{c}$ to $\tau_\text{c}+\tau_\text{i}$ according to the timing of failure. The mean time of data regeneration is $\tau_\text{c} + \frac{\tau_\text{i}}{2}$.

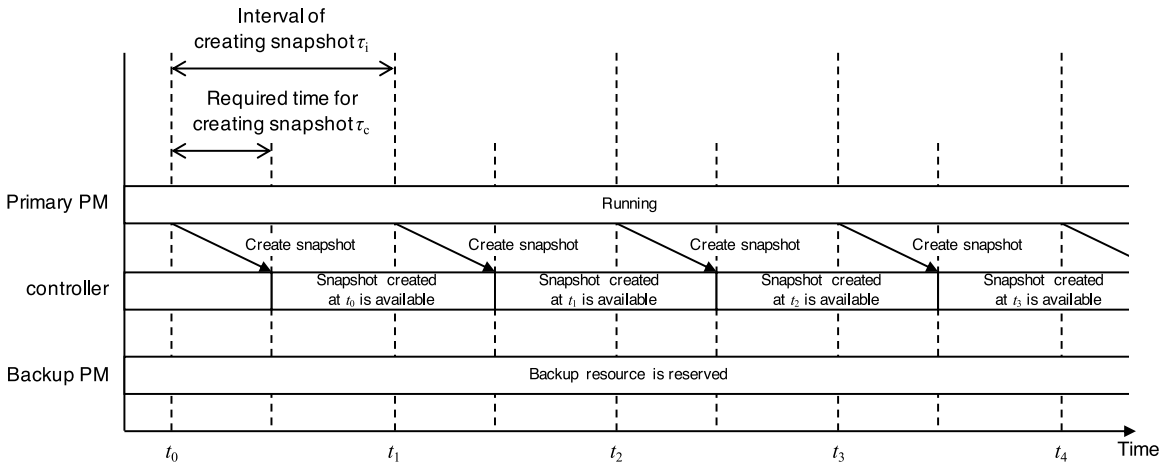
Therefore, by using the above symbols, MTTR in the case where the prefetch of VM snapshot is not executed can be expressed as

$$\begin{aligned}\text{MTTR} = &\left(1 - \epsilon - \prod_{i\in L_k}(1 - p_i)\right) \times \left(T_\text{P} + \tau_\text{c} + \frac{\tau_\text{i}}{2}\right) \\ &+ \epsilon \times \left(T_\text{R} + \tau_\text{c} + \frac{\tau_\text{i}}{2}\right).\end{aligned} \tag{2}$$

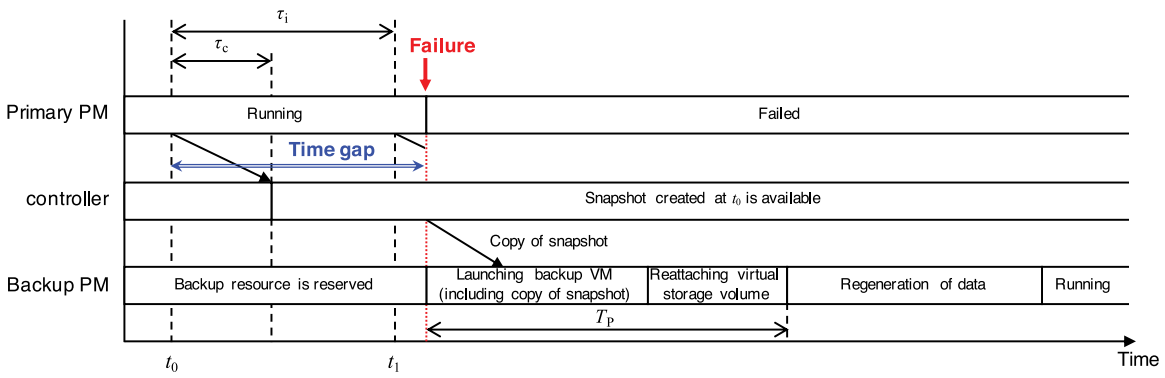
Note that the delay time for the detection of VM failure is not included in (2). In the case of considering this delay time, we can simply add a half time of the interval of monitoring each VM to the MTTR in (2).

### B. MODEL FOR PREFETCH POLICY

Next, we consider the case where the prefetch of VM snapshot is executed. Let $\tau_\text{f}$ represent the required time for the prefetch of VM snapshot from the controller to the backup PM. We assume that $\tau_\text{i}$ is larger than $\tau_\text{c} + \tau_\text{f}$ so that multiple snapshots of the same VM are not created and




(a) Before primary PM fails.




(b) When primary PM fails and prompt failure recovery succeeds.

**FIGURE 7.** Operations of primary PM, controller, and backup PM in case of non-prefetch policy.

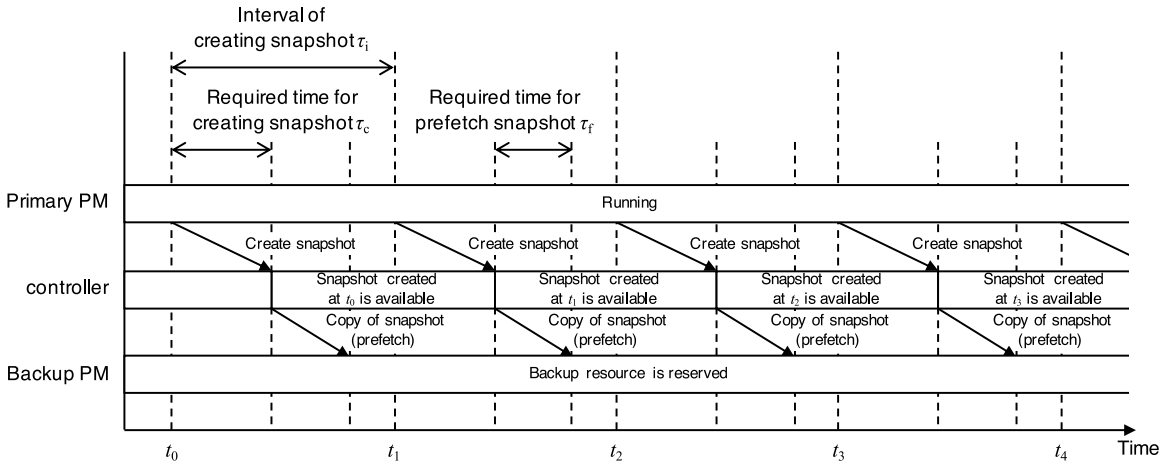
prefetched in parallel. This means that the backup VM is always configured by using the latest snapshot of the VM in the primary PM.

Fig. 8(a) shows the operations of the primary PM, the controller, and the backup PM in the timeline before the failure of primary PM happens. In the same way as Fig. 7(a), a snapshot of the VM created at time $t$ becomes available in the controller at time $t+\tau_c$. Then the snapshot is prefetched from the controller to the backup PM. The prefetch of snapshot is completed at $t + \tau_c + \tau_f$.

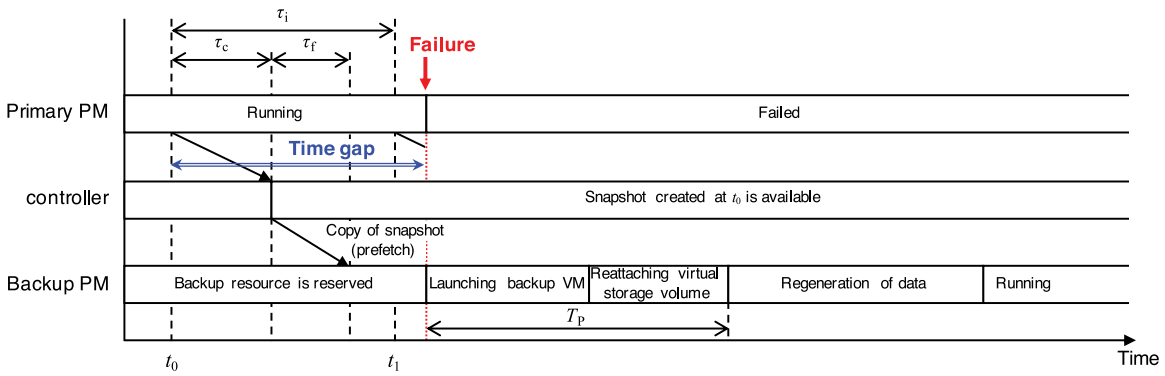
Figs. 8(b) and 8(c) show the operation of each component when the failure of primary PM happens and the prompt failure recovery succeeds. Fig. 8(b) is the operations when the failure happens in the time periods other than the execution of prefetch function. Fig. 8(c) is the operations when the failure happens in the time period of the execution of prefetch function. Under the assumption that the failure probability of primary PM does not depend on time, the probabilities that the situations of Fig. 8(b) and Fig. 8(c) happen are $1 - \frac{\tau_f}{\tau_i}$ and $\frac{\tau_f}{\tau_i}$, respectively. In the case of Fig. 8(b), the backup PM starts launching a backup VM by using the prefetched snapshot just after the primary PM fails. The required time for data regeneration, which has the same duration with the *time gap*, varies from $\tau_c + \tau_f$ to $\tau_i + \tau_c$. The mean time of data regeneration in this case is $\tau_c + \frac{\tau_f+\tau_i}{2}$. In the case of Fig. 8(c), a backup VM is launched after the prefetch of the latest snapshot is completed. Let $\delta$ represent the remaining time to complete the prefetch function. The required time for data regeneration, which has the same duration with the *time gap*, is $\tau_c + \tau_f - \delta$.

Therefore, MTTR in the case where the prefetch of VM snapshot is executed can be expressed as

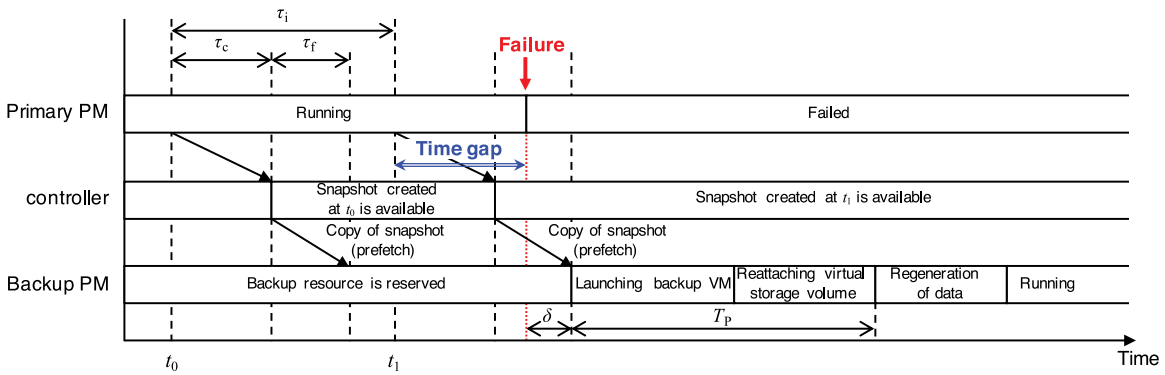
$$\begin{aligned}\mathrm{MTTR} = &\left(1 - \epsilon - \prod_{i \in L_k}(1 - p_i)\right) \\ &\times \left\{\left(\frac{\tau_f}{\tau_i}\right) \times (T_P + \tau_c + \tau_f) + \left(1 - \frac{\tau_f}{\tau_i}\right) \right. \\ &\left. \times \left(T_P + \tau_c + \frac{\tau_f + \tau_i}{2}\right)\right\} \\ &+ \epsilon \times \left\{\left(\frac{\tau_f}{\tau_i}\right) \times (T_R + \tau_c + \tau_f) + \left(1 - \frac{\tau_f}{\tau_i}\right) \right. \\ &\left. \times \left(T_R + \tau_c + \frac{\tau_f + \tau_i}{2}\right)\right\}. \end{aligned} \tag{3}$$




(a) Before primary PM fails.




(b) When primary PM fails during other than prefetch and prompt failure recovery succeeds.




(c) When primary PM fails during prefetch and prompt failure recovery succeeds.

**FIGURE 8.** **Operations of primary PM, controller, and backup PM in case of prefetch policy.**

### *C. CASE STUDIES*

We provide case studies of availability analysis based on the experiment results obtained in Section IV. The failure recovery time imposed when the prompt failure recovery succeeds, $T_P$, corresponds to the results of experiment in Section IV-A. The required time for VM snapshot creation, $\tau_c$, corresponds to the results of experiment in Section IV-B.

We study two configurations of cloud system in this paper. The first configuration is the prototype cloud system, which includes two primary PMs and one backup PM, i.e., $|W| = 2$ and $|B| = 1$. The backup PM protects both primary PMs, i.e., $|L_1| = 2$. The second configuration is the cloud system including six primary PMs and three backup PMs, i.e., $|W| = 6$ and $|B| = 3$. We assume that, as a result of backup resource allocation, each backup PM protects five primary

**TABLE 4.** Required $\tau_i$ in order to achieve certain level of availability in case of $|W| = 2$, $|B| = 1$, $|L_k| = 2$ (prototype cloud system).

| Environment and policy | Target availability | $\epsilon = 0.0001$ | $\epsilon = 0.0005$ | $\epsilon = 0.001$ | $\epsilon = 0.005$ | $\epsilon = 0.01$ |
|---|---|---|---|---|---|---|
| Single-site, Non-prefetch | 99.99995% | 639.3 s | 351.8 s | – | – | – |
| | 99.9999% (six nines) | 1543.8 s | 1256.4 s | 897.1 s | – | – |
| | 99.9995% | 8780.0 s | 8492.6 s | 8133.3 s | 5259.1 s | 1666.2 s |
| | 99.999% (five nines) | 17825.4 s | 17538.0 s | 17178.7 s | 14304.4 s | 10711.6 s |
| Multi-site, Non-prefetch | 99.99995% | 633.7 s | 346.4 s | – | – | – |
| | 99.9999% (six nines) | 1538.2 s | 1250.9 s | 891.8 s | – | – |
| | 99.9995% | 8774.4 s | 8487.1 s | 8128.0 s | 5254.9 s | 1663.5 s |
| | 99.999% (five nines) | 17819.8 s | 17532.5 s | 17173.4 s | 14300.2 s | 10708.8 s |
| Single-site, Prefetch | 99.99995% | 675.4 s | 386.8 s | – | – | – |
| | 99.9999% (six nines) | 1580.2 s | 1292.0 s | 931.7 s | – | – |
| | 99.9995% | 8816.6 s | 8528.4 s | 8168.2 s | 5286.5 s | 1684.3 s |
| | 99.999% (five nines) | 17862.0 s | 17573.8 s | 17213.6 s | 14331.9 s | 10729.9 s |
| Multi-site, Prefetch | 99.99995% | 666.9 s | 378.6 s | – | – | – |
| | 99.9999% (six nines) | 1571.7 s | 1283.6 s | 923.6 s | – | – |
| | 99.9995% | 8808.0 s | 8520.0 s | 8160.1 s | 5280.1 s | 1680.1 s |
| | 99.999% (five nines) | 17853.4 s | 17565.4 s | 17205.4 s | 14325.5 s | 10725.6 s |

**TABLE 5.** Required $\tau_i$ in order to achieve certain level of availability in case of $|W| = 6$, $|B| = 3$, $|L_k| = 5$.

| Environment and policy | Target availability | $\epsilon = 0.0001$ | $\epsilon = 0.0005$ | $\epsilon = 0.001$ | $\epsilon = 0.005$ | $\epsilon = 0.01$ |
|---|---|---|---|---|---|---|
| Single-site, Non-prefetch | 99.99995% | 144.7 s | (28.0 s) | – | – | – |
| | 99.9999% (six nines) | 512.0 s | 395.3 s | 249.4 s | – | – |
| | 99.9995% | 3450.2 s | 3333.5 s | 3187.6 s | 2020.5 s | 561.7 s |
| | 99.999% (five nines) | 7122.9 s | 7006.2 s | 6860.4 s | 5693.3 s | 4234.5 s |
| Multi-site, Non-prefetch | 99.99995% | 139.1 s | (22.4 s) | – | – | – |
| | 99.9999% (six nines) | 506.4 s | 389.7 s | 243.9 s | – | – |
| | 99.9995% | 3444.6 s | 3327.9 s | 3182.1 s | 2015.5 s | 557.2 s |
| | 99.999% (five nines) | 7117.4 s | 7000.7 s | 6854.9 s | 5688.3 s | 4230.0 s |
| Single-site, Prefetch | 99.99995% | 179.5 s | (58.6 s) | – | – | – |
| | 99.9999% (six nines) | 548.1 s | 430.9 s | 284.2 s | – | – |
| | 99.9995% | 3486.8 s | 3369.8 s | 3223.5 s | 2053.4 s | 590.4 s |
| | 99.999% (five nines) | 7159.6 s | 7042.6 s | 6896.4 s | 5726.3 s | 4263.7 s |
| Multi-site, Prefetch | 99.99995% | 171.1 s | (50.1 s) | – | – | – |
| | 99.9999% (six nines) | 539.6 s | 422.5 s | 276.0 s | – | – |
| | 99.9995% | 3478.2 s | 3361.3 s | 3215.1 s | 2045.7 s | 583.6 s |
| | 99.999% (five nines) | 7151.0 s | 7034.1 s | 6887.9 s | 5718.6 s | 4256.8 s |

PMs, i.e., $|L_1| = |L_2| = |L_3| = 5$. An IaaS provider sets $\epsilon$ in order to support a certain level of availability. Among the parameters introduced in Sections V-A and V-B, the provider is able to decide the interval of creating snapshots, $\tau_i$.

The time required for setting up another PM when the prompt failure recovery does not succeed, which is included in $T_R$, depends on the management structure of the cloud computing system and this is out of the scope of this paper. $p_i, \forall i \in W$, $T_R$, and MTBF depend on the hardware and software performance. In this case study, we set the failure probability of each primary PM, $p_i$, to 0.01. We set the values of $T_R$ and MTBF of the cloud system to 2 hours and 5,000 hours, respectively.

We set $T_P$ to the average failure recovery time obtained in Section IV-A; for example, $T_P$ is set to 50.3 seconds in the case of single-site environment and non-prefetch policy. Let us assume that there is continuous storage access from the VM to the virtual storage volume. From the experiment results in Section IV-B, $\tau_c$ is set to 46.4 seconds. The required time for the prefetch of VM snapshot, $\tau_f$, is assumed to be the difference of the launching time of backup VM in the non-prefetch policy and that of the prefetch policy; we set $\tau_f = 34.9 - 16.4 = 18.5$ in the single-site environment and $\tau_f = 37.5 - 20.5 = 17.0$ in the multi-site environment.

Tables 4 and 5 show the values of $\tau_i$ required to achieve a certain level of availability, which are calculated by using (1), (2), and (3), in the two configurations. Note that "–" means that the target availability cannot be achieved only by changing $\tau_i$. Values of $\tau_i$ enclosed in parentheses do not satisfy the assumptions of $\tau_i > \tau_c$ (in the non-prefetch policy) and $\tau_i > \tau_c + \tau_f$ (in the prefetch policy).

The availability that the cloud provider can support increases by selecting small values for $\epsilon$ and $\tau_i$. Compared to the single-site environment, the multi-site environment requires shorter $\tau_i$ to achieve a certain level of availability due to the increase in failure recovery time. It is also observed that the value of $\tau_i$ can be relaxed by adopting the prefetch function; it helps the cloud provider to reduce computation and traffic loads of snapshotting.

The second configuration, where each backup PM protects more primary PMs than the first configuration, requires shorter $\tau_i$ to achieve a certain level of availability. For example, when the cloud provider sets $\epsilon = 0.001$ and aims to achieve 99.9999% (six nines) of availability, a snapshot needs to be created about every 15 minutes in the first scenario, while it needs to be created about every 4 minutes in the second scenario.

### D. DISCUSSIONS

The availability analytical model can be adopted to larger cloud systems, i.e., cloud systems consisting of more primary and backup PMs, than those used in the case studies in Section V-C. As seen in (2) and (3), MTTR of a service depends on the size of $L_k$, which is a set of primary PMs protected by backup PM $k \in B$. As $|L_k|$ increases, the MTTR becomes longer and the availability deteriorates as a result. This means that, to support a certain level of availability in a large-scale cloud system, $|L_k|$ needs to be kept small. The backup resource allocation can be computed by using a heuristic approach such as simulated annealing for large-scale cloud systems [7]. A constraint that determines the upper limit of $|L_k|$ needs to be introduced into the procedure of backup resource allocation computation.

We used the experiment results of failure recovery time and VM snapshot creation time as the values of $T_{\mathrm{P}}$ and $\tau_{\mathrm{c}}$. As $T_{\mathrm{P}}$ and $\tau_{\mathrm{c}}$ increase, the MTTR becomes longer and the availability deteriorates. The value of $T_{\mathrm{P}}$ depends on the implementation of cloud system, including the PM locations (i.e., single-site or multi-site), the prefetch policy, and the VM image file size. $\tau_{\mathrm{c}}$ can vary depending on the frequency of storage access from a VM to a virtual storage volume. In the case studies, we set $\tau_{\mathrm{c}}$ to the VM snapshot creation time measured under the existence of continuous storage access, which can be considered as an extreme case. If a service provided by the VM makes less storage access, we can set $\tau_{\mathrm{c}}$ to a smaller value and relax the interval of creating VM snapshots, $\tau_{\mathrm{i}}$, to achieve a certain level of availability.

## VI. RELATED WORKS

With the wide spread of cloud services, a great deal of effort has been made on researches on availability and survivability of cloud computing systems. Since the implementation and operation costs of redundant backup resources directly affect the service usage fee, various high availability methods with cost effectiveness have been studied.

Li *et al.* [15] presented the 1:M VM backup approach based on the idea of time sharing. A single backup VM provides backup services for multiple failed VMs by using different time slots so as to improve the utilization of backup VM. Zheng *et al.* [16] presented a framework for building fault-tolerant cloud applications with the limited amount of redundant components. The framework creates a ranking of application components for which fault-tolerant mechanisms should be provided. Based on the ranking, the framework automatically determines an optimal fault-tolerant strategy.

Several works presented a VM placement method which provides $k$-fault tolerance for cloud services [17], [18], [19]. $k$-fault tolerance means that the cloud is able to continue providing the service when $k$ failures happen simultaneously. Machida *et al.* [17] presented a redundant VM placement method in consolidated server systems in which multiple VMs are deployed to provide an application. VM placement of each application is determined so that the minimum number of VMs required to provide the service are alive in the case of any $k$ failures of host servers. Zhou *et al.* [18] presented another VM placement approach which tolerates simultaneous failures of $k$ VMs. This approach places all primary VMs and $k$ backup VMs in different host servers. A placement of VMs is provided so that a task of failed primary VM is reassigned to a backup VM without consuming much data transfer time and network resources. The authors evaluated the effectiveness of the approach by using their own experimental platform [19]. In contrast to these works, the backup resource allocation strategy considered in this paper determines the level of availability by setting the probability of unsuccessful recovery, $\epsilon$, instead of $k$.

There also have been several existing works on the availability analysis of IaaS cloud systems [20], [21], [22], [23]. Longo *et al.* [20], [21] presented an availability analysis method based on stochastic analytic models. In order to make the method applicable for a large scale IaaS cloud, they decompose the availability model into interacting sub-models based on Markov chain. Undheim *et al.* [22] developed an availability model for a cloud data center including the network which connects the data center to customers. The developed model considers the physical location of replicated VMs deployed in separated data centers. Jhawar and Piuri [23] studied a fault tolerance mechanism for cloud applications composing multiple virtualized components such as servers and power distributions. The mechanism took into account the failure behavior of each components. Based on this mechanism, the authors presented a methodology to select appropriate fault tolerance techniques for each user's requirement.

Fan *et al.* [24] studied a server allocation problem in a datacenter network which provides network services with certain availability for users. A network service is represented by a service function chain (SFC) consisting of multiple virtualized network functions (VNFs). Their framework iteratively allocates VMs for running VNFs to SFC requests until their availability requirements are satisfied. The availability model in [24] assumes that state changes (i.e., failure and repair) of network devices follow an alternating renewal process, and the switching time from primary VNFs to their corresponding backup VNFs is considered to be zero. On the other hand, the availability analytical model in this paper considers the failure recovery time, i.e., the required time to switch a primary PM to the corresponding backup PM when a failure occurs. The failure recovery time varies depending on whether the prompt failure recovery succeeds or not. Therefore, the availability model in [24] cannot be directly applied to the analysis of the system that we consider.

Kanizo *et al.* [25], [26] presented network function virtualization (NFV)-based backup scheme for middlebox functions. The availability models considered in [25] and [26] determine an assignment between backup servers and middlebox functions, where each server has a capability to support several functions and recover at most one of the functions when a function failure occurs. In [25], two

optimization problems were studied under the condition where the failure probability of each function is given; the objective of one problem is to maximize the probability for a recovery of all failed functions, and that of the other problem is to maximize the expected number of running functions. In [26], a problem to obtain an assignment that guarantees a recovery from any pattern of $t$ function failures, where $t = 1, 2, 3,$ and 4, was studied. The authors derived some properties of the problems from a graph representation of an assignment between servers and functions. A major difference between the models in [25], [26] and that in this paper lies in how primary resources fail. The models in [25] and [26] assume that middlebox functions fail individually; there is no correlation between failures of different functions. The availability analytical model in this paper considers failure of primary PMs, each of which can contain multiple VMs. When a primary PM fails, VMs which formerly run in the primary PM need to be relaunched in backup PMs at the same time. Therefore, the models in [25] and [26] cannot be directly applied to the analysis of the system that we consider. On the other hand, the system in this paper do not consider function types of VMs; this expansion is left for a future work.

Zahavi *et al.* [27] introduced a cloud architecture named *links as a service* (LaaS), which provides an isolated network in addition to dedicated servers to each tenant in a shared cloud system. An algorithm presented in [27] finds an allocation of servers and links to a new tenant in an online manner so that it is guaranteed that each tenant obtains the same bandwidth and delay without being affected by other tenants. The allocation of network resources in the cloud system considered in this paper should be also addressed as a future work.

## VII. CONCLUSION

This paper presented the experimental study of the backup resource allocation strategy based on backup resource sharing and probabilistic protection guarantee on the cloud computing system. We implemented an OpenStack-based prototype cloud system and examined the failure recovery scenario based on the strategy. The required time for failure recovery was measured in two geographic environments and two prefetch policies. We developed the availability analytical model for the backup resource allocation strategy. The case studies using the measurement results of the experiment showed that the availability can be derived from the probability of unsuccessful recovery, $\epsilon$, which is one of the configurable parameters in the strategy, and can be adjusted by setting the interval of creating VM snapshots, $\tau_i$.

## REFERENCES


[1] J. Peng, X. Zhang, Z. Lei, B. Zhang, W. Zhang, and Q. Li, "Comparison of several cloud computing platforms," in *Proc. 2nd Int. Symp. Inf. Sci. Eng.*, Dec. 2009, pp. 23–27.

[2] S. T. Maguluri, R. Srikant, and L. Ying, "Heavy traffic optimal resource allocation algorithms for cloud computing clusters," *Perform. Eval.*, vol. 81, pp. 20–39, Nov. 2014.

[3] M. Hadji and D. Zeghlache, "Minimum cost maximum flow algorithm for dynamic resource allocation in clouds," in *Proc. IEEE 5th Int. Conf. Cloud Comput.*, Jun. 2012, pp. 876–882.

[4] A. Khajeh-Hosseini, D. Greenwood, and I. Sommerville, "Cloud migration: A case study of migrating an enterprise IT system to IaaS," in *Proc. IEEE 3rd Int. Conf. Cloud Comput.*, Jul. 2010, pp. 450–457.

[5] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1107–1117, Jun. 2013.

[6] D. Ardagna, G. Casale, M. Ciavotta, J. F. Pérez, and W. Wang, "Quality-of-service in cloud computing: Modeling techniques and their applications," *J. Internet Serv. Appl.*, vol. 5, p. 11, Feb. 2014.

[7] F. He, T. Sato, B. C. Chatterjee, T. Kurimoto, S. Urushidani, and E. Oki, "Robust optimization model for backup resource allocation in cloud provider," in *Proc. IEEE Int. Conf. Commun.*, May 2018, pp. 1–6.

[8] T. Sato, F. He, E. Oki, T. Kurimoto, and S. Urushidani, "Implementation and testing of failure recovery based on backup resource sharing model for distributed cloud computing system," in *Proc. IEEE 7th Int. Conf. Cloud Netw.*, Oct. 2018, pp. 1–3.

[9] The OpenStack Foundation. *OpenStack*. Accessed: May 14, 2018. [Online]. Available: https://www.openstack.org/

[10] Í. Goiri, F. Julià, J. Guitart, and J. Torres, "Checkpoint-based fault-tolerant infrastructure for virtualized service providers," in *Proc. IEEE Netw. Oper. Manag. Symp.*, Apr. 2010, pp. 455–462.

[11] M. Johnston, H.-W. Lee, and E. Modiano, "A robust optimization approach to backup network design with random failures," *IEEE/ACM Trans. Netw.*, vol. 23, no. 4, pp. 1226–1228, Aug. 2015.

[12] *OpenStack Newton*. Accessed: May 14, 2018. [Online]. Available: https://www.openstack.org/software/newton/

[13] *Science Information NETwork 5*. Accessed: May 14, 2018. [Online]. Available: https://www.sinet.ad.jp/en/top-en

[14] *National Institute of Informatics*. Accessed: May 14, 2018. [Online]. Available: https://www.nii.ac.jp/en/

[15] X. Li, Y. Qi, P. Chen, and X. Zhang, "Optimizing backup resources in the cloud," in *Proc. IEEE 9th Int. Conf. Cloud Comput.*, Jun. 2016, pp. 790–797.

[16] Z. Zheng, T. C. Zhou, M. R. Lyu, and I. King, "Component ranking for fault-tolerant cloud applications," *IEEE Trans. Serv. Comput.*, vol. 5, no. 4, pp. 540–550, Nov. 2012.

[17] F. Machida, M. Kawato, and Y. Maeno, "Redundant virtual machine placement for fault-tolerant consolidated server clusters," in *Proc. IEEE Netw. Oper. Manag. Symp.*, Apr. 2010, pp. 32–39.

[18] A. Zhou *et al.*, "Cloud service reliability enhancement via virtual machine placement optimization," *IEEE Trans. Serv. Comput.*, vol. 10, no. 6, pp. 902–913, Nov./Dec. 2017.

[19] A. Zhou, S. Wang, C. Yang, L. Sun, Q. Sun, and F. Yang, "FTCloudSim: Support for cloud service reliability enhancement simulation," *Int. J. Web Grid Serv.*, vol. 11, no. 4, pp. 347–361, 2015.

[20] F. Longo, R. Ghosh, V. K. Naik, and K. S. Trivedi, "A scalable availability model for infrastructure-as-a-service cloud," in *Proc. IEEE/IFIP 41st Int. Conf. Depend. Syst. Netw.*, Jun. 2011, pp. 335–346.

[21] R. Ghosh, F. Longo, F. Frattini, S. Russo, and K. S. Trivedi, "Scalable analytics for IaaS cloud availability," *IEEE Trans. Cloud Comput.*, vol. 2, no. 1, pp. 57–70, Jan.–Mar. 2014.

[22] A. Undheim, A. Chilwan, and P. Heegaard, "Differentiated availability in cloud computing SLAs," in *Proc. IEEE/ACM 12th Int. Conf. Grid Comput.*, Sep. 2011, pp. 129–136.

[23] R. Jhawar and V. Piuri, "Fault tolerance management in IaaS clouds," in *Proc. IEEE 1st AESS Eur. Conf. Satellite Telecommun.*, Oct. 2012, pp. 1–6.

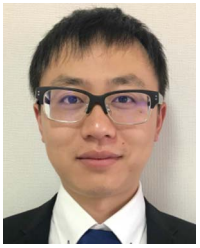
[24] J. Fan *et al.*, "A framework for provisioning availability of NFV in data center networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2246–2259, Oct. 2018.

[25] Y. Kanizo, O. Rottenstreich, I. Segall, and J. Yallouz, "Optimizing virtual backup allocation for middleboxes," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 2759–2772, Oct. 2017.

[26] Y. Kanizo, O. Rottenstreich, I. Segall, and J. Yallouz, "Designing optimal middlebox recovery schemes with performance guarantees," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2373–2383, Oct. 2018.

[27] E. Zahavi, A. Shpiner, O. Rottenstreich, A. Kolodny, and I. Keslassy, "Links as a service (LaaS): Guaranteed tenant isolation in the shared cloud," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1072–1084, May 2019.

**TAKEHIRO SATO** (Member, IEEE) received the B.E., M.E., and Ph.D. degrees in engineering from Keio University, Japan, in 2010, 2011, and 2016, respectively. He is currently an Assistant Professor with the Graduate School of Informatics, Kyoto University, Japan. From 2011 to 2012, he was a Research Assistant with the Keio University, Global COE Program by Ministry of Education, Culture, Sports, Science and Technology, Japan. From 2012 to 2015, he was a Research Fellow with the Japan Society for the Promotion of Science. From 2016 to 2017, he was a Research Associate with the Graduate School of Science and Technology, Keio University.



**FUJUN HE** (Student Member, IEEE) received the B.E. and M.E. degrees from the University of Electronic Science and Technology of China, Chengdu, China, in 2014 and 2017, respectively. He is currently pursuing the Ph.D. degree with Kyoto University, Kyoto, Japan. He was an Exchange Student with the University of Electro-Communications, Tokyo, Japan, from 2015 to 2016. His research interests include modeling, algorithm, optimization, resource allocation, survivability, and optical networks.

**EIJI OKI** (Fellow, IEEE) received the B.E. and M.E. degrees in instrumentation engineering and the Ph.D. degree in electrical engineering from Keio University, Yokohama, Japan, in 1991, 1993, and 1999, respectively. He was with Nippon Telegraph and Telephone Corporation Laboratories, Tokyo, from 1993 to 2008, and with the University of Electro-Communications, Tokyo, from 2008 to 2017. From 2000 to 2001, he was a Visiting Scholar with the Polytechnic Institute of New York University, Brooklyn. In 2017, he joined Kyoto University, Japan, where he is currently a Professor. His research interests include routing, switching, protocols, optimization, and traffic engineering in communication and information networks.

**TAKASHI KURIMOTO** (Member, IEEE) is an Associate Professor with the National Institute of Informatics (NII), Japan. From 1996 to 2014, he worked for NTT, where he was engaged in both network planning and research and development of next generation network. In 2015, he joined NII and involved in the design and implementation of national research and educational network. His research interests are the switching technology for high-speed computer networks and the next generation network.

**SHIGEO URUSHIDANI** (Member, IEEE) received the B.E. and M.E. degrees from Kobe University in 1983 and 1985, respectively, and the Ph.D. degree from the University of Tokyo in 2002. He is a Deputy Director General and a Professor with the National Institute of Informatics (NII), Japan. From 1985 to 2006, he worked for NTT, where he was engaged in the research and development of ATM, AIN, IP/MPLS, and optical switching systems. In 2006, he moved to NII, where he is currently involved in the design and implementation of the science information network, as well as in the research and development on future network architectures.