IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 14, 2019, accepted December 28, 2019, date of publication January 1, 2020, date of current version January 10, 2020.

Digital Object Identifier 10.1109/ACCESS.2019.2963397

Analyzing Software Rejuvenation Techniques in a Virtualized System: Service Provider and User Views

JING BAI[1], XIAOLIN CHANG[1], FUMIO MACHIDA[2], KISHOR S. TRIVEDI[3], AND ZHEN HAN[1]

[1]Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, Beijing 100044, China
[2]Department of Computer Science, University of Tsukuba, Tsukuba 305-8577, Japan
[3]Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708, USA

Corresponding author: Xiaolin Chang (xlchang@bjtu.edu.cn)

The work of Jing Bai and Xiaolin Chang was supported in part by the National Natural Science Foundation of China under Grant U1836105 and in part by the Fundamental Research Funds for the Central Universities of China under Grant 2018JBZ103. The work of Kishor S. Trivedi was supported by the National Natural Science Foundation of China under Grant 61872169.

ABSTRACT Virtualization technology has promoted the fast development and deployment of cloud computing, and is now becoming an enabler of Internet of Everything. Virtual machine monitor (VMM), playing a critical role in a virtualized system, is software and hence it suffers from software aging after a long continuous running as well as software crashes due to elusive faults. Software rejuvenation techniques can be adopted to reduce the impact of software aging. Although there existed analytical model-based approaches for evaluating software rejuvenation techniques, none analyzed both application service (AS) availability and job completion time in a virtualized system with live virtual machine (VM) migration. This paper aims to quantitatively analyze software rejuvenation techniques from service provider and user views in a virtualized system deploying VMM reboot and live VM migration techniques for rejuvenation, under the condition that all the aging time, failure time, VMM fixing time and live VM migration time follow general distributions. We construct an analytical model by using a semi-Markov process (SMP) and derive formulas for calculating AS availability and job completion time. By analytical experiments, we can obtain the optimal migration trigger intervals for achieving the approximate maximum AS availability and the approximate minimum job completion time, and then service providers can make decisions for maximizing the benefits of service providers and users by adjusting parameter values.

INDEX TERMS Semi-Markov process, software aging, software rejuvenation, server virtualization technology.

I. INTRODUCTION

Server virtualization (SV) and operating system (OS) virtualization technologies have been widely used in various fields. SV technology allows that virtual machines (VMs) with different OS (namely, different OS kernels) can run on a single physical machine (PM). Different from SV technology, OS virtualization technology enables a single OS kernel to support multiple isolated user-space instances [1]. PMs in cloud datacenters (CDCs) usually use SV/OS virtualization technology to be capable of constantly running online services and tolerating varying user workloads [2] with critical demands of high availability [3]. SV/OS virtualization technology is also explored for achieving dynamic network resource management [4] in Network Function Virtualization, which is essential for deploying 5G networks [4]. In addition, with the fast development of Internet of Things (IoT) technology, much more delay-sensitive IoT applications like complex event processing and streaming video need to be processed in Edge Computing (EC) [5], [6]. Live VM [7] or live container [8] migration can help achieve efficient allocation of resources like memory and network bandwidth in EC [9].

Without loss of generality, this paper focuses on the SV technology for delivering services. That is, each service runs in each independent VM [7] and then live VM

The associate editor coordinating the review of this manuscript and approving it for publication was Lei Wu.

migration can be used to migrate services between PMs, helping users receive high Quality of Service (It is a combination of attributes of service and is used for evaluating services from the perspective of server providers [10].) when their hosted PMs cannot work [11]. Live VM migration and service migration are used interchangeably in the rest of this paper.

Virtual machine monitor (VMM), which can allow multiple VMs to share the same physical machine safely [12], plays a critical role in an SV-based system. However, it is software and then is subject to software aging [13] (a phenomenon of software performance degradation after a long continuous running [14]) and crash due to certain elusive faults [15]. That is, software aging can degrade application service (AS) availability (which is defined as percentage of normal service provision time for the system, in other words, percentage of system available time) and then degrade user Quality of Experience. Here, QoE denotes the user's expected experience effect [16]. In this paper, AS is assumed to be a constantly running service, through which each user can execute his/her job. Amazon Web Services (AWS) Greengrass, as an instance of SV-based system hosting ASs, has suffered from failures that affected AS availability [17], and promises to provide at least 99.9% of the normal monthly uptime percentage for each AWS region [18]. It is obvious that the degradation of AS availability can lead to the increase in the overall job completion time (namely, the total required time to complete a job including the downtime). However, maximizing AS availability does not always mean the achievement of the minimum job completion time, since the AS state affects the job processing rate. For example, job processing rate will be reduced when the running system is in software aging. It is necessary to analyze both AS availability from the service provider view and job completion time from the user view such that service providers can decide when to trigger live VM migration technique for maximizing the benefits of service providers and users, respectively.

Software rejuvenation (a proactive solution for preventing the faults due to software aging) techniques can reduce the impact of software aging [19]. They have already been offered in production clouds, such as Azure [20] and VMware ESXi [21]. Analytical modeling is an effective approach for evaluating the software rejuvenation techniques in terms of AS availability and job completion time. There were various analytical model-based analysis of job performance and/or availability degradation caused by software aging in CDCs. They often assumed that all the time intervals follow exponential distributions [22]–[27]. Moreover, while many studies focused on analyzing either AS availability or job completion time [28]–[30], a few papers analyzed both of AS availability and job completion time together in a system [31], [32]. In particular, none of the existing analytical models has analyzed both AS availability from the service provider view and job completion time from the user view in an SV-based system using live VM migration.

In this paper, we consider an SV-based system composed of hosts for executing a job and supporting live VM migration among the hosts. VMM, running in every host, is software that is subject to software aging after a long continuous running [13]. The system deploys VMM reboot and live VM migration techniques based on the time-based rejuvenation after software aging detection. Aging time, failure time, VMM fixing time and live VM migration time follow general distributions. For such an SV-based system, we quantitatively investigate the impact of the software rejuvenation techniques on AS availability and job completion time. To analyze the state transitions of the SV-based system deploying VMM reboot and live VM migration techniques, we construct a semi-Markov process (SMP) which is used to evaluate the measures of interest. To the best of our knowledge, it is the first time to quantitatively analyze AS availability and job completion time in the aforementioned SV-based system. The main contributions are summarized as follows:

- We propose a SMP model for capturing the behaviors of a complicated SV-based system deploying VMM reboot and live VM migration techniques for rejuvenation. Our model can capture the detailed aging process and consider job processing rate during VMM aging thereby job completion time can be calculated more accurately.
- We derive formulas for calculating AS availability and job completion time in order to analyze software rejuvenation techniques from the view of service providers and users quantitatively.
- We conduct analytical experiments for analyzing AS availability and job completion time over a variety of system parameters. Analytical experiments are also carried out to determine migration trigger intervals for achieving the approximate optimal AS availability and job completion time.

The rest of the paper is organized as follows. In the second section, we discuss related work. Section III describes the system considered in this paper and presents a SMP model for analyzing AS availability and job completion time. Section IV presents the results of analytical experiments. Finally, the conclusion is drawn and future work is discussed in Section V.

II. RELATED WORK

The past years witnessed significant efforts made for analytical model-based evaluation of AS availability and/or job completion time. See [13], [22]–[31] and references therein.

Changa *et al.* [22] proposed the continuous time Markov chain (CTMC) model for analyzing VM survivability in the system, where VM failover and live VM migration techniques were applied to improve service survivability. They also quantitatively compared the capability of rejuvenation techniques in an SV-based system in [13]. Okamura and Dohi [23] proposed a phase-expanded software rejuvenation model in order to investigate the interval reliability and solved it by reducing the model to a CTMC model. Rahme and Xu [24] presented an extended Dynamic Fault Tree model to calculate the system reliability and used CTMC technique to verify the capability of their approach.

Machida and Miyoshi [25] modeled the system with condition-based rejuvenation as an M/M/1 queue for the rejuvenation decision. Nguyen *et al.* [26] presented a Stochastic Reward Net model of a system under live VM migration technique. They evaluated merely AS availability, downtime and downtime cost. Torquato *et al.* [27] analyzed live VM migration based on warm-standby and cold-standby redundancy schemes. They constructed availability models by using Stochastic Petri Nets to evaluate the impact of live VM migration on system availability and power consumption. Note that these studies [13] and [22]–[27] assumed that all time intervals in the models followed exponential distributions. Our work in this paper relaxes this assumption by allowing aging time, failure time, VMM fixing time and live VM migration time follow general distributions, in order to devise a more general model for correctly capturing system behaviors.

There were modeling-based studies [28]–[31] in which some time intervals followed general distributions. Machida *et al.* [28] captured the aging and rejuvenation behaviors of a server virtualized system by using SMP models. Ning *et al.* [29] evaluated AS availability and overall loss probability by using a Markov regenerative process. Based on a SMP, Loganathan *et al.* [30] studied the availability of a manufacturing system. These studies [28]–[30] investigated either AS availability or job completion time. While a few papers presented the analytical models for evaluating both AS availability and job completion time in a virtualized system [31], [32], the models did not used live VM migration technique for rejuvenation. Machida *et al.* [31] presented a SMP to analyze a software execution environment suffering from software aging from the aspect of both service availability and job completion time. There are two major differences between [31] and our work:

- The system modeled in [31] is different from our system. When aging is detected in the host with a running job, the authors in [31] proposed to reduce performance degradation by adding more computing resources to this host. If such solution cannot prevent aging, certain rejuvenation technique is employed. But our system uses live VM migration in order to prevent service performance degradation when aging is detected. In addition, the authors in [31] assumed system crashes only caused by software aging. But we consider other crashing factors, such as certain elusive faults [15]. Namely, we consider the scenario where job crash occurring at any time.
- The model proposed in [31] ignored the system state where aging occurs but this event is not detected. That is, the model proposed in [31] didn't capture the job performance variation and thereby didn't calculate the overall job completion time effectively. We use FIGURE 1 to illustrate this point. We assume that there is no aging in a host running a job in $[t_1, t_2]$. VMM aging occurs from time instant t_2 and live VM migration is triggered at time t_3. Note that aging causes the variation of job performance in unit time (denoted as job unit performance). The model in [31] ignored the system state in $[t_2, t_3]$. Namely, they assumed job unit performance in $[t_2, t_3]$ is same as that in $[t_1, t_2]$. Actually, it is not true. Our model developed in this paper captures the variation in job unit performance. We also consider the decrease in job processing rate due to VMM aging in calculating the overall job completion time to make the results more accurate.

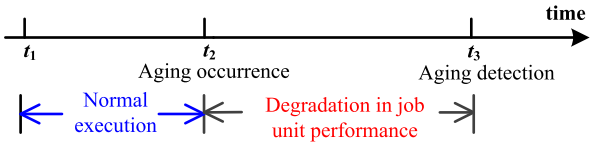

FIGURE 1. Job performance variation over time.

Recently event transition based methodology is developed to evaluate the performance of time-dependent systems. Levitin *et al.* [33] studied both full and partial rejuvenations in a real-time software system by extending event transition based methodology. They focused on evaluating job completion probability. They further [34] considered an operational software system, which has performing real-time tasks and multiple performance degradation levels. Then, they explored an event transition-based numerical method to investigate the optimal state-based rejuvenation policy by minimizing the total expected mission cost in this system. Unlike them, we quantitatively analyze software rejuvenation techniques from AS availability and job completion time.

Besides analytical modeling and event transition based approaches, researchers explored measurement-based approaches. Bovenzi *et al.* [35] evaluated Kexec and Phase-based reboot techniques in terms of downtime overhead reduction, performance penalty and rejuvenation coverage. Huang *et al.* [36] used an adaptive sampling technique for signal reconstruction to detect trends in reconstructed signals and evaluated whether the reconstructed signals can be used to track the gradual change of system performance related to software aging. There are two major differences from our paper. One is that we study both job completion time and AS availability. The other is that we apply VMM reboot and live VM migration techniques to achieve high AS availability and small job completion time. Note that their experiment results can be complementary to our work for better evaluation of AS availability and job completion time.

III. SYSTEM DESCRIPTION AND MODELS

This section first presents the SV-based system architecture considered in this paper, showed in FIGURE 2. Then the SMP model is explained. Finally, the formulas for calculating AS availability and job completion time are derived.

A. SYSTEM DESCRIPTION

The SV-based system mainly consists of one powerful and many weak computing capabilities hosts and Management Host. Jobs can be executed in the hosts. The host with

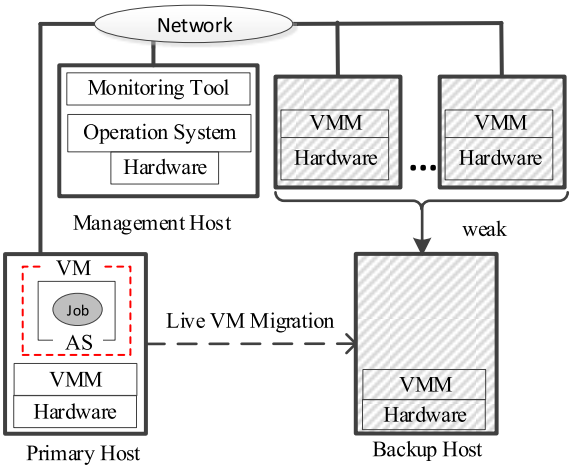

FIGURE 2. SV-based system architecture.

powerful computing capability can be regarded as Primary Host, which includes a VMM hosting an active VM for executing AS. AS is assumed to a constantly running service, through which each user can execute his/her job. One of the hosts with weak computing capability can be used as Backup Host, which includes a VMM and is used to support live VM migration. The monitoring tool deployed in Management Host is responsible for monitoring the behaviors of VMM in each host. These hosts are connected through the network. We consider the execution process of a job in the system.

At the beginning, the job runs in Primary Host. If VMM aging is detected during the job execution, Management Host will immediately examine the state of VMM of the remaining hosts in system and select a host with weak computing capability that does not suffer from software aging or crash as the Backup Host. We assume that there is always an available Backup Host. Note that we leave the relaxation of this assumption in our future work. The selection and examination time of Management Host is negligible. Then live VM migration is triggered and Backup Host will take charge of the job. As long as the job leaves the current host, VMM of this host is rebooted in order to eliminate the possible aging errors. Request and session with established open network connections are not lost during live VM migration (namely, all of the phases of live VM migration) [37]. Live VM migration technique ensures that job can continue its execution from the preempted point. Namely, the job execution follows a preemptive-resume (PRS) discipline [38]. Differently, if VMM reboot technique is used, the job is restarted. Namely, the job execution follows a preemptive-repeat (PRT) discipline [31].

It is reasonable to assume that the VMM reboot time are far less than software aging time. Since we assume Primary Host has powerful computing capacity, job may be completed quickly if it is migrated back to Primary Host when this host is ready. Thus, as soon as the Primary Host is ready, the job is moved back to Primary Host shown in FIGURE 3.

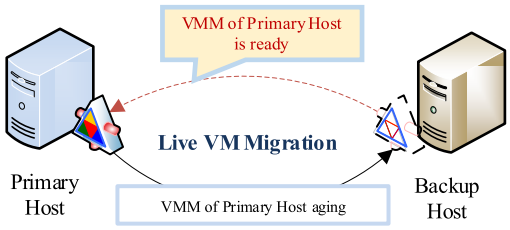

FIGURE 3. The job migration process.

If VMM of Primary Host with the job crashes, the job execution stops according to the dependencies among AS, VM and VMM. After VMM of Primary Host completes its fixing, VMM, VM, AS and job are rebooted/restarted in sequence. Similarly, if VMM of active Backup Host crashes, the job is restarted in Primary Host after VMM of Backup Host completes fixing.

The above description suggests that a system state can be described by a 2-tuple index (i, j). Here, i and j denote the states of Primary Host and Backup Host, respectively. There are five host states: Free, Running, Failed, Migration and Aging, denoted by 0, 1, 2, 3 and 4, respectively. The meaning of each host state is given as follows:

- **Host State** 0 (Free). The job is not running in this host.
- **Host State** 1 (Running). The host is robust and the job is running in it. Both VMM reboot technique and VMM fixing can bring the host back to this state.
- **Host State** 2 (Failed). The host at this state is unavailable, which is caused by VMM crash due to certain elusive faults [15].
- **Host State** 3 (Migration). At this state, the job is ready to move from one host to another via live VM migration.
- **Host State** 4 (Aging). The host at this state can work but its performance is degraded due to VMM aging.

TABLE 1. Meaningful state definition.

No.	System state	Primary Host state	Backup Host state	The system available or not
S_0	(1,0)	Running	Free	Yes
S_1	(4,0)	Aging	Free	Yes
S_2	(3.0)	Migration	Free	No
S_3	(0,1)	Free	Running	Yes
S_4	(0,2)	Free	Failed	No
S_5	(2,0)	Failed	Free	No

There are total $5*5 = 25$ system states, among which there are 17 meaningless system states. These meaningless states can be ignored. Take system state (1,1) and (4,2) for example, a job considered in this paper cannot run in Primary Host and Backup Host simultaneously. Therefore, system state (1,1) is meaningless. When a job is running in one host, the state of the other host is always 0 (Free). The change of system state depends on the state of the host with a running job. Therefore, system state (4,2) is meaningless. TABLE 1 defines eight meaningful system states of the system.

TABLE 2. Definition of variables used in the models.

Symbol	Definition	Distribution	Default Values
T_{A1}	A random variable with distribution function $F_{f1}(t)$ denoting the holding time of Primary Host from Running state to Failed state.	General distribution	21 days [31]
T_{A2}	A random variable with distribution function $F_{f2}(t)$ denoting the holding time of Primary Host VMM suffers from crash after software aging.	General distribution	16 days [31]
T_{A3}	A random variable with distribution function $F_{f3}(t)$ denoting the holding time of Backup Host from Running state to Failed state.	General distribution	19 days
T_{R1}	A random variable with distribution function $G_{r1}(t)$ denoting the holding time of VMM fixing and rebooting in Primary Host.	General distribution	0.1 hours
T_{R2}	A random variable with distribution function $G_{r2}(t)$ denoting the holding time of VMM fixing and rebooting in Backup Host.	General distribution	0.6 hours
T_{Q1}	A random variable with distribution function $F_{q1}(t)$ denoting the holding time of Primary Host from Running state to Aging state.	General distribution	30 days
T_{M}	A random variable with distribution function $M(t)$ denoting the holding time of live VM migration	General distribution	0.5 minutes
T_{U1}	A random variable with distribution function $F_0(t) = u(t - a_1)$ denoting the holding time of Primary Host from Aging state to Migration state.	Unit step function	-
a_1	The migration trigger interval of VM from Primary Host to Backup Host.	-	-

B. SMP MODEL

TABLE 2 defines variables to be used later. FIGURE 4 illustrates the SMP model for capturing behaviors of the SV-based system. Note that the holding time of system from state (4,0) to state (2,0) has the same general distribution $F_{f2}(t)$ as that of the system from state (3,0) to state (2,0). No matter whether it is from state (4,0) to state (2,0) or from state (3,0) to state (3,0), it indicates the holding time that the Primary Host VMM suffers from crash after aging.

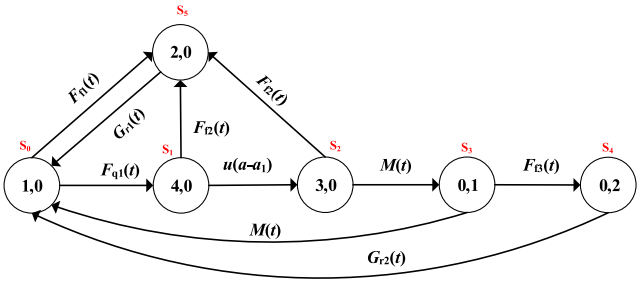

FIGURE 4. SMP model for the SV-based system deploying VMM reboot and live VM migration techniques.

In this model, VMM fixing and VMM reboot will bring the SV-based system to a state without error. In addition, the timer to be used for next VMM fixing or VMM reboot is restarted after completing VMM fixing or VMM reboot. From this point of view, we define $\{Z_s(t) = Z(Y_n, T_n)|Y_n \in Y, T_n \in T\}$ is a stochastic process. The sequence of system states $Y = \{Y_0, Y_1, Y_2, Y_3, Y_4, \ldots, Y_n\}$ $(n \geq 0)$ (including the occurrence of VMM aging, live VM migration, VMM failure, VMM fixing and VMM reboot.) corresponds to Markov renewal moments $T = \{T_0, T_1, T_2, T_3, T_4, \ldots, T_n\}$ $(n \geq 0)$. $Y = \{Y_0, Y_1, Y_2, Y_3, Y_4, \ldots, Y_n\}$ $(n \geq 0)$ is the embedded discrete time Markov chain (DTMC). Thus, the stochastic process $\{Z_s(t)|t \geq 0\}$ is called a SMP [39].

C. FORMULAS FOR CALCULATING AS AVAILABILITY

This section describes the process of calculating AS availability. We use $\mathbf{S_0}$-$\mathbf{S_5}$ to represent the system states. See TABLE 1. The details are as follows.

First of all, we construct the kernel matrix K(t), which can be represented as in Equation (1).

$$\mathbf{K}(t)=\begin{pmatrix} 0 & k_{\mathbf{S_0S_1}}(t) & 0 & 0 & 0 & k_{\mathbf{S_0S_5}}(t) \\ 0 & 0 & k_{\mathbf{S_1S_2}}(t) & 0 & 0 & k_{\mathbf{S_1S_5}}(t) \\ 0 & 0 & 0 & k_{\mathbf{S_2S_3}}(t) & 0 & k_{\mathbf{S_2S_5}}(t) \\ k_{\mathbf{S_3S_0}}(t) & 0 & 0 & 0 & k_{\mathbf{S_3S_4}}(t) & 0 \\ k_{\mathbf{S_4S_0}}(t) & 0 & 0 & 0 & 0 & 0 \\ k_{\mathbf{S_5S_0}}(t) & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{1}$$

The non-null element $k_{01}(t)$ is defined in Equation (2).

$$\begin{aligned} &k_{\mathbf{S_0S_1}}(t) \\ &\quad = Pr\{\text{Aging of primary host occures within time } t\} \\ &\quad = Pr\{Y_1 = S_1, T_1 \leq t | Y_0 = S_0\} \\ &\quad = \int_0^t (1 - F_{f1}(x)) dF_{q1}(x) \end{aligned} \tag{2}$$

The left elements in $\mathbf{K}(t)$ have similar definitions, given in Appendix in the supplemental section of the paper. By solving its one-step transition probability matrix (TPM) $\mathbf{P} = [p_{S_iS_j}]$, we can characterize the sequence of system states. The one-step TPM is $\mathbf{P} = \lim_{t\to\infty} \mathbf{K}(t)$ for the embedded DTMC of the SMP. Then, we can get matrix $\mathbf{P}$.

$$\mathbf{P}=\begin{pmatrix} 0 & p_{\mathbf{S_0S_1}} & 0 & 0 & 0 & p_{\mathbf{S_0S_5}} \\ 0 & 0 & p_{\mathbf{S_1S_2}} & 0 & 0 & p_{\mathbf{S_1S_5}} \\ 0 & 0 & 0 & p_{\mathbf{S_2S_3}} & 0 & p_{\mathbf{S_2S_5}} \\ p_{\mathbf{S_3S_0}} & 0 & 0 & 0 & p_{\mathbf{S_3S_4}} & 0 \\ p_{\mathbf{S_4S_0}} & 0 & 0 & 0 & 0 & 0 \\ p_{\mathbf{S_5S_0}} & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{3}$$

where the equations for calculating the non-null elements of the matrix are given in Appendix in the supplemental section of the paper. In order to obtain steady-state probability vector $\mathbf{V}$ of the embedded DTMC, we can solve the linear system of equations:

$$\mathbf{V} = \mathbf{VP} \text{ subject to } \mathbf{V}e^T = 1 \tag{4}$$

Then, we can get the equation of $v_{\mathbf{S_0}}$ as follows:

$$\begin{aligned} v_{\mathbf{S_0}} = &-1/(p_{\mathbf{S_0}\mathbf{S_1}}p_{\mathbf{S_1}\mathbf{S_2}}p_{\mathbf{S_2}\mathbf{S_3}}p_{\mathbf{S_3}\mathbf{S_0}} \\ &- p_{\mathbf{S_0}\mathbf{S_1}}p_{\mathbf{S_1}\mathbf{S_2}}p_{\mathbf{S_2}\mathbf{S_3}} - p_{\mathbf{S_0}\mathbf{S_1}}p_{\mathbf{S_1}\mathbf{S_2}} - p_{\mathbf{S_0}\mathbf{S_1}} - 2) \end{aligned} \tag{5}$$

The equations for calculating $v_{\mathbf{S_0}}, v_{\mathbf{S_1}}, v_{\mathbf{S_2}}, v_{\mathbf{S_3}}, v_{\mathbf{S_4}}$ and $v_{\mathbf{S_5}}$ are given in Appendix in the supplemental section of the paper. Once $\mathbf{V}$ is calculated, we need to obtain the mean sojourn times $h_{\mathbf{S_i}}$ at system state S_i, which is to be used in Equation (7). The formula can be written as follows:

$$h_{\mathbf{S_i}} = \int_0^\infty (1 - H_{\mathbf{S_i}}(t))dt \tag{6}$$

where $H_{\mathbf{S_i}}(t)$ is the sojourn time distribution at system state $\mathbf{S_i}$. The equations for calculating $h_{\mathbf{S_0}}, h_{\mathbf{S_1}}, h_{\mathbf{S_2}}, h_{\mathbf{S_3}}, h_{\mathbf{S_4}}$ and $h_{\mathbf{S_5}}$ are shown in Appendix in the supplemental section of the paper.

Then the steady-state probability $\pi_{\mathbf{S_i}}$ for the system state $\mathbf{S_i}$ is calculated by using Equation (7) according to [39]:

$$\pi_{\mathbf{S_i}} = \frac{v_{\mathbf{S_i}} h_{\mathbf{S_i}}}{\sum_{\mathbf{S_j}} v_{\mathbf{S_j}} h_{\mathbf{S_j}}} \tag{7}$$

where $v_{\mathbf{S_i}}$ and $h_{\mathbf{S_i}}$ can be obtained by Equation (4) and (6).

In this model, the steady-state availability of the system A_1 is computed by the sum of the steady-state probability of system state $\mathbf{S_0}$ ($\pi_{\mathbf{S_0}}$), system state $\mathbf{S_1}$ ($\pi_{\mathbf{S_1}}$) and system state $\mathbf{S_3}$ (π_{S_3}) and presented as follows:

$$A_1 = \pi_{\mathbf{S_0}} + \pi_{\mathbf{S_1}} + \pi_{\mathbf{S_3}} \tag{8}$$

D. FORMULAS FOR CALCULATING JOB COMPLETION TIME

This section analyzes job completion time is defined to denote the amount of time to complete a job. The work requirement for this job is work units. We assume that a work unit is processed in an hour in the execution environment. If the job encounters a failure at time instant ($h > 0$), it will be restarted. The details of calculating job completion time are given in the following.

According to FIGURE 4, we assume that the job starts its execution from system state $\mathbf{S_0}$. If h is not less than x, job completion time $C(x)$ is equal to x. If h is less than x, job completion time $C(x)$ becomes the sum of h, VMM fixing time and $C(x)$. The details are as follows:

1) The migration trigger interval a_1 is larger than x.

2) When migration trigger interval a_1 is less than x, there are two cases as follows:

2.1). h is less than migration trigger interval a_1. We define that the Primary Host VMM suffers from software aging at time a. There are two situations in which the job fails. One is that Primary Host failure occurs before a and the other is after a.

2.2). h is larger than migration trigger interval a_1.

The mean job completion time is represented as follows [31]:

$$E(C(x)) = -\frac{\partial \Phi_C^{\sim}(s,x)}{\partial s}|_{s=0} \tag{9}$$

where Laplace-Stieltjes transforms (LST) of job completion time $\Phi_C^{\sim}(s,x)$ is derived as follows:

$$\begin{aligned} \Phi_C^{\sim}(s,x) = &\, e^{-sx} F_{\mathrm{q1}}^{\sim}(s)(1 - F_{\mathrm{f2}}(a_1)) \int_x^\infty dF_{\mathrm{f3}}(h - a_1) \\ &+ G_{\mathrm{r1}}^{\sim}(s)\Phi_C^{\sim}(s,x) \int_0^a e^{-sh} dF_{\mathrm{f1}}(h) \\ &+ G_{\mathrm{r1}}^{\sim}(s)(1 - F_{\mathrm{f1}}(a)) \int_a^{a_1} e^{-sh} dF_{\mathrm{f2}}(h - a)\Phi_C^{\sim}(s,x) \\ &+ G_{\mathrm{r2}}^{\sim}(s)(1 - F_{\mathrm{f2}}(a_1))F_{\mathrm{q1}}^{\sim}(s) \\ &\int_{a_1}^x e^{-sh} dF_{\mathrm{f3}}(h - a_1)\Phi_C^{\sim}(s,x) \end{aligned} \tag{10}$$

Solving Equation (9), we can get the mean job completion time. In addition, if the effect of job processing rate r_1 at Aging state of Primary Host and job processing rate r_2 at Running state of Backup Host on job completion time is considered, $\Phi_C^{\sim}(s,x)$ in Equation (10) can be written as Equation (11):

$$\begin{aligned} \Phi_C^{\sim}(s,x) = &\, e^{-s(a + \frac{a_1 - a}{r_1} + \frac{x - a_1}{r_2})} F_{\mathrm{q1}}^{\sim}(s) \\ &(1 - F_{\mathrm{f2}}(a + \frac{a_1 - a}{r_1})) \int_{(a + \frac{a_1 - a}{r_1} + \frac{x - a_1}{r_2})}^\infty \\ &dF_{\mathrm{f3}}(h - (a + \frac{a_1 - a}{r_1})) \\ &+ G_{\mathrm{r1}}^{\sim}(s)\Phi_C^{\sim}(s,x) \int_0^a e^{-sh} dF_{\mathrm{f1}}(h) \\ &+ G_{\mathrm{r1}}^{\sim}(s)(1 - F_{\mathrm{f1}}(a)) \int_a^{a + \frac{a_1 - a}{r_1}} \\ &e^{-sh} dF_{\mathrm{f2}}(h - a)\Phi_C^{\sim}(s,x) \\ &+ G_{\mathrm{r2}}^{\sim}(s)(1 - F_{\mathrm{f2}}(a + \frac{a_1 - a}{r_1}))\Phi_C^{\sim}(s,x) \\ &F_{\mathrm{q1}}^{\sim}(s) \int_{a + \frac{a_1 - a}{r_1}}^{(a + \frac{a_1 - a}{r_1} + \frac{x - a_1}{r_2})} \\ &e^{-sh} dF_{\mathrm{f3}}(h - (a + \frac{a_1 - a}{r_1})) \end{aligned} \tag{11}$$

The $\Phi_C^{\sim}(s,x)$ in Equation (11) is the overall job completion time considering both Running state and Aging state together by setting different job processing rate at these two states.

IV. ANALYTICAL EXPERIMENTS

In this section, we apply our proposed equations to investigate AS availability (using Equation (8)) and the mean job completion time (using Equation (11)) over various

system parameters. Section IV-*A* introduces experiment configuration.Section IV-*B* and Section IV-*C* describe the mean job completion time and AS availability under varying parameters.

A. EXPERIMENT CONFIGURATION

Failure time is assumed to have an Increasing Failure Rate distribution because the failure rate caused by software aging tends to increase with time [40]. Hypo-exponential distribution is a typical Increasing Failure Rate distribution [31]. The time of Primary Host (Backup Host) from Running state to Failed state follows the Hypo-exponential distribution, corresponding distribution function $F_{f1}(t) = HYPO(\lambda_1, \lambda_2)$ ($F_{f3}(t) = HYPO(\lambda_1, \lambda_4)$). The time of Primary Host (Backup Host) from Aging state to Failed state is assumed to follow the Hypo-exponential distribution, corresponding distribution function $F_{f2}(t) = HYPO(\lambda_1, \lambda_3)$ ($F_{f4}(t) = HYPO(\lambda_5, \lambda_6)$). In addition, random variables T_{R1}, T_{R2}, T_{Q1}, T_{Q2} and T_M are assumed to follow the exponential distribution with parameter α,γ,κ,μ and σ, respectively. The use of Hypo-exponential distribution and exponential distribution is just as an example. Other distributions can be used for analytical experiments. Some paraments used for solving AS availability and the mean job completion time are set according to [31]. The left parameters are set in order to demonstrate the effectiveness of our model proposed in this paper. The default settings of parameters are given in 0, where '-' in the 'Distribution' column denotes that variables do not follow any distribution, while '-' in the 'Default Values' column indicates no default settings of parameters. Analytical experiments are conducted on MAPLE [41].

B. EFFECT OF MIGRATION TRIGGER INTERVAL ON JOB COMPLETION TIME

This section describes the relationship between the mean job completion time and migration trigger interval under varying job processing rate and the mean VMM fixing time $1/\alpha$. We assume $a < a_1 < a_2 < x$ in the formula for calculating job completion time. x, a and a_2 are set to be 360 hours [31], 50 hours and 354 hours, respectively. Thus, migration trigger interval varies from 100 hours to 350 hours.

1) JOB PROCESSING RATE r_1 AT AGING STATE OF PRIMARY HOST

First, we investigate the job completion time by varying migration trigger interval a_1 and job processing rate r_1 at Aging state of Primary Host. Job processing rate r_1 at Aging state of Primary Host is set to be 0.6, 0.7 and 0.8 respectively. The left parameters are fixed. FIGURE 5 shows experimental results. We can observe:

- When job processing rate r_1 at Aging state of Primary Host is 0.6, the mean job completion time is approximately minimized to 1370.4227 hours at $a_1 = 160$ hours. It is denoted by (160,1370.4227) in FIGURE 5. Similarly, (204,1305.9980) and (243.7, 1238.9651) denote the approximate minimum job completion time and the corresponding optimal migration trigger intervals at $r = 0.7$ and 0.8, respectively.
- With the increasing job processing rate r_1 at Aging state of Primary Host, the mean job completion time decreases gradually. It can be explained that work units completed increase in unit time when job processing rate r_1 at Aging state of Primary Host increases.
- After the mean job completion time reaches its minimum value, it increases gradually with the increasing migration trigger interval a_1. When migration trigger interval a_1 is small, the frequency of migration increases, which results in the increase in the mean job completion time. When migration trigger interval a_1 is large, the probability of system failure increases, which results in the increase in the mean job completion time. Consequently, the mean job completion time first decreases and then increases with the increasing migration trigger interval a_1.

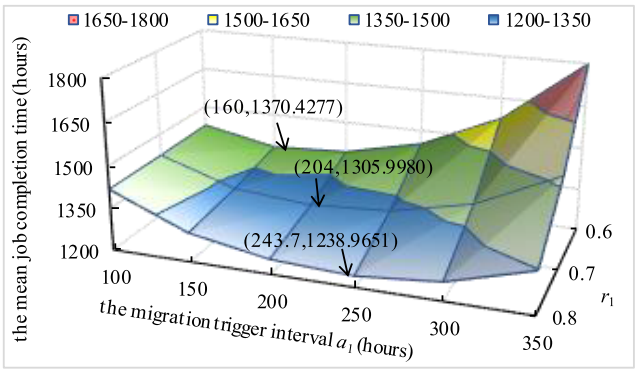

FIGURE 5. The mean job completion time under different job processing rate r_1 at Aging state of Primary Host.

2) JOB PROCESSING RATE r_2 AT RUNNING STATE OF BACKUP HOST

First, we investigate the job completion time by varying migration trigger interval a_1 and job processing rate r_2 at Running state of Backup Host. Job processing rate r_2 at Running state of Backup Host is set to be 0.7, 0.8 and 0.9 respectively. The left parameters are fixed. FIGURE 6 shows experimental results. We can observe:

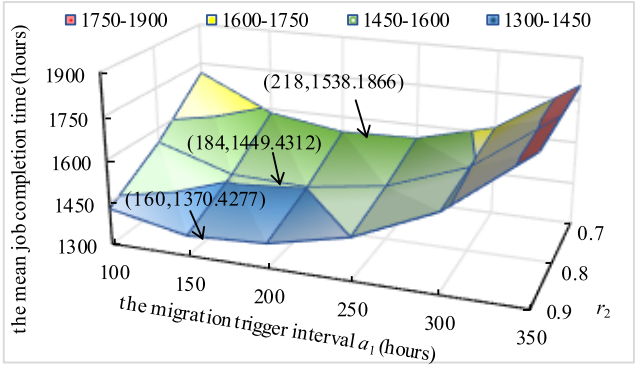

FIGURE 6. The mean job completion time under different job processing rate r_2 at Running state of Backup Host.

TABLE 3. The approximate minimum mean job completion time and the corresponding optimal migration trigger interval.

$1/\alpha$ =0.5	$1/\alpha$ =0.6	$1/\alpha$ =0.7
(160.38, 1370.3993)	(160, 1370.4277)	(160.41, 1370.4532)

- When job processing rate r_2 at Running state of Backup Host is 0.9, the mean job completion time is approximately minimized to 1370.4277 hours at $a_1 = 160$ hours. It is denoted by (160, 1370.4277) in FIGURE 6. Similarly, (184, 1449.4312) and (218, 1538.1866) denote the approximate minimum job completion time and the corresponding optimal migration trigger intervals at $r = 0.8$ and 0.7, respectively.
- With the increasing job processing rate r_2 at Running state of Backup Host, the mean job completion time decreases gradually. It can be explained that work units completed increase in unit time when job processing rate r_2 at Running state of Backup Host increases.
- After the mean job completion time reaches its minimum value, it increases gradually with the increasing migration trigger interval a_1. When migration trigger interval a_1 is small, the frequency of migration increases, which results in the increase in the mean job completion time. The reason is the same as in Section IV-*B (1)*.

3) MEAN VMM FIXING TIME $1/\alpha$

Next, we investigate the job completion time by varying migration trigger interval a_1 and mean VMM fixing time $1/\alpha$. Mean VMM fixing time $1/\alpha$ is set to be 0.5 hours, 0.6 hours and 0.7 hours, respectively, while the left parameters are fixed. FIGURE 7 and IV-C.1 show experimental results. We can observe:

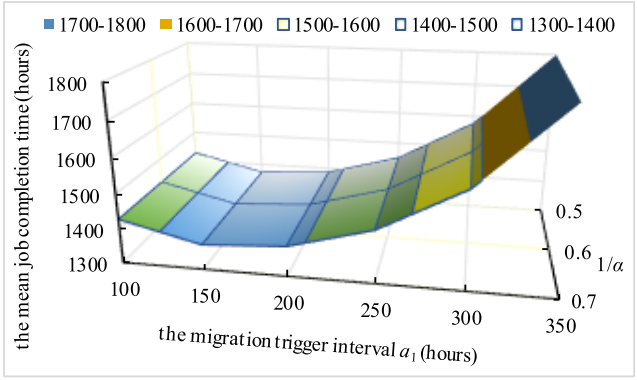

FIGURE 7. The mean job completion time under mean VMM fixing time $1/\alpha$.

- When mean VMM fixing time $1/\alpha$ is 0.5 hours, the mean job completion time is approximately minimized to 1370.3993 hours at $a_1 = 160.38$ hours. It is denoted by (160.38, 1370.3993) in IV-C.1. Similarly, (160, 1370.4277) and (160.41, 1370.4532) denote the approximate minimum job completion time and the corresponding optimal migration trigger intervals at $1/\alpha = 0.6$ hours and 0.7 hours, respectively.
- With the increasing mean VMM fixing time $1/\alpha$, the mean job completion time increases gradually. It can be explained that the increase in job completion time due to the increase in mean VMM fixing time $1/\alpha$.
- After the mean job completion time reaches its minimum value, it increases gradually with the increasing migration trigger interval a_1. The reason is the same as in Section IV-*B (1)*.

C. EFFECT OF MIGRATION TRIGGER INTERVAL ON AS AVAILABILITY

This section describes the relationship between AS availability and the migration trigger interval under different failure rate parameter λ_2 and the mean VMM fixing time $1/\alpha$. Moreover, the relationship between the approximate maximum AS availability and the corresponding optimal migration trigger interval under different live VM migration rate σ is investigated. AS availability is closely related to the sojourn time in each system state. As modeled in Section III, migration occurs after system state $\mathbf{S_1}$. By Equation (A.27) in the Appendix in the supplemental section of the paper, the approximate maximum sojourn time in system state $\mathbf{S_1}$ is computed to 336 hours. The migration trigger interval is varied from 350 hours to 950 hours.

1) FAILURE RATE PARAMETER λ_2

First, we investigate AS availability by varying migration trigger interval a_1 and failure rate parameter λ_2. The failure rate parameter λ_2 is set to be 0.00495, 0.00595 and 0.00695 respectively while the left parameters are fixed. FIGURE 8 illustrates the experimental results. We can observe:

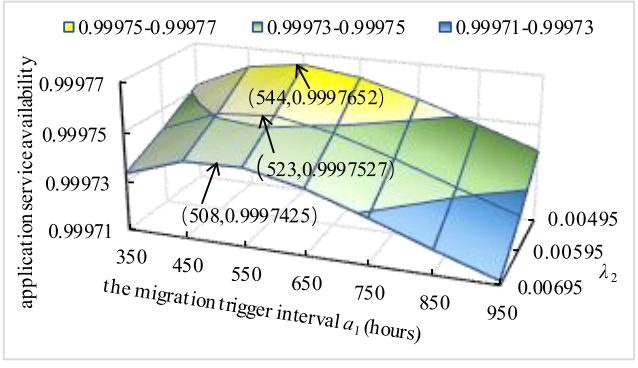

FIGURE 8. AS availability under different failure rate parameter λ_2.

- When failure rate parameter λ_2 is 0.00695, the AS availability is approximately maximized to 0.9997425 at $a_1 = 508$ hours. The maximum point is denoted by (508, 0.9997425) in FIGURE 8. Similarly, (523, 0.9997527) and (544, 0.9997652) denote the approximate maximum AS availabilities and the corresponding optimal migration trigger intervals at $\lambda_2 = 0.00595$ and 0.00495, respectively.
- With the increasing failure rate parameter λ_2, AS availability decreases gradually. It can be explained that

the holding time of system staying at available states decreases when failure rate parameter λ_2 increases.

- AS availability decreases with the increasing migration trigger interval a_1 after it reaches its maximum value. When migration trigger interval a_1 is small, the holding time of system staying at available states increases, which leads to the increase in AS availability. When live VM migration trigger interval a_1 is large, the probability of system failure increases, which leads to the decline in AS availability. Consequently, AS availability increases up to the maximum value and then decreases with the increasing migration trigger interval a_1.

2) MEAN VMM FIXING TIME $1/\alpha$

Next, we investigate AS availability by varying migration trigger interval a_1 and VMM fixing time $1/\alpha$. Mean VMM fixing time $1/\alpha$ is set to be 0.5 hours, 0.6 hours and 0.7 hours respectively, while the left parameters are fixed. FIGURE 9 illustrates the experimental results. We can observe:

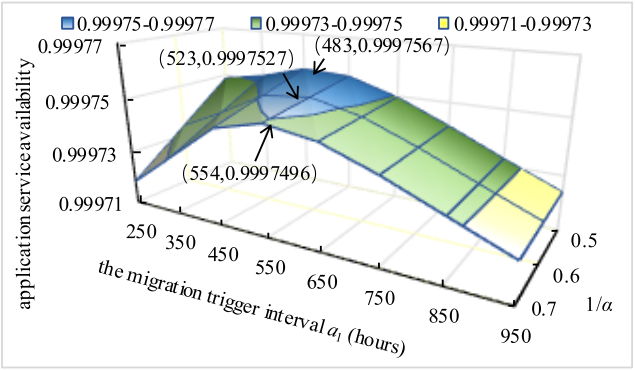

FIGURE 9. AS availability under different mean VMM fixing time $1/\alpha$.

- When mean VMM fixing time $1/\alpha$ is 0.5 hours, the AS availability is approximately maximized to 0.9997567 at $a_1 = 483$ hours. The maximum point is denoted by (483, 0.9997567) in FIGURE 9. Similarly, (523, 0.9997527) and (554, 0.9997496) denote the approximate maximum AS availabilities and the corresponding optimal migration trigger intervals at $1/\alpha = 0.6$ hours and 0.7 hours, respectively.
- With the increasing mean VMM fixing time $1/\alpha$, AS availability decreases gradually. It can be explained that the holding time of system staying at unavailable states increases with the increasing mean VMM fixing time.
- After AS availability reaches its maximum value, it decreases with the increasing migration trigger interval a_1. The reason is that the probability of Primary Host failure before the service migration to Backup Host increases when migration trigger interval a_1 becomes large.

3) LIVE VM MIGRATION RATE σ

Finally, we investigate the relationship between the maximum AS availability and the corresponding optimal migration trigger interval under different live VM migration rate σ. Mean live VM migration rate σ is set to be 48, 72, 96, 120 and 144, while the left parameters are fixed. FIGURE 10 shows the experimental results.

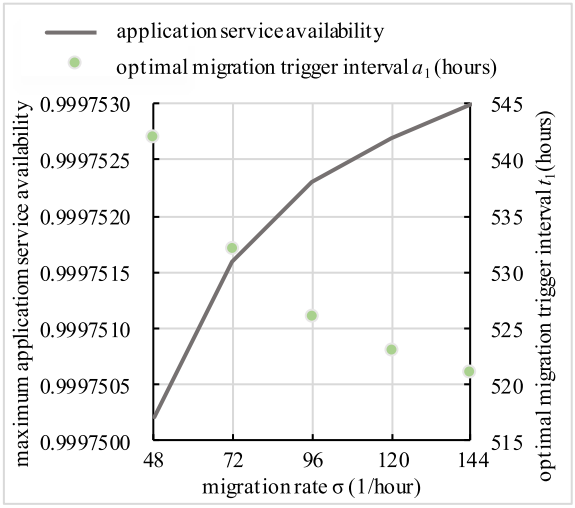

FIGURE 10. The maximum AS availability and the optimal migration trigger interval under different live VM migration rate σ.

We observe that the maximum AS availability increases and the corresponding optimal migration trigger interval decreases with the increasing σ. The reason is that σ determines time of system staying at unavailable states. When live VM migration rate σ is large, the holding time of system staying at unavailable states decreases, which leads to the increasing maximum AS availability.

V. CONCLUSION AND FUTURE WORK

In this paper, we apply the SMP to quantitatively study the AS availability and job completion time in an SV-based system deploying VMM reboot and live VM migration techniques. We derive the equations for calculating AS availability and job completion time under various migration trigger intervals. Finally, we determine the optimal migration trigger intervals for achieving the approximate maximum AS availability and the approximate minimum job completion time through analytical experiments to help service providers make decisions for maximizing the benefits of service providers and users.

Note that this paper considers VMM reboot and live VM migration techniques. Future work includes the investigation of the scenarios where more rejuvenation techniques are adopted for improving AS availability and job completion time. In addition, we want to calculate more evaluation metrics, such as cost and the mean time to failure, etc., in order to evaluate the effectiveness of the software rejuvenation techniques. In addition, we will investigate whether the extended deterministic and stochastic Petri nets can be applied to model the system considered in this paper.

APPENDIX

This section provides formulas for calculating AS availability (Section III-*C* of the main paper). The definition of

parameters involved in the equations is shown in TABLE 2 of the main paper. The detailed solution process is as follows:

First, we obtain the equations for calculating the elements of the kernel matrix $\mathbf{K}(t)$, given in Equation (A.1)-(A.10).

$$\begin{aligned} k_{\mathbf{S_0S_1}}(t) &= Pr\{Y_1 = \mathbf{S_1}, \quad T_1 \le t | Y_0 = \mathbf{S_0}\} \\ &= \int_0^t (1 - F_{\mathrm{f1}}(x)) dF_{\mathrm{q1}}(x) \end{aligned} \tag{A.1}$$

$$\begin{aligned} k_{\mathbf{S_0S_5}}(t) &= Pr\{Y_1 = \mathbf{S_5}, \quad T_1 \le t | Y_0 = \mathbf{S_0}\} \\ &= \int_0^t (1 - F_{\mathrm{q1}}(x)) dF_{\mathrm{f1}}(x) \end{aligned} \tag{A.2}$$

$$\begin{aligned} k_{\mathbf{S_1S_2}}(t) &= Pr\{Y_1 = S_2, \quad T_1 \le t | Y_0 = \mathbf{S_1}\} \\ &= \int_0^t (1 - F_{\mathrm{f2}}(x)) dF_0(x) \end{aligned} \tag{A.3}$$

$$\begin{aligned} k_{\mathbf{S_1S_5}}(t) &= Pr\{Y_1 = \mathbf{S_5}, \quad T_1 \le t | Y_0 = \mathbf{S_1}\} \\ &= \int_0^t (1 - F_0(x)) dF_{\mathrm{f2}}(x) \end{aligned} \tag{A.4}$$

$$\begin{aligned} k_{\mathbf{S_2S_3}}(t) &= Pr\{Y_1 = \mathbf{S_3}, \quad T_1 \le t | Y_0 = \mathbf{S_2}\} \\ &= \int_0^t (1 - F_{\mathrm{f2}}(x)) dM(x) \end{aligned} \tag{A.5}$$

$$\begin{aligned} k_{\mathbf{S_2S_5}}(t) &= Pr\{Y_1 = \mathbf{S_5}, \quad T_1 \le t | Y_0 = \mathbf{S_2}\} \\ &= \int_0^t (1 - M(x)) dF_{\mathrm{f2}}(x) \end{aligned} \tag{A.6}$$

$$\begin{aligned} k_{\mathbf{S_3S_0}}(t) &= Pr\{Y_1 = \mathbf{S_0}, \quad T_1 \le t | Y_0 = \mathbf{S_3}\} \\ &= \int_0^t (1 - F_{\mathrm{f3}}(x)) dM(x) \end{aligned} \tag{A.7}$$

$$\begin{aligned} k_{\mathbf{S_3S_4}}(t) &= Pr\{Y_1 = \mathbf{S_4}, \quad T_1 \le t | Y_0 = \mathbf{S_3}\} \\ &= \int_0^t (1 - M(x)) dF_{\mathrm{f3}}(x) \end{aligned} \tag{A.8}$$

$$k_{\mathbf{S_4S_0}}(t) = Pr\{Y_1 = \mathbf{S_0}, \quad T_1 \le t | Y_0 = \mathbf{S_4}\} = G_{\mathrm{r2}}(t) \tag{A.9}$$

$$k_{\mathbf{S_5S_0}}(t) = Pr\{Y_1 = \mathbf{S_0}, \quad T_1 \le t | Y_0 = \mathbf{S_5}\} = G_{\mathrm{r1}}(t) \tag{A.10}$$

Then, we can get the one-step TPM $\mathbf{P} = [p_{\mathrm{S_iS_j}}]$ by $\mathbf{P} = \lim_{t \to \infty} \mathbf{K}(t)$. The equations for calculating the elements of the one-step TPM $\mathbf{P}$ are derived as in Equation (A.11)-(A.20).

$$p_{\mathbf{S_0S_1}} = 1 - F_{\mathrm{f1}}(t) \tag{A.11}$$

$$p_{\mathbf{S_0S_5}} = F_{\mathrm{f1}}(t) \tag{A.12}$$

$$p_{\mathbf{S_1S_2}} = 1 - F_{\mathrm{f2}}(t) \tag{A.13}$$

$$p_{\mathbf{S_5S_0}} = 1 \tag{A.14}$$

$$p_{\mathbf{S_1S_5}} = F_{\mathrm{f2}}(t) \tag{A.15}$$

$$p_{\mathbf{S_2S_3}} = 1 - F_{\mathrm{f2}}(t) \tag{A.16}$$

$$p_{\mathbf{S_2S_5}} = F_{\mathrm{f2}}(t) \tag{A.17}$$

$$p_{\mathbf{S_3S_0}} = 1 - F_{\mathrm{f3}}(t) \tag{A.18}$$

$$p_{\mathbf{S_3S_4}} = F_{\mathrm{f3}}(t) \tag{A.19}$$

$$p_{\mathbf{S_4S_0}} = 1 \tag{A.20}$$

Next, we calculate the steady-state probability vector $\mathbf{V}$ by solving $\mathbf{V} = \mathbf{VP}$ (subject to $\mathbf{V}e^T = 1$) in embedded DTMC. The equations for calculating the steady-state probability $v_{\mathrm{S_i}}$ of state $\mathbf{S_i}$ in embedded DTMC are derived as in Equation (A.21)-(A.26).

$$\begin{aligned} v_{\mathbf{S_0}} = &-1/(p_{\mathbf{S_0S_1}}p_{\mathbf{S_1S_2}}p_{\mathbf{S_2S_3}}p_{\mathbf{S_3S_0}} \\ &- p_{\mathbf{S_0S_1}}p_{\mathbf{S_1S_2}}p_{\mathbf{S_2S_3}} - p_{\mathbf{S_0S_1}}p_{\mathbf{S_1S_2}} - p_{\mathbf{S_0S_1}} - 2) \end{aligned} \tag{A.21}$$

$$\begin{aligned} v_{\mathbf{S_1}} = &-(p_{\mathbf{S_0S_1}})/(p_{\mathbf{S_0S_1}}p_{\mathbf{S_1S_2}}p_{\mathbf{S_2S_3}}p_{\mathbf{S_3S_0}} \\ &- p_{\mathbf{S_0S_1}}p_{\mathbf{S_1S_2}}p_{\mathbf{S_2S_3}} - p_{\mathbf{S_0S_1}}p_{\mathbf{S_1S_2}} - p_{\mathbf{S_0S_1}} - 2) \end{aligned} \tag{A.22}$$

$$\begin{aligned} v_{\mathbf{S_2}} = &-(p_{\mathbf{S_0S_1}}p_{\mathbf{S_1S_2}})/(p_{\mathbf{S_0S_1}}p_{\mathbf{S_1S_2}}p_{\mathbf{S_2S_3}}p_{\mathbf{S_3S_0}} \\ &- p_{\mathbf{S_0S_1}}p_{\mathbf{S_1S_2}}p_{\mathbf{S_2S_3}} - p_{\mathbf{S_0S_1}}p_{\mathbf{S_1S_2}} - p_{\mathbf{S_0S_1}} - 2) \end{aligned} \tag{A.23}$$

$$\begin{aligned} v_{\mathbf{S_3}} = &-(p_{\mathbf{S_2S_3}}p_{\mathbf{S_0S_1}}p_{\mathbf{S_1S_2}})/(p_{\mathbf{S_0S_1}}p_{\mathbf{S_1S_2}}p_{\mathbf{S_2S_3}}p_{\mathbf{S_3S_0}} \\ &- p_{\mathbf{S_0S_1}}p_{\mathbf{S_1S_2}}p_{\mathbf{S_2S_3}} - p_{\mathbf{S_0S_1}}p_{\mathbf{S_1S_2}} - p_{\mathbf{S_0S_1}} - 2) \end{aligned} \tag{A.24}$$

$$\begin{aligned} v_{\mathbf{S_4}} = &-(p_{\mathbf{S_3S_4}}p_{\mathbf{S_2S_3}}p_{\mathbf{S_0S_1}}p_{\mathbf{S_1S_2}})/(p_{\mathbf{S_0S_1}}p_{\mathbf{S_1S_2}}p_{\mathbf{S_2S_3}}p_{\mathbf{S_3S_0}} \\ &- p_{\mathbf{S_0S_1}}p_{\mathbf{S_1S_2}}p_{\mathbf{S_2S_3}} - p_{\mathbf{S_0S_1}}p_{\mathbf{S_1S_2}} - p_{\mathbf{S_0S_1}} - 2) \end{aligned} \tag{A.25}$$

$$\begin{aligned} v_{\mathbf{S_5}} = &-(p_{\mathbf{S_0S_1}}p_{\mathbf{S_1S_2}}p_{\mathbf{S_2S_3}}p_{\mathbf{S_3S_0}} + p_{\mathbf{S_0S_1}}p_{\mathbf{S_1S_2}}p_{\mathbf{S_2S_3}}p_{\mathbf{S_3S_4}} \\ &- 1)/(p_{\mathbf{S_0S_1}}p_{\mathbf{S_1S_2}}p_{\mathbf{S_2S_3}}p_{\mathbf{S_3S_0}} - p_{\mathbf{S_0S_1}}p_{\mathbf{S_1S_2}}p_{\mathbf{S_2S_3}} \\ &- p_{\mathbf{S_0S_1}}p_{\mathbf{S_1S_2}} - p_{\mathbf{S_0S_1}} - 2) \end{aligned} \tag{A.26}$$

What's more, we get the mean sojourn time $h_{\mathrm{S_i}}$ in system state $\mathrm{S_i}$ as presented in Equation (A.27)-(A.32).

$$h_{\mathbf{S_0}} = E[H_{\mathbf{S_0}}] = \int_0^\infty (1 - F_{\mathrm{f1}}(t))(1 - F_{\mathrm{q1}}(t)) dt \tag{A.27}$$

$$h_{\mathbf{S_1}} = E[H_{\mathbf{S_1}}] = \int_0^\infty (1 - F_{\mathrm{f2}}(t))(1 - F_0(t)) dt \tag{A.28}$$

$$h_{\mathbf{S_2}} = E[H_{\mathbf{S_2}}] = \int_0^\infty (1 - F_{\mathrm{f2}}(t))(1 - M(t)) dt \tag{A.29}$$

$$h_{\mathbf{S_3}} = E[H_{\mathbf{S_3}}] = \int_0^\infty (1 - F_{\mathrm{f3}}(t))(1 - M(t)) dt \tag{A.30}$$

$$h_{\mathbf{S_4}} = E[H_{\mathbf{S_4}}] = \int_0^\infty (1 - G_{\mathrm{r2}}(t)) dt \tag{A.31}$$

$$h_{\mathbf{S_5}} = E[H_{\mathbf{S_5}}] = \int_0^\infty (1 - G_{\mathrm{r1}}(t)) dt \tag{A.32}$$

Finally, we solve AS availability by Equation (7) of the main paper, in which parameters can be obtained by Equation (A.21)-(A.32).

REFERENCES

[1] T. Kamarainen, Y. Shan, M. Siekkinen, and A. Ylä-Jääski, "Virtual machines vs. Containers in cloud gaming systems," in *Proc. Int. Workshop Netw. Syst. Support Games (NetGames)*, Dec. 2015, pp. 1–6.

[2] S. Mireslami, L. Rakai, M. Wang, and B. H. Far, "Dynamic cloud resource allocation considering demand uncertainty," *IEEE Trans. Cloud Comput.*, early access, 2019, doi: 10.1109/TCC.2019.2897304.

[3] M. Nabi, M. Toeroe, and F. Khendek, "Availability in the cloud: State of the art," *J. Netw. Comput. Appl.*, vol. 60, pp. 54–67, Jan. 2016.

[4] T.-W. Kuo, B.-H. Liou, K. C.-J. Lin, and M.-J. Tsai, "Deploying chains of virtual network functions: On the relation between link and server usage," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1562–1576, Aug. 2018.

[5] W. Zhang, Z. Zhang, S. Zeadally, and H.-C. Chao, "Efficient task scheduling with stochastic delay cost in mobile edge computing," *IEEE Commun. Lett.*, vol. 23, no. 1, pp. 4–7, Jan. 2019.
[6] W. Zhang, Z. Zhang, S. Zeadally, H.-C. Chao, and V. C. M. Leung, "MASM: A multiple–algorithm service model for energy–delay optimization in edge artificial intelligence," *IEEE Trans. Ind. Informat.*, vol. 15, no. 7, pp. 4216–4224, Jul. 2019.
[7] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Dynamic service migration in mobile edge computing based on Markov decision process," *IEEE/ACM Trans. Netw.*, vol. 27, no. 3, pp. 1272–1288, Jun. 2019.
[8] R. Morabito, V. Cozzolino, A. Y. Ding, N. Beijar, and J. Ott, "Consolidate IoT edge computing with lightweight virtualization," *IEEE Netw.*, vol. 32, no. 1, pp. 102–111, Jan. 2018.
[9] T. G. Rodrigues, K. Suto, H. Nishiyama, N. Kato, and K. Temma, "Cloudlets activation scheme for scalable mobile edge computing with transmission power control and virtual machine migration," *IEEE Trans. Comput.*, vol. 67, no. 9, pp. 1287–1300, Sep. 2018.
[10] V. X. Tran, "WS-QoSOnto: A QoS ontology for Web services," in *Proc. IEEE Int. Symp. Service-Oriented Syst. Eng.* Dec. 2008, pp. 233–238.
[11] M. Noshy, A. Ibrahim, and H. A. Ali, "Optimization of live virtual machine migration in cloud computing: A survey and future directions," *J. Netw. Comput. Appl.*, vol. 110, pp. 1–10, May 2018.
[12] P. A. Karger and D. R. Safford, "I/O for virtual machine monitors: Security and performance issues," *IEEE Secur. Privacy*, vol. 6, no. 5, pp. 16–23, Sep./Oct. 2008.
[13] X. Chang, T. Wang, R. J. Rodríguez, and Z. Zhang, "Modeling and analysis of high availability techniques in a virtualized system," *Comput. J.*, vol. 61, no. 2, pp. 180–198, Feb. 2018.
[14] L. Li, K. Vaidyanathan, and K. Trivedi, "An approach for estimation of software aging in a Web server," in *Proc. Int. Symp. Empirical Softw. Eng.*, Jun. 2003, pp. 91–102.
[15] Y. Bao, X. Sun, and K. Trivedi, "A workload-based analysis of software aging, and rejuvenation," *IEEE Trans. Rel.*, vol. 54, no. 3, pp. 541–548, Sep. 2005.
[16] J. Kishigami, "The Role of QoE on IPTV Services style," in *Proc. 9th IEEE Int. Symp. Multimedia (ISM)*, Dec. 2007, pp. 11–13.
[17] (May 6, 2019). *What Is AWS IoT Greengrass?* [Online]. Available: https://docs.aws.amazon.com/zh_cn/greengrass/latest/developerguide/gg-troubleshooting.html
[18] (May 20, 2019). *AWS IoT Greengrass Service Level Agreement*. [Online]. Available: https://aws.amazon.com/cn/greengrass/sla/
[19] Y. Huang, C. Kintala, N. Kolettis, and N. D. Fulton, "Software rejuvenation: Analysis, module and applications," in *Proc. 25th Int. Symp. Fault-Tolerant Comput. Dig. Papers*, Jun. 1995, pp. 381–390.
[20] (Jun. 3, 2019). *2018-09-12-Gartner-Forecasts-Worldwide-Public-Cloud-Revenue-to-Grow-17-Percentin-2019*. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2018-09-12-gartner-forecasts-worldwide-public-cloud-revenue-to-grow-17-percent-in-2019
[21] S. A. Herrod, "Systems research and development at VMware," *SIGOPS Oper. Syst. Rev.*, vol. 44, no. 4, pp. 1–2, Dec. 2010.
[22] X. Changa, Z. Zhang, X. Li, and K. S. Trivedi, "Model-based survivability analysis of a virtualized system," in *Proc. IEEE 41st Conf. Local Comput. Netw. (LCN)*, Nov. 2016, pp. 611–614.
[23] H. Okamura and T. Dohi, "A phase expansion approach for transient analysis of software rejuvenation model," in *Proc. IEEE Int. Symp. Softw. Rel. Eng. Workshops (ISSREW)*, Oct. 2016, pp. 98–103.
[24] J. Rahme and H. Xu, "Dependable and reliable cloud-based systems using multiple software spare components," in *Proc. IEEE SmartWorld, Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Cloud Big Data Comput., Internet People Smart City Innov. (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, Aug. 2017, pp. 1–8.
[25] F. Machida and N. Miyoshi, "Analysis of an optimal stopping problem for software rejuvenation in a deteriorating job processing system," *Rel. Eng. Syst. Saf.*, vol. 168, pp. 128–135, Dec. 2017.
[26] T. A. Nguyen, D. Min, and E. Choi, "Stochastic reward net-based modeling approach for availability quantification of data center systems," in *Dependability Engineering*. London, U.K.: IntechOpen, 2018.
[27] M. Torquato, I. M. Umesh, and P. Maciel, "Models for availability and power consumption evaluation of a private cloud with VMM rejuvenation enabled by VM Live Migration," *J. Supercomput.*, vol. 74, no. 9, pp. 4817–4841, Sep. 2018.
[28] F. Machida, V. F. Nicola, and K. S. Trivedi, "Job completion time on a virtualized server subject to software aging and rejuvenation," in *Proc. IEEE 3rd Int. Workshop Softw. Aging Rejuvenation*, Nov./Dec. 2011, pp. 44–49.
[29] G. Ning, J. Zhao, Y. Lou, J. Alonso, R. Matias, K. S. Trivedi, B.-B. Yin, and K.-Y. Cai, "Optimization of two–granularity software rejuvenation policy based on the Markov regenerative process," *IEEE Trans. Rel.*, vol. 65, no. 4, pp. 1630–1646, Dec. 2016.
[30] M. Loganathan, G. Kumar, and O. Gandhi, "Availability evaluation of manufacturing systems using Semi–Markov model," *Int. J. Comput. Integr. Manuf.*, vol. 29, no. 7, pp. 720–735, Jul. 2016.
[31] F. Machida, J. Xiang, K. Tadano, and Y. Maeno, "Lifetime extension of software execution subject to aging," *IEEE Trans. Rel.*, vol. 66, no. 1, pp. 123–134, Mar. 2017.
[32] F. Machida, V. F. Nicola, and K. S. Trivedi, "Job completion time on a virtualized server with software rejuvenation," *J. Emerg. Technol. Comput. Syst.*, vol. 10, no. 1, pp. 1–26, Jan. 2014.
[33] G. Levitin, L. Xing, and H.-Z. Huang, "Optimization of partial software rejuvenation policy," *Rel. Eng. Syst. Saf.*, vol. 188, pp. 289–296, Aug. 2019.
[34] G. Levitin, L. Xing, and Y. Xiang, "Cost minimization of real-time mission for software systems with rejuvenation," *Rel. Eng. Syst. Saf.*, vol. 193, Jan. 2020, Art. no. 106593.
[35] A. Bovenzi, J. Alonso, H. Yamada, S. Russo, and K. S. Trivedi, "Towards fast OS rejuvenation: An experimental evaluation of fast OS reboot techniques," in *Proc. IEEE 24th Int. Symp. Softw. Rel. Eng. (ISSRE)*, Nov. 2013, pp. 61–70.
[36] T. Huang, N. Kandasamy, H. Sethu, and M. C. Stamm, "An efficient strategy for online performance monitoring of datacenters via adaptive sampling," *IEEE Trans. Cloud Comput.*, vol. 7, no. 1, pp. 155–169, Jan. 2019.
[37] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proc. NSDI*, 2005, pp. 273–286.
[38] V. G. Kulkarni, V. F. Nicola, and K. S. Trivedi, "The completion time of a job on multimode systems," *Adv. Appl. Probab.*, vol. 19, no. 4, pp. 932–954, Dec. 1987.
[39] D. Chen and K. Trivedi, "Analysis of periodic preventive maintenance with general system failure distribution," in *Proc. Pacific Rim Int. Symp. Dependable Comput.*, Nov. 2002, pp. 103–110.
[40] D. HoPark, "Testing whether failure rate changes its trend," *IEEE Trans. Rel.*, vol. 37, no. 4, pp. 375–378, Oct. 1988.
[41] (Jun. 15, 2019). *Maplesoft*. [Online]. Available: http://www.maplesoft.com/products/maple

JING BAI is currently pursuing the Ph.D. degree with the Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University. Her interest includes software reliability and availability.

XIAOLIN CHANG is currently a Professor with the School of Computer and Information Technology, Beijing Jiaotong University. Her current research interests include edge/cloud computing, network security, and security and privacy in machine learning.

FUMIO MACHIDA was a Principal Researcher at NEC Corporation. He was a Visiting Scholar with the Department of Electrical and Computer Engineering, Duke University, in 2010. He is currently an Associate Professor with the Computer Science Department, University of Tsukuba. His research interests include modeling and analysis of system dependability, software aging and rejuvenation, and reliability of machine learning systems. He is a member of ACM. He was a recipient of the Young Scientists' Prize of Japan, in 2014.

KISHOR S. TRIVEDI currently holds the Hudson Chair with the Department of Electrical and Computer Engineering, Duke University. He has authored *Probability and Statistics with Reliability, Queuing and Computer Science Applications*, (John Wiley). He has published more than 500 articles and has supervised more than 45 Ph.D. dissertations. He received the IEEE Computer Society Technical Achievement Award for his research on Software Aging and Rejuvenation.

ZHEN HAN is currently a Professor with the School of Computer and Information Technology, Beijing Jiaotong University. His main research interests are trusted computing, cryptographic protocols, privacy preserving, and network security.