PAPER

Optimal Rejuvenation Policies for Non-Markovian Availability Models with Aperiodic Checkpointing

Junjun ZHENG[†a)], **Hiroyuki OKAMURA**[††b)], ***and*** **Tadashi DOHI**[††c)], ***Members***

SUMMARY In this paper, we present non-Markovian availability models for capturing the dynamics of system behavior of an operational software system that undergoes aperiodic time-based software rejuvenation and checkpointing. Two availability models with rejuvenation are considered taking account of the procedure after the completion of rollback recovery operation. We further proceed to investigate whether there exists the optimal rejuvenation schedule that maximizes the steady-state system availability, which is derived by means of the phase expansion technique, since the resulting models are not the trivial stochastic models such as semi-Markov process and Markov regenerative process, so that it is hard to solve them by using the common approaches like Laplace-Stieltjes transform and embedded Markov chain techniques. The numerical experiments are conducted to determine the optimal rejuvenation trigger timing maximizing the steady-state system availability for each availability model, and to compare both two models.

key words: *software rejuvenation, checkpointing, optimal rejuvenation policy, non-Markovian process, phase expansion, steady-state availability, point-wise availability*

1. Introduction

The performance of software systems degrades with time due to software aging phenomenon, which is caused by aging-related bugs [1] such as the accumulation of errors during the execution of software, and eventually results in the crash/hang failures of the system. Huang et al. [2] reported the aging phenomenon in real telecommunication billing applications where the application experiences a crash or a hang failure over time. Software aging phenomenon does exist and is inevitable, but nevertheless can be controlled or even reversed [1], [3]. To cope with the software aging, software rejuvenation [3], one of proactive fault management techniques, has been a promising approach to improve the system dependability by refreshing the system internal states. Alonso et al. [4] argued that software rejuvenation can address aging issues well, but typically involves an overhead, and that the overhead impact of the rejuvenation techniques is related to their granularity, e.g., the OS-level and the application-level rejuvenation schemes discussed in [5]. Thus, determining an optimal rejuvenation schedule is crucial for improving the system dependability and performance efficiently [5]–[7]. For example, Zheng et al. [5] addressed the optimization problems facing on both time-based and workload-based software rejuvenation strategies for transaction-based systems with Markovian arrivals through quasi birth-and-death process and Markov regenerative process (MRGP) analysis. Dohi et al. [7] considered two basic software rejuvenation models described by MRGPs, and tried to solve the transient solutions by means of Laplace-Stieltjes transform (LST) and its numerical inversion. In their work, the optimal software rejuvenation policy that maximizes the interval reliability was numerically determined.

On the other hand, as another important technique in the software fault-tolerance, checkpointing, the procedure for saving the current data in the main memory in the secondary storage, provides an efficient method for saving re-execution time in the presence of faults [8], and has been widely-adopted to prevent increasing security threats in data protection of software systems such as database systems [9]–[11]. Fukumoto et al. [9] and Dohi et al. [11] introduced different checkpointing schemes, and Ranganathan and Upadhyaya [10] offered a macroscopic view of the temporal behavior related to the database system states. To date, several authors discussed the optimal schedules on both software rejuvenation and checkpointing together by maximizing the system dependability or performance [12]–[14]. For example, in [12], a comprehensive evaluation of aperiodic checkpointing and rejuvenation schemes in an operational software system was carried out, based on two kinds of maintenance policies and dynamic programming approach. Zheng et al. [14] recently presented two non-Markovian state transition diagrams, from the original stochastic reward Petri nets [15], to describe the dynamics of an operational software system that undergoes aperiodic checkpointing, with and without software rejuvenation, respectively, aiming to identify and measure on how a rejuvenation technique can improve the steady-state system availability of the targeted system. A numerical comparison between two models shown that the rejuvenation technique did help improve the system availability, but did not always bring the positive improvement effect, and further revealed the upper and lower bounds of the mean rejuvenation trigger interval for improving the system availability.

In this paper, we focus on the similar to, but

Manuscript received December 10, 2019.
Manuscript revised May 26, 2020.
Manuscript publicized July 16, 2020.

[†]The author is with the Department of Information Science and Engineering, Ritsumeikan University, Kusatsu-shi, 525–8577 Japan.

[††]The authors are with the Department of Information Engineering, Graduate School of Engineering, Hiroshima University, Higashihiroshima-shi, 739–8527 Japan.

a) E-mail: jzheng@asl.cs.ritsumei.ac.jp
b) E-mail: okamu@hiroshima-u.ac.jp
c) E-mail: dohi@hiroshima-u.ac.jp

DOI: 10.1587/transinf.2019EDP7321

somewhat different software systems from [14] where both aperiodic checkpointing and software rejuvenation are executed, and determine the optimal software rejuvenation schedule that maximizes the steady-state availability of the system. Thanks to the stochastic nature, the system behavior is modeled by a non-Markovian state transition diagram, which is not one of the trivial stochastic models such as semi-Markov process (SMP) and MRGP. Therefore, the common approaches, e.g., the LST and the embedded Markov chain techniques, are unavailable for solving the non-Markovian models in this paper. Instead, the phase expansion approach is proposed to address the above issue. The phase expansion [16] is an approximation technique by using phase-type (PH) distribution, which is defined by the probability distribution of the absorbing time in a continuous-time Markov chain (CTMC) with absorbing states. Since the PH distribution is dense for any probability distribution defined on positive domain, the PH distribution can approximate any probability distribution with high precision. The phase expansion has been proven effective in both transient and stationary analysis of non-Markovian models [17], [18]. Most significantly, the main challenge in this work is not based on the trivial CTMC analysis, but is modeling of the system behavior with multiple competitive events and derivation of the underlying approximate CTMC using the phase expansion. In addition, the previous work [14] has proven the advantages of using the rejuvenation technique, via two models without and with rejuvenation, so in this paper, two availability models with rejuvenation concentrated are considered, together with taking account of the effects of different checkpointing schemes. And apart from the stationary perspective [14], the transient analysis is also introduced. For brevity, the main contributions of this paper are summarized as follows:

- Availability modeling of an operational software system with aperiodic time-based software rejuvenation and two different checkpointing schemes via non-Markovian state transition diagrams;
- Both stationary and transient solutions of non-Markovian availability models by using the phase expansion;
- Comparison of two models and evaluation of their optimal software rejuvenation timings, clarifying the effects of both the rejuvenation and two different checkpointing schemes.

The rest of this paper is organized as follows. In Sect. 2, two non-Markovian state transition diagrams for an operational software system with two different checkpointing schemes are described. The underlying approximate (alternatively, PH-expanded) CTMCs are further derived from the non-Markovian models using the phase expansion in Sect. 3. In particular, we formulate both stationary and transient solutions, i.e., the steady-state availability and the point-wise availability measures of the system, aiming at determining the optimal software rejuvenation schedules that maximize the steady-state system availability from the long-run operational perspective, and characterizing the transient behavior of the system. Section 4 is devoted to the numerical illustration of our models. Finally, in Sect. 5, we conclude this paper with some remarks.

2. Model Description

2.1 Assumptions

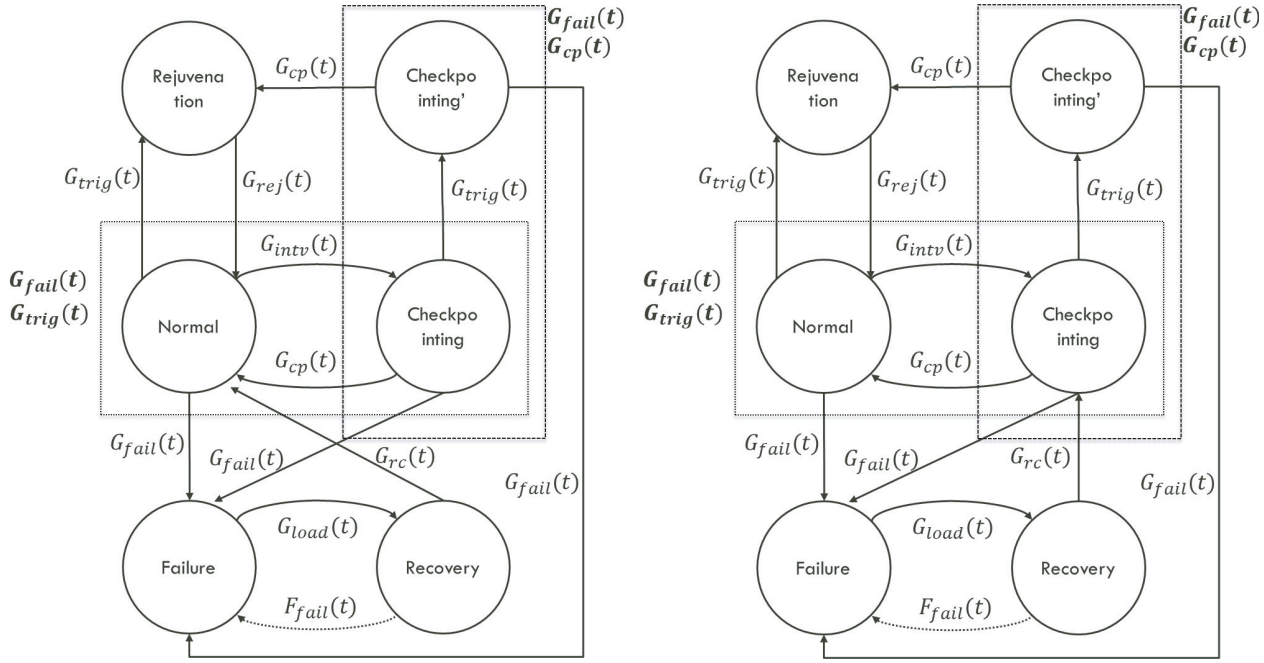
Consider an operational software system, which suffers from software aging and undergoes both rejuvenation and checkpointing by generating rejuvenation points and checkpoints aperiodically. When a system failure occurs, the system undergoes a series of recovery operations including loading of checkpointed data and rollback recovery. We assume that human error-related failures do not occur during checkpointing and that the time to generate aging-related failure follows an absolutely continuous and non-decreasing failure probability (cumulative distribution function (c.d.f.)) $G_{fail}(t)$, while the time distribution for failures generated during rollback recovery due to incorrect recovery operations is given by $F_{fail}(t)$. In general, the c.d.f. $G_{fail}(t)$ for the aging-related failure time is supposed to have an IFR (increasing failure rate) property. The checkpoint interval is assumed to follow c.d.f. $G_{intv}(t)$, and $G_{cp}(t)$ is the c.d.f. of the time needed for checkpointing. $G_{load}(t)$ and $G_{rc}(t)$ are the c.d.f.'s for the loading time of checkpointed data and the time needed until rollback recovery ends, respectively. Since the rejuvenation points are generated aperiodically, the distribution function of the time required to trigger a rejuvenation is assumed to be $G_{trig}(t)$, and the c.d.f. of the rejuvenation overhead (i.e., the time needed for rejuvenation completion) is represented by $G_{rej}(t)$.

If either the checkpoint point or the rejuvenation point is reached before the system failure, the corresponding fault-tolerant operation (checkpointing or rejuvenation) is executed immediately. It should be noted that: (i) the checkpointing operation does not refresh the system aging, in other words, the lifetime is not reset after the execution of checkpointing, while the system is supposed to be as good as new after the rejuvenation; (ii) the clock of rejuvenation trigger is not reset when the system executes the checkpointing; and (iii) in the case where the rejuvenation point is reached when the checkpointing is under execution, the system will wait for rejuvenation and then will rejuvenate itself immediately once the execution of checkpointing is completed.

In this paper, we consider two availability models, including the one with rejuvenation in [14], from the macroscopic view based on the procedure just after the completion of recovery process.

- **Model-I**: The system state moves to the execution process immediately after a rollback recovery [14].
- **Model-II**: The system executes checkpointing immediately after a rollback recovery.

The reason to take place checkpointing after the rollback recovery is that the starting point of the recovery operation is



(a) Model-I. (b) Model-II.

Fig. 1 State transition diagrams.

Table 1 Notation in state transition diagrams.

State	Description
Normal	Normal execution process in the main memory
Checkpointing	Checkpointing execution with a disabled rejuvenation
Checkpointing'	Checkpointing execution with an enabled rejuvenation
Failure	System failure (such as system hang and crash) occurrence
Recovery	Rollback recovery to recover from a system failure
Rejuvenation	Software rejuvenation execution to refresh system internal states

updated from the past to the current time by saving the recovered data in the secondary storage. In addition, it should be noticed that the checkpointing operation does not refresh the system aging and reset the clock of rejuvenation trigger.

2.2 Non-Markovian State Transition Diagrams

The availability models corresponding to Model-I and Model-II are described by non-Markovian state transition diagrams, which are neither the SMPs nor the MRGPs, resulting in difficult numerical analysis.

Figure 1 depicts the state transition diagrams for both Model-I and Model-II. Each model has six states defined in Table 1 and the c.d.f.'s of the corresponding transitions are given in Table 2. In Table 2, GEN means the general distribution, while EXP refers to the exponential distribution. The reasons of assumptions of the probability distributions are: (i) the exponential distribution is characterized by a constant failure rate (CFR) and is often used for the occurrence of relatively "rare events" [19], such as the failure during rollback recovery; (ii) the aging-related system failure has an IFR, so that it can be modeled by a Weibull distribution, which is a general family of the exponential distribution and is widely adopted to model the failure events with an IFR [20]; and (iii) the times spent on other events in Fig. 1 can be modeled by a log-normal distribution, which is frequently used to mode the times spent on the recovery operations to repair a maintainable system, the software rejuvenation against aging phenomena, and the checkpointing schemes for saving the current data [21]–[24]. Without loss of generality, state *Normal* is considered as the initial state, indicating that the system is in the normal execution process in the main memory and waits for the checkpointing, as well as the rejuvenation. At the same

Table 2 The c.d.f.'s of transitions in state transition diagrams.

C.d.f.	Description	Type
$G_{intv}(t)$	c.d.f. of the checkpoint interval.	GEN
$G_{fail}(t)$	c.d.f. of the time to generate aging-related failure.	GEN
$G_{cp}(t)$	c.d.f. of the time needed for checkpointing.	GEN
$G_{load}(t)$	c.d.f. of the loading time of checkpointed data.	GEN
$G_{rc}(t)$	c.d.f. of the time needed for rollback recovery.	GEN
$G_{trig}(t)$	c.d.f. of the time required to trigger a rejuvenation.	GEN
$G_{rej}(t)$	c.d.f. of the rejuvenation overhead.	GEN
$F_{fail}(t)$	c.d.f. of the time to generate a failure during rollback recovery.	EXP

time, the system aging accumulates and may lead to the system failure. Once the aging-related failure occurs, the system enters to state *Failure*, in which a series of recovery operations are forced to undergo, that is, the system moves to state *Recovery* after the loading of checkpointed data, and becomes normal again after the completion of the rollback recovery. The system can go to state *Rejuvenation* or state *Checkpointing* before the occurrence of system failure, according to the c.d.f.'s of transitions to rejuvenation points and checkpoints. In particular, state *Checkpointing'* represents the checkpointing execution state of the system with an enabled rejuvenation (i.e., a rejuvenation point has been reached).

The main difference between Model-I and Model-II is the procedure just after the completion of recovery process. In Model-I, the system goes back to the normal state from state *Recovery*, whereas the system executes checkpointing immediately after the rollback recovery and enters to state *Checkpointing* in Model-II.

3. System Availability Analysis

In this section, we first introduce the definition of the PH distribution [25], and then derive the underlying approximate CTMCs for the non-Markovian state transition diagrams in Fig. 1 with the phase approach by replacing all general distributions with the corresponding PH distributions. Finally, both stationary and transient solutions for Model-I and Model-II through CTMC analysis are presented. The measures of interest are the steady-state availability and the point-wise availability of the system.

3.1 PH Distribution

The PH distribution is defined as the time to absorption in a finite Markov chain with one absorbing state. Strictly speaking, the PH distribution is classified into continuous and discrete PH distributions. This paper deals with continuous PH distribution. Without loss of generality, the infinitesimal generator $\boldsymbol{Q}$ of CTMC is assumed to be partitioned as follows:

$$\boldsymbol{Q} = \left(\begin{array}{c|c} \boldsymbol{T} & \boldsymbol{\xi} \\ \hline \boldsymbol{0} & 0 \end{array}\right), \tag{1}$$

where $\boldsymbol{T}$ and $\boldsymbol{\xi}$ correspond to transition rates between transient states and the exit rates from transient states to the absorbing state, respectively. Let $\boldsymbol{\alpha}$ be an initial probability vector over the transient states. The c.d.f. and p.d.f. (probability density function) of PH distribution are given by

$$F_{PH}(t) = 1 - \boldsymbol{\alpha}\exp(\boldsymbol{T}t)\mathbf{1}, \quad f_{PH}(t) = \boldsymbol{\alpha}\exp^{\boldsymbol{T}t}\boldsymbol{\xi}, \tag{2}$$

in the above equation, $\mathbf{1}$ is a column vector containing all 1 elements. Note that $\boldsymbol{\xi} = -\boldsymbol{T}\mathbf{1}$, the transient states are called *phases*.

The PH distribution has several sub-classes according to the structure of $\boldsymbol{T}$ (see e.g., [26]). In particular, the acyclic PH distribution (APH) is the widest class among mathematically tractable PH distributions. Cumani [27] derived the canonical forms (CFs) as the minimal representation of APH, which has the smallest number of free parameters. The CF1 (canonical form 1) is defined by

$$\boldsymbol{\alpha} = \left(\begin{array}{cccc} \alpha_1 & \alpha_2 & \cdots & \alpha_m \end{array}\right), \tag{3}$$

$$\boldsymbol{T} = \left(\begin{array}{cccccc} -\beta_1 & \beta_1 & & & & \boldsymbol{O} \\ & -\beta_2 & \beta_2 & & & \\ & & \ddots & \ddots & & \\ & & & & -\beta_{m-1} & \beta_{m-1} \\ \boldsymbol{O} & & & & & -\beta_m \end{array}\right), \quad \boldsymbol{\xi} = \left(\begin{array}{c} 0 \\ 0 \\ \vdots \\ 0 \\ \beta_m \end{array}\right), \tag{4}$$

where $\alpha_i \geq 0$, $\sum_i \alpha_i = 1$ and $0 < \beta_1 \leq \cdots \leq \beta_m$ for m phases. The PH distribution can approximate any kind of probability distribution defined on non-negative domain, and the accuracy of approximation depends on the number of phases and the phase structure. The purpose of approximation is to determine the PH parameters $(\boldsymbol{\alpha}, \boldsymbol{T}, \boldsymbol{\xi})$ to approximate the original distribution with the fitted PH distribution. In general, three types of methods have been proposed; moment matching, maximum likelihood estimation (MLE), and Bayes estimation [25].

3.2 PH-Expanded CTMCs

The general distributions in Table 2 are approximated by PH distributions with appropriate phases:

$$\begin{aligned} F_x^{PH}(t) &= 1 - \boldsymbol{\alpha}_x \exp(\boldsymbol{T}_x t)\mathbf{1}_x, \\ f_x^{PH}(t) &= \boldsymbol{\alpha}_x \exp(\boldsymbol{T}_x t)\boldsymbol{\xi}_x \end{aligned} \tag{5}$$

for $x \in \{intv, fail, cp, load, rc, trig, rej\}$. The PH parameters $(\boldsymbol{\alpha}_x, \boldsymbol{T}_x, \boldsymbol{\xi}_x)$ for each approximate distribution are estimated by using a PH fitting algorithm which is based on MLE with EM (expectation-maximization) principle [25], [28].

Using the above PH distributions, the PH-expanded CTMCs for the state transition diagrams in Fig. 1 are derived and formulated in Eqs. (6) and (7) based on the Kronecker representation [16]. More precisely, Eq. (6) indicates the infinitesimal generator matrix of the approximate CTMC for Model-I and Eq. (7) represents the infinitesimal generator matrix of the approximate CTMC for Model-II with the order {*Normal, Checkpointing, Checkpointing', Rejuvenation, Failure, Recovery*}. In these equations, $\oplus$ and $\otimes$ are the Kronecker product and sum [29], $\boldsymbol{I}$ is an identity matrix, and $1/\lambda_{fail}$ is the mean value of c.d.f. $F_{fail}(t)$, namely, the mean time to failure during rollback recovery operation.

Since the lifetime is not reset when the system enters to the state *Checkpointing*, thus the entry of either $\boldsymbol{Q}_1$ or $\boldsymbol{Q}_2$, $(\boldsymbol{\xi}_{intv}\boldsymbol{\alpha}_{cp}) \otimes \boldsymbol{I} \otimes \boldsymbol{I}$, indicates that only the initial states of PH-expanded CTMC related to $G_{cp}(t)$ can be entered. On the other hand, because the checkpointing operation does not refresh the system aging, the initial states of PH-expanded CTMC related to $G_{fail}(t)$ can not be reached after the transition from state *Checkpointing* to state *Normal*, which is

$$\boldsymbol{Q}_1 = \begin{pmatrix} \boldsymbol{T}_{intv} \oplus \boldsymbol{T}_{fail} \oplus \boldsymbol{T}_{trig} & (\boldsymbol{\xi}_{intv}\boldsymbol{\alpha}_{cp}) \otimes \boldsymbol{I} \otimes \boldsymbol{I} & & (\mathbf{1}_{intv} \otimes \mathbf{1}_{fail} \otimes \boldsymbol{\xi}_{trig})\boldsymbol{\alpha}_{rej} & (\mathbf{1}_{intv} \otimes \boldsymbol{\xi}_{fail} \otimes \mathbf{1}_{trig})\boldsymbol{\alpha}_{load} \\ (\boldsymbol{\xi}_{cp}\boldsymbol{\alpha}_{intv}) \otimes \boldsymbol{I} \otimes \boldsymbol{I} & \boldsymbol{T}_{cp} \oplus \boldsymbol{T}_{fail} \oplus \boldsymbol{T}_{trig} & \boldsymbol{I} \otimes \boldsymbol{I} \otimes \boldsymbol{\xi}_{trig} & & (\mathbf{1}_{cp} \otimes \mathbf{1}_{trig} \otimes \boldsymbol{\xi}_{fail})\boldsymbol{\alpha}_{load} \\ & & \boldsymbol{T}_{fail} \oplus \boldsymbol{T}_{cp} & (\mathbf{1}_{fail} \otimes \boldsymbol{\xi}_{cp})\boldsymbol{\alpha}_{rej} & (\boldsymbol{\xi}_{fail} \otimes \mathbf{1}_{cp})\boldsymbol{\alpha}_{load} \\ \boldsymbol{\xi}_{rej}(\boldsymbol{\alpha}_{intv} \otimes \boldsymbol{\alpha}_{fail} \otimes \boldsymbol{\alpha}_{trig}) & & & \boldsymbol{T}_{rej} & \\ & & & & \boldsymbol{T}_{load} \quad \boldsymbol{\xi}_{load}\boldsymbol{\alpha}_{rc} \\ \boldsymbol{\xi}_{rc}(\boldsymbol{\alpha}_{intv} \otimes \boldsymbol{\alpha}_{fail} \otimes \boldsymbol{\alpha}_{trig}) & & & & (\lambda_{fail} \otimes \mathbf{1}_{rc})\boldsymbol{\alpha}_{load} \quad (-\lambda_{fail}) \oplus \boldsymbol{T}_{rc} \end{pmatrix}, \tag{6}$$

$$\boldsymbol{Q}_2 = \begin{pmatrix} \boldsymbol{T}_{intv} \oplus \boldsymbol{T}_{fail} \oplus \boldsymbol{T}_{trig} & (\boldsymbol{\xi}_{intv}\boldsymbol{\alpha}_{cp}) \otimes \boldsymbol{I} \otimes \boldsymbol{I} & & (\mathbf{1}_{intv} \otimes \mathbf{1}_{fail} \otimes \boldsymbol{\xi}_{trig})\boldsymbol{\alpha}_{rej} & (\mathbf{1}_{intv} \otimes \boldsymbol{\xi}_{fail} \otimes \mathbf{1}_{trig})\boldsymbol{\alpha}_{load} \\ (\boldsymbol{\xi}_{cp}\boldsymbol{\alpha}_{intv}) \otimes \boldsymbol{I} \otimes \boldsymbol{I} & \boldsymbol{T}_{cp} \oplus \boldsymbol{T}_{fail} \oplus \boldsymbol{T}_{trig} & \boldsymbol{I} \otimes \boldsymbol{I} \otimes \boldsymbol{\xi}_{trig} & & (\mathbf{1}_{cp} \otimes \mathbf{1}_{trig} \otimes \boldsymbol{\xi}_{fail})\boldsymbol{\alpha}_{load} \\ & & \boldsymbol{T}_{fail} \oplus \boldsymbol{T}_{cp} & (\mathbf{1}_{fail} \otimes \boldsymbol{\xi}_{cp})\boldsymbol{\alpha}_{rej} & (\boldsymbol{\xi}_{fail} \otimes \mathbf{1}_{cp})\boldsymbol{\alpha}_{load} \\ \boldsymbol{\xi}_{rej}(\boldsymbol{\alpha}_{intv} \otimes \boldsymbol{\alpha}_{fail} \otimes \boldsymbol{\alpha}_{trig}) & & & \boldsymbol{T}_{rej} & \\ & & & & \boldsymbol{T}_{load} \quad \boldsymbol{\xi}_{load}\boldsymbol{\alpha}_{rc} \\ & \boldsymbol{\xi}_{rc}(\boldsymbol{\alpha}_{cp} \otimes \boldsymbol{\alpha}_{fail} \otimes \boldsymbol{\alpha}_{trig}) & & & (\lambda_{fail} \otimes \mathbf{1}_{rc})\boldsymbol{\alpha}_{load} \quad (-\lambda_{fail}) \oplus \boldsymbol{T}_{rc} \end{pmatrix}. \tag{7}$$

represented by $(\boldsymbol{\xi}_{cp}\boldsymbol{\alpha}_{intv}) \otimes \boldsymbol{I} \otimes \boldsymbol{I}$. Moreover, the entries, $(\mathbf{1}_{fail} \otimes \boldsymbol{\xi}_{cp})\boldsymbol{\alpha}_{rej}$ and $(\boldsymbol{\xi}_{fail} \otimes \mathbf{1}_{cp})\boldsymbol{\alpha}_{load}$, represent the transitions from the state *Checkpointing'* to the state *Rejuvenation* and to the state *Failure*, respectively. The resets of the checkpoint interval, the clocks of lifetime and rejuvenation trigger due to the software rejuvenation and recovery in Model-I, are represented by the entries, $\boldsymbol{\xi}_{rej}(\boldsymbol{\alpha}_{intv} \otimes \boldsymbol{\alpha}_{fail} \otimes \boldsymbol{\alpha}_{trig})$ and $\boldsymbol{\xi}_{rc}(\boldsymbol{\alpha}_{intv} \otimes \boldsymbol{\alpha}_{fail} \otimes \boldsymbol{\alpha}_{trig})$, respectively. While the entry, $\boldsymbol{\xi}_{rc}(\boldsymbol{\alpha}_{cp} \otimes \boldsymbol{\alpha}_{fail} \otimes \boldsymbol{\alpha}_{trig})$, in $\boldsymbol{Q}_2$ indicates the checkpointing procedure after the completion of a rollback recovery.

3.3 System Availability Formulation

Let $\boldsymbol{\pi}_{ss}$ be the steady-state probability vector of a PH-expanded CTMC, $\boldsymbol{Q}$. Then it can be computed by solving the following linear equation [30]:

$$\boldsymbol{\pi}_{ss}\boldsymbol{Q} = \mathbf{1}, \quad \boldsymbol{\pi}_{ss}\mathbf{1} = 1. \tag{8}$$

The steady-state system availability is given by

$$A_{ss} = \boldsymbol{\pi}_{ss}\boldsymbol{r}, \tag{9}$$

in which $\boldsymbol{r}$ is the reward (column) vector of the PH-expanded CTMC.

On the other hand, when taking account of the system state probability, $\boldsymbol{\pi}(t)$, at an arbitrary time, we define $\boldsymbol{\pi}_0$ as the initial state probability vector. Then the transient probability vector of the states in the PH-expanded CTMC can be expressed by

$$\boldsymbol{\pi}(t) = \boldsymbol{\pi}_0 \exp(\boldsymbol{Q}t). \tag{10}$$

Here, $\exp(\boldsymbol{Q}t)$ is the matrix exponential and is given by the power series $\exp(\boldsymbol{Q}t) = \sum_{k=0}^{\infty} (\boldsymbol{Q}t)^k/k!$. The uniformization technique [31] is well-known as one of the most effective methods to solve Eq. (10). Similar to Eq. (9), the point-wise system availability is formulated by

$$A(t) = \boldsymbol{\pi}(t)\boldsymbol{r}. \tag{11}$$

The point-wise availability, usually known as instantaneous availability, of the system gives the probability that at a specified operation time t, the system is operating [17], [18], [32].

Since the software rejuvenation technique is adopted to achieve the long-run operation of software systems, the problem of interest is to solve the optimization problem of software rejuvenation policy for the target software system, aiming to find the optimal rejuvenation timing that maximizes the steady-state availability of the system.

4. Numerical Examples

In this section, we analyze quantitatively the system availability of a database system, such as a Microsoft SQL server with aperiodic checkpointing and software rejuvenation. Table 3 summarizes the model parameters, whose values are given according to related literatures [11], [33]. For example, the Weibull distribution with IFR is used to model the system failure time related to software aging with mean 10 hours, and the recovery time of rollback recovery is assumed to be log-normal distributed with coefficient of variation (CV) equal to 0.2, indicating a less variation among recovery times.

All general distributions in Table 3 are approximated by PH distributions with appropriate phases. To our knowledge, the large number of phases is needed to approximate accurately the original distribution with a small CV, while in the cases where the values of CV are large, the PH distributions with small phases such as 10 are accurate enough [18]. As examples, Figs. 2 (a) to 2 (d) illustrate the original general distributions and their corresponding PH distributions for $G_{intv}(t)$, $G_{fail}(t)$, $G_{rc}(t)$, and $G_{trig}(t)$. In these figures, the points represent the exact distributions while the dotted

Table 3 The c.d.f.'s of state transitions.

c.d.f.	Distribution	Mean (hrs.)	CV
$G_{intv}(t)$	Lognormal	1-10	0.2
$G_{fail}(t)$	Weibull	10	0.5
$G_{cp}(t)$	Lognormal	0.05	0.2
$G_{load}(t)$	Lognormal	0.5	0.2
$G_{rc}(t)$	Lognormal	0.5	0.2
$G_{trig}(t)$	Lognormal	5–35	0.1
$G_{rej}(t)$	Lognormal	0.5	0.2
$F_{fail}(t)$	Exponential	16.67	1

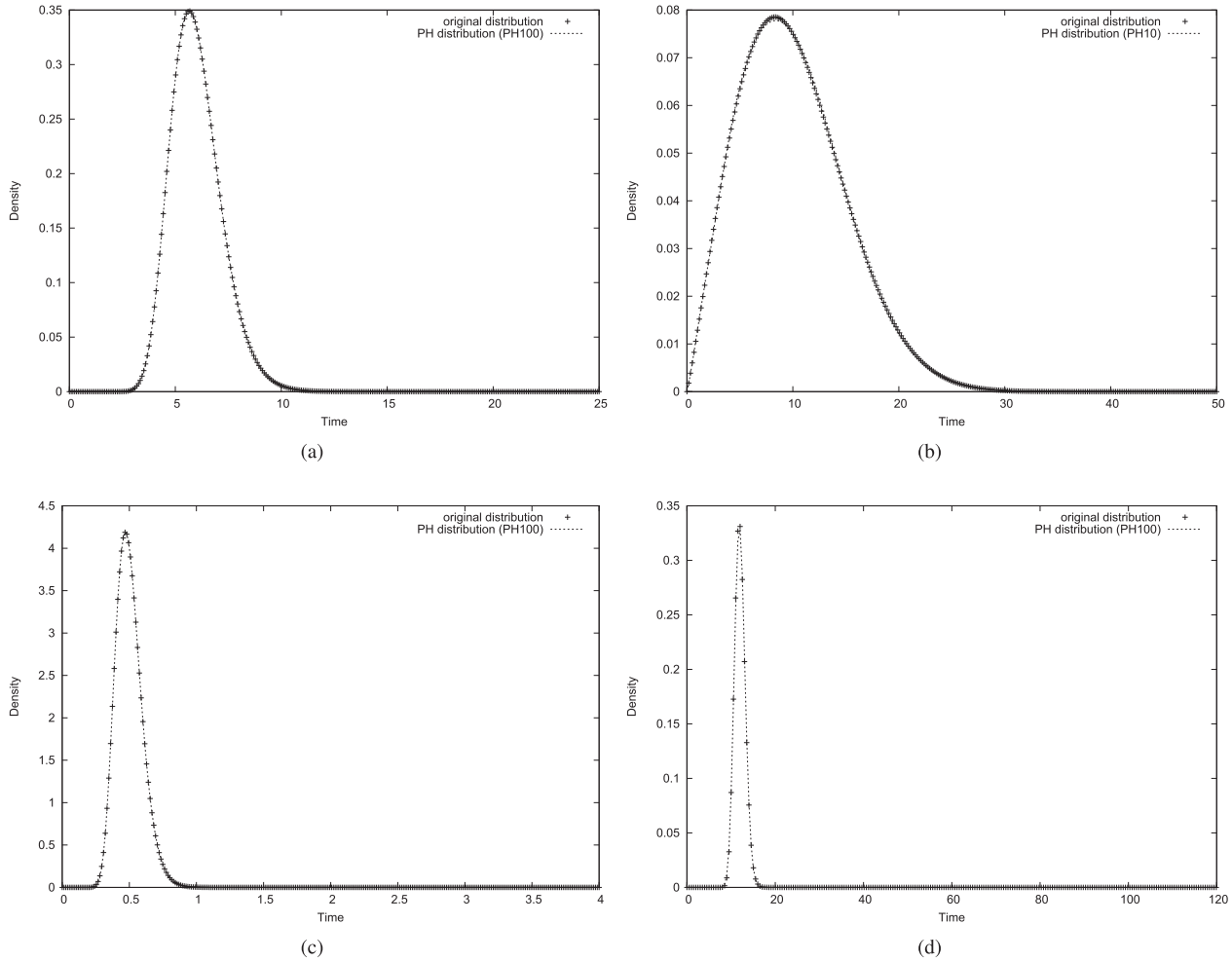


Fig. 2 Approximate PH distributions ((a) $G_{intv}(t)$ (MCI = 6 hrs.), (b) $G_{fail}(t)$, (c) $G_{rc}(t)$, (d) $G_{trig}(t)$ (MRTI = 12 hrs.).

$$\boldsymbol{\alpha}_{fail} = \begin{pmatrix} 4.112827\text{e-}01 & 6.721564\text{e-}02 & 5.504224\text{e-}02 & 1.590405\text{e-}01 & 1.721057\text{e-}01 & 5.630395\text{e-}02 & 4.259914\text{e-}02 & 3.055624\text{e-}02 & 5.838983\text{e-}03 \\ 1.486544\text{e-}05 \end{pmatrix},$$

$$\boldsymbol{T}_{fail} = \begin{pmatrix} -0.6151369 & 0.6151369 & & & & & & & & & \\ & -0.6151506 & 0.6151506 & & & & & & & & \\ & & -0.6151597 & 0.6151597 & & & & & & & \\ & & & -0.6410000 & 0.6410000 & & & & & & \\ & & & & -0.6634417 & 0.6634417 & & & & & \\ & & & & & -0.6911681 & 0.6911681 & & & & \\ & & & & & & -0.7780073 & 0.7780073 & & & \\ & & & & & & & -0.9173205 & 0.9173205 & & \\ & & & & & & & & -1.1043270 & 1.1043270 & \\ & & & & & & & & & -1.347526 \end{pmatrix},$$

$$\boldsymbol{\xi}_{fail} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1.347526 \end{pmatrix}. \tag{12}$$

lines refer to the approximate PH distributions. From these figures, it is clear that both estimated PH distributions can approximate all the general distributions accurately enough. In particular, for the distribution $G_{fail}(t)$ with CV = 0.5, the approximate PH distribution with 10 phases were estimated with parameters in Eq. (12).

Table 4 Steady-state availability of Model-I.

MCI (hrs).	MRTI = 5 hrs.	MRTI = 7 hrs.	MRTI = 10 hrs.	MRTI = 13 hrs.	MRTI = 15 hrs.
1	0.84850	0.86206	0.86796	0.86821	0.86758
2	0.87067	0.88438	0.89024	0.89025	0.88942
3	0.87876	0.89200	0.89788	0.89779	0.89692
4	0.88154	0.89626	0.90174	0.90162	0.90073
5	0.88469	0.89810	0.90415	0.90393	0.90303
6	0.88735	0.89954	0.90576	0.90548	0.90456
7	0.88867	0.90117	0.90666	0.90665	0.90567
8	0.88913	0.90254	0.90741	0.90744	0.90652
9	0.88926	0.90341	0.90818	0.90800	0.90714
10	0.88929	0.90387	0.90892	0.90849	0.90761

Table 5 Steady-state availability of Model-II.

MCI (hrs).	MRTI = 5 hrs.	MRTI = 7 hrs.	MRTI = 10 hrs.	MRTI = 13 hrs.	MRTI = 15 hrs.
1	0.84682	0.85978	0.86492	0.86458	0.86367
2	0.86905	0.88213	0.88718	0.88658	0.88547
3	0.87716	0.88976	0.89482	0.89412	0.89295
4	0.87996	0.89402	0.89868	0.89794	0.89675
5	0.88312	0.89587	0.90109	0.90025	0.89906
6	0.88579	0.89731	0.90270	0.90180	0.90058
7	0.88710	0.89895	0.90360	0.90297	0.90169
8	0.88756	0.90033	0.90435	0.90375	0.90254
9	0.88769	0.90119	0.90513	0.90431	0.90316
10	0.88772	0.90164	0.90586	0.90480	0.90363

In order to evaluate effects of the checkpoint interval and the rejuvenation trigger interval on the system availability, as seen in Table 3, let the mean checkpoint interval (MCI) vary from 1 to 10 hours and the mean rejuvenation trigger interval (MRTI) does from 5 to 35 hours.

4.1 Steady-State Availability

This subsection gives the steady-state system availabilities for both Model-I and Model-II shown in Tables 4 and 5, respectively where MRTI = 5, 7, 10, 13, and 15 hours.

From Table 4, it is observed that the steady-state availability under each case of MRTI increases as the mean checkpoint interval increases, indicating that the shorter checkpoint interval (i.e., more frequent checkpointing) decreases the steady-state system availability. This is due to the fact that only the checkpointing operation is not able to refresh the system aging, but brings the undesired downtime, which results in the decrease of system availability. In addition, we can see that in each case of MCI the steady-state availability increases when MRTI $\leq$ 10 hours and decreases when MRTI $>$ 10 hours. These mean there might exist an optimal MRTI that maximizes the steady-state system availability. Table 5 shows the steady-state availability of Model-II, so that the similar conclusions to Model-I are derived. Since the system is unavailable during the checkpointing operation, it is reasonable to consider that the Model-II involves much more downtime than the Model-I, so that the steady-state availability of Model-II is smaller than that of Model-I.

In the next subsection, we turn our attention to find the optimal MRTIs maximizing the steady-state system availabilities of Model-I and Model-II.

4.2 Optimal Rejuvenation Trigger Timings

Here, we compute the steady-state system availabilities in cases where the mean rejuvenation trigger interval varies from 5 to 35 hours and the mean checkpoint interval is fixed as 1, 2, . . ., and 10, respectively, aiming to determine the optimal rejuvenation timing from the viewpoint of steady-state system availability. The sensitivity of the steady-state availability on the mean rejuvenation trigger interval in the cases of MCI = 2, 4, 6, 8 and 10 for both Model-I and Model-II is illustrated in Fig. 3.

In both of Figs. 3 (a) and 3 (b), we see that the steady-state availabilities show unimodal curves with respect to the mean rejuvenation trigger interval in all the cases of MCI. In other words, there exists an optimal rejuvenation trigger timing that maximizes the steady-state availability of the system in each case. More specifically, when the rejuvenation trigger interval is too short, the time overhead caused by the rejuvenation operation is very sensitive to the steady-state system availability, and decreases the steady-state availability against the motivation of triggering software rejuvenation. On the contrary, if the rejuvenation trigger interval is too long, the downtime due to the system failure also causes relatively smooth decrease in the steady-state system availability.

The optimal rejuvenation trigger timings and their corresponding maximum steady-state availabilities for Model-I and Model-II in all the cases are presented in Table 6. From this table, the optimal MRTIs for all cases of MCI in both of Model-I and Model-II are very similar, that means, the

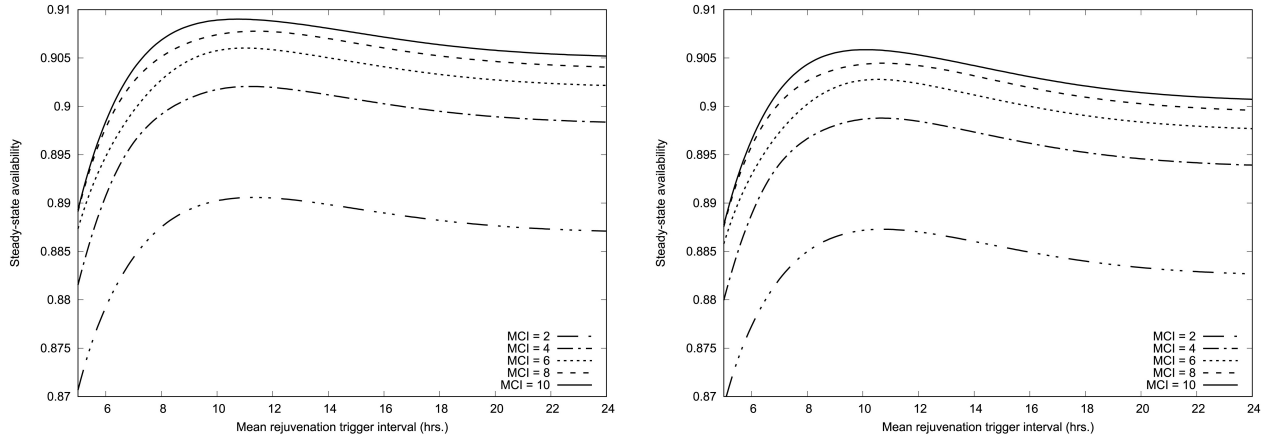


Fig. 3 Sensitivity of the system availabilities on the mean rejuvenation trigger timing for (a) Model-I and (b) Model-II.

Table 6 Optimal rejuvenation trigger timings.

MCI (hrs.)	Model-I		Model-II	
	MRTI (hrs.)	A_{ss}	MRTI (hrs.)	A_{ss}
1	11.6	0.86841	10.9	0.86509
2	11.3	0.89059	10.7	0.88729
3	11.2	0.89819	10.6	0.89491
4	11.2	0.90206	10.7	0.89879
5	11.1	0.90435	10.4	0.90112
6	11.0	0.90603	10.5	0.90279
7	11.3	0.90708	10.8	0.90377
8	11.4	0.90777	10.7	0.90446
9	11.1	0.90838	10.4	0.90515
10	10.7	0.90902	10.1	0.90586

variation in the checkpoint interval has only a very slight influence on the optimal rejuvenation trigger timing. It should be pointed out that the optimal MRTIs of Model-II are slightly smaller than those of Model-I, in other words, for the system where the checkpointing operation is executed immediately after the rollback recovery, a rejuvenation schedule with a relatively shorter rejuvenation trigger interval is preferred, compared with the case where the system state moves to the execution process immediately after the rollback recovery.

4.3 Point-Wise Availability

In this subsection, the point-wise system availabilities of two models are considered to characterize the transient behavior of the system. Figure 4 depicts the point-wise availabilities of Model-I and Model-II by time $t = 12$ hours in the case of MCI = 6 hours and MRTI = 11.0 hours. In particular, the curve in solid line illustrates the point-wise system availability of Model-I with respect to the operational time when the optimal rejuvenation trigger timing is applied for the case of MCI = 6 hours as in Table 6. The oscillations observed can be explained by the fact that at the beginning, no rejuvenation is performed and the point-wise availability first decreases due to aging-related failures, at later it is

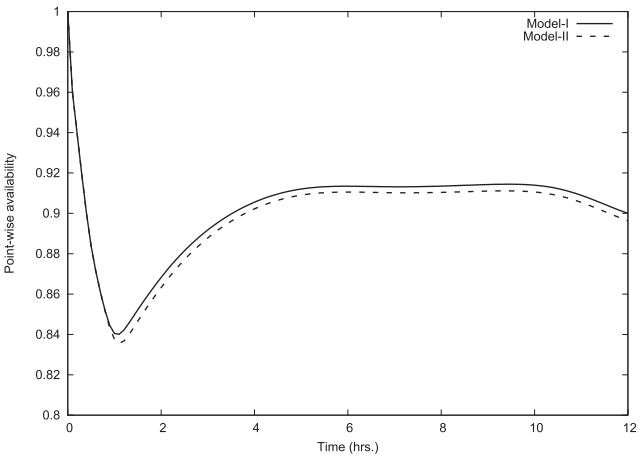


Fig. 4 Point-wise system availabilities for two models by $t = 12$ hours (MCI = 6, MRTI = 11.0).

improved by the execution of rejuvenation. The similar trend on the point-wise availability can be found under Model-II. The point-wise availability of Model-II is slightly lower than that of Model-I, because of an additional checkpointing operation.

On another hand, the point-wise system availability of Model-II under an optimal rejuvenation trigger timing (i.e., MCI = 6, MRTI = 10.5) is shown in solid curve of Fig. 5, while the dotted curve gives the point-wise availability in the case of MCI = 6 hours and MRTI = 11.0 hours. From this figure, the results in two cases are very close before $t = 10$ hours, and the point-wise availability with the optimal rejuvenation trigger timing becomes lower, compared with the case of MRTI = 11.0 hours, after $t = 10$ hours. This implies that the optimal rejuvenation trigger timing, which maximizes the steady-state system availability, may not be the best choice for maximizing the point-wise system availability.

Although Model-I outperforms Model-II from the perspective of system availability (both stationary and

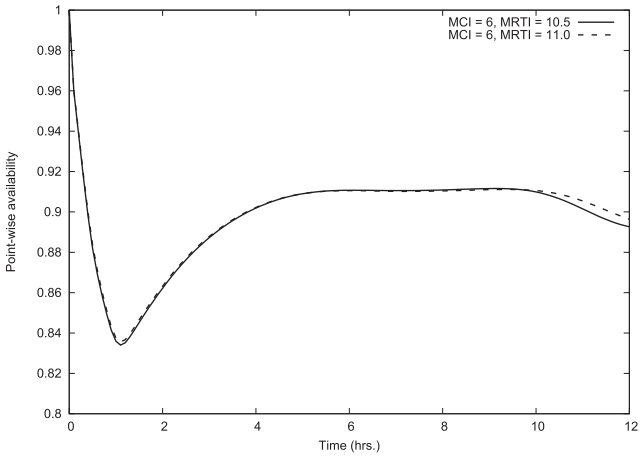


Fig. 5 Point-wise system availability for Model-II.

transient viewpoints), there are no remarkable difference between these results. But significantly, that under Model-II, the starting point of the recovery operation can be updated from the past to the current time.

5. Conclusions

In this paper, we have considered two non-Markovian availability models for the software systems that undergo aperiodic checkpointing and software rejuvenation. Due to the non-Markovian property and complexity, the state transition diagrams were converted to the approximate CTMCs via phase expansion, and then the system availabilities for both models were formulated based on the CTMC analysis. It should be noted that the challenge in our approach is not based on the trivial CTMC analysis, but is reduced to both modeling of the system behavior with multiple competitive events and derivation of the approximate CTMC using the phase expansion. Our numerical experiments have clarified: (i) the checkpointing operation was not able to refresh the system aging, but unfortunately involved downtime, resulting in the decrease of the system availability; and (ii) there existed the optimal rejuvenation trigger timing that maximizes the steady-state system availability.

In the future, we will extend our work to solve more complicated software systems with the other rejuvenation and checkpointing schemes, for example, the time-based periodic rejuvenation or checkpointing for failures related to human error caused by the system operator's mis-operations during checkpointing.

References

[1] M. Grottke and K.S. Trivedi, "Fighting bugs: remove, retry, replicate, and rejuvenate," IEEE Computer, vol.40, no.2, pp.107–109, 2007. DOI:10.1109/MC.2007.55

[2] Y. Huang, C. Kintala, N. Kolettis, and N.D. Funton, "Software rejuvenation: analysis, module and applications," Proc. 25th IEEE International Symposium on Fault Tolerant Computing (FTC'95), IEEE CPS, pp.381–390, 1995. DOI:10.1109/FTCS.1995.466961

[3] K.S. Trivedi and K. Vaidyanathan, "Software aging and rejuvenation," Wiley Encyclopedia of Computer Science and Engineering, pp.1–8, John Wiley & Sons, 2007. DOI:10.1002/9780470050118.ecse394

[4] J. Alonso, R. Matias, E. Vicente, A. Maria, and K.S. Trivedi, "A comparative experimental study of software rejuvenation overhead," Performance Evaluation, vol.70, no.3, pp.231–250, 2013. DOI:10.1016/j.peva.2012.09.002

[5] J. Zheng, H. Okamura, L. Li, and T. Dohi, "A comprehensive evaluation of software rejuvenation policies for transaction systems with Markovian arrivals," IEEE Trans. Rel., vol.66, no.4, pp.1157–1177, 2017. DOI:10.1109/TR.2017.2741526

[6] 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, 2016. DOI:10.1109/TR.2016.2570539

[7] T. Dohi, J. Zheng, H. Okamura, and K.S. Trivedi, "Optimal periodic software rejuvenation policies based on interval reliability criteria," Reliability Engineering and System Safety, vol.180, pp.463–475, 2018. DOI:10.1016/j.ress.2018.08.009

[8] Y. Zhang and K. Chakrabarty, "Fault recovery based on checkpointing for hard real-time embedded systems," Proc. 18th IEEE Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'03), IEEE CPS, pp.320–327, 2003. DOI:10.1109/DFTVS.2003.1250127

[9] S. Fukumoto, N. Kaio, and S. Osaki, "Optimal checkpointing policies using the checkpointing density," Journal of Information Processing, vol.15, no.1, pp.87–92, 1992.

[10] A. Ranganathan and S.J. Upadhyaya, "Performance evaluation of rollback-recovery techniques in computer programs," IEEE Trans. Rel., vol.42, no.2, pp.220–226, 1993. DOI:10.1109/24.229490

[11] T. Dohi, S. Osajima, N. Kaio, and S. Osaki, "On the effects of checkpoint institution methods for a macroscopic database model," Electronics and Communications in Japan (Part III: Fundamental Electronic Science), vol.83, no.9, pp.23–33, 2000.

[12] H. Okamura and T. Dohi, "Comprehensive evaluation of aperiodic checkpointing and rejuvenation schemes in operational software system," Journal of Systems and Software, vol.83, no.9, pp.1591–1604, 2010. DOI:10.1016/j.jss.2009.06.058

[13] G. Levitin, L. Xing, and L. Luo, "Joint optimal checkpointing and rejuvenation policy for real-time computing tasks," Reliability Engineering and System Safety, vol.182, pp.63–72, 2019. DOI:10.1016/j.ress.2018.10.006

[14] J. Zheng, H. Okamura, and T. Dohi, "A phase expansion for non-Markovian availability models with time-based aperiodic rejuvenation and checkpointing," Communications in Statistics - Theory and Methods, vol.49, no.15, pp.3712–3729, 2020. DOI:10.1080/03610926.2019.1708400

[15] G. Bolch, S. Greiner, H. De Meer, and K.S. Trivedi, Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications, 2nd ed., John Wiley & Sons, New York, NY, USA, 2006.

[16] K.S. Trivedi and A. Bobbio, Reliability and Availability Engineering: Modeling, Analysis, and Applications, Cambridge University Press, 2017.

[17] J. Zheng, H. Okamura, and T. Dohi, "Security evaluation of a VM-based intrusion-tolerant system with pull-type patch management," Proc. 2019 IEEE 19th International Symposium on High Assurance Systems Engineering (HASE'19), IEEE CPS, pp.156–163, 2019. DOI:10.1109/HASE.2019.00032

[18] J. Zheng, H. Okamura, and T. Dohi, "A transient interval reliability analysis for software rejuvenation models with phase expansion," Software Quality Journal, vol.28, pp.173–194, 2020. DOI:10.1007/s11219-019-09458-1

[19] K. Balakrishnan, Exponential Distribution: Theory, Methods and Applications, Routledge, 2018.

[20] H. Rinne, The Weibull Distribution: A Handbook, CRC press, 2008.

[21] B. Schroeder and G.A. Gibson, "A large-scale study of failures in high-performance computing systems," IEEE Transactions on Dependable and Secure Computing, vol.7, no.4, pp.337–350, 2010. DOI:10.1109/TDSC.2009.4
[22] O. Connor, Practical Reliability Engineering, John Wiley & Sons, 2012.
[23] R. Vinayak and S. Dharmaraja, "Semi-Markov modeling approach for deteriorating systems with preventive maintenance," International Journal of Performability Engineering, vol.8, no.5, pp.515–526, 2012. DOI:10.23940/ijpe.12.5.p515.mag
[24] K. Wolter, "Stochastic models for restart, rejuvenation and checkpointing," Habilitation Thesis, Humboldt-University, Institut Informatik, Berlin, Tech. Rep., 2008.
[25] H. Okamura and T. Dohi, "Fitting phase-type distributions and Markovian arrival processes: algorithms and tools," Principles of Performance and Reliability Modeling and Evaluation, F. Lance and P. Antonio (eds.), pp.49–75, Springer, 2016. DOI:10.1007/978-3-319-30599-8_3
[26] P. Kemper, D. Müller, and A. Thümmler, "Combining response surface methodology with numerical methods for optimization of Markovian models," IEEE Transactions on Dependable and Secure Computing, vol.3, no.3, pp.259–269, 2006. DOI:10.1109/TDSC.2006.28
[27] A. Cumani, "On the canonical representation of homogeneous Markov processes modelling failure-time distributions," Microelectronics Reliability, vol.22, no.3, pp.583–602, 1982. DOI:10.1016/0026-2714(82)90033-6
[28] H. Okamura, T. Dohi, and K.S. Trivedi, "Improvement of expectation-maximization algorithm for phase-type distributions with grouped and truncated data," Applied Stochastic Models in Business and Industry, vol.29, no.2, pp.141–156, 2013. DOI:10.1002/asmb.1919
[29] T. Dayar, Analyzing Markov Chains Using Kronecker Products: Theory and Applications, Springer Science & Business Media, 2012.
[30] K.S. Trivedi, Probability and Statistics with Reliability, Queuing, and Computer Science Applications, 2nd ed., John Wiley & Sons, 2001.
[31] A. Reibman and K.S. Trivedi, "Numerical transient analysis of Markov models," Computers & Operations Research, vol.15, no.1, pp.19–36, 1988. DOI:10.1016/0305-0548(88)90026-3
[32] G.H. Sandler, System Reliability Engineering, Prentice-Hall, Englewood Cliffs, New Jersey, 1963.
[33] C.H.C. Leung and E. Currie, "The effect of failures on the performance of long-duration database transactions," The Computer Journal, vol.38, no.6, pp.471–478, 1995. DOI:10.1093/comjnl/38.6.471

Junjun Zheng received the B.S.E. degree in engineering from Fujian Normal University, Fuzhou, China, in 2010, and the M.S. and D.Eng. degrees in engineering from Hiroshima University, Higashihiroshima, Japan, in 2013 and 2016, respectively. In 2016 and 2017, he was a Visiting Researcher with the Department of Information Engineering, Graduate School of Engineering, Hiroshima University. Since 2018, he has been an Assistant Professor with the Department of Information Science and Engineering, Ritsumeikan University, Japan. His research interests include performance evaluation and dependable computing. Dr. Zheng is a member of the Operations Research Society of Japan, the Reliability Engineering Association of Japan, the Institute of Electrical, Information and Communication Engineers, and the Institute of Electrical and Electronics Engineers.

Hiroyuki Okamura received the B.S.E., M.S., and D.Eng. degrees in engineering from Hiroshima University, Higashihiroshima, Japan, in 1995, 1997, and 2001, respectively. In 1998, he joined Hiroshima University as an Assistant Professor, where he has been an Associate Professor with the Department of Information Engineering, Graduate School of Engineering, since 2003. He is now a Processor with the Department of Information Engineering, Graduate School of Engineering, since 2018. His research interests include performance evaluation, dependable computing, and applied statistics. Dr. Okamura is a member of the Operations Research Society of Japan, the Institute of Electrical, Information and Communication Engineers, the Japan Society for Industrial and Applied Mathematics, the Information Processing Society of Japan, the Association for Computing Machinery, and the Institute of Electrical and Electronics Engineers.

Tadashi Dohi received the B.S.E., M.S., and D.Eng. degrees in engineering from Hiroshima University, Higashihiroshima, Japan, in 1989, 1991, and 1995, respectively. In 1992 and 2000, he was a Visiting Researcher with the Faculty of Commerce and Business Administration, University of British Columbia, Canada, and the Hudson School of Engineering, Duke University, USA, respectively, on the leave from Hiroshima University. Since 2002, he has been a Professor with the Department of Information Engineering, Graduate School of Engineering, Hiroshima University. His research interests include reliability engineering, software reliability, and dependable computing. Dr. Dohi is a member of the Operations Research Society of Japan, the Institute of Electrical, Information and Communication Engineers, the Information Processing Society of Japan, the Reliability Engineering Association of Japan, and the Institute of Electrical and Electronics Engineers. He is also an Associate Editor of the IEEE TRANSACTIONS ON RELIABILITY among others.