A Study on Sensitivity Approaches for Dependable Systems Design

(ディペンダブルシステムデザインのための感度分析アプローチに関する研究)

Dissertation submitted in partial fulfillment for the
degree of Doctor of Engineering

Junjun Zheng

Under the supervision of
Associate Professor Hiroyuki Okamura

Dependable Systems Laboratory,
Department of Information Engineering,
Graduate School of Engineering,
Hiroshima University, Higashi-Hiroshima, Japan

March 2016

A Study on Sensitivity Approaches for Dependable Systems Design

Dissertation submitted in partial fulfillment for the degree of Doctor of Engineering

Junjun Zheng

Under the supervision of
Associate Professor Hiroyuki Okamura

Dependable Systems Laboratory,
Department of Information Engineering,
Graduate School of Engineering,
Hiroshima University, Higashi-Hiroshima, Japan

March 2016

A Study on Sensitivity Approaches for Dependable Systems Design

Junjun Zheng

Hiroshima University

CONTENTS IN BRIEF

CONTENTS

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude to Associate Professor Hiroyuki Okamura, the supervisor of my study, for his constant guidance, patience, kind advice and continuous encouragement throughout the progress of this work. I am especially grateful for his confidence and the freedom he gave me to do this work. Also, my sincere thanks go to Professor Tadashi Dohi and Professor Chuzo Iwamoto, for their insightful and invaluable comments and constant support and encouragement, but also for the hard question which incented me to widen my research from various perspectives.

I extend my special pleasure to acknowledge the hospitality and encouragement of the past and present members of the Dependable Systems Laboratory, Department of Information Engineering, Graduate School of Engineering, Hiroshima University.

I owe special gratitude to my family for their continuous and unconditional support, understanding, and love.

Finally, I would like to thank the China Scholarship Council (CSC) for the financial support.

J. Zheng
Hiroshima, Japan
March, 2016

ABSTRACT

Dependability is an all-encompassing definition for reliability, availability, safety and security, and is required in computer applications such as safety-critical control systems for road vehicles, airplanes and medical devices, and business-critical systems for e-commerce and financial transactions. To assure high dependability of systems, redundancy has been widely applied and plays an important role in enhancing system reliability. In general, there are two commonly-used types of system designs; component (or subsystem) redundancy and environmental redundancy. The component redundancy is the use of additional components or subsystems beyond the number actually required for the system to operate reliably, such as k-out-of-n redundant systems with spares. In the environmental redundancy, the system re-executes some of its initialization procedures to obtain a fresh environment that might in turn make the system less prone to failures, such as rejuvenation techniques that reboot the system before failures occur. In fact, redundancy increases not only the complexity of a system, but also the complexity of associated problems such as common-mode error. Thus, in order to detect the optimal design of systems, model-based analysis is important in the system design. Fault trees (FTs), reliability block diagrams (RBDs), Markov chains (e.g., discrete-time Markov chain (DTMC) and continuous-time Markov chain (CTMC)) and stochastic Petri nets (SPNs) are commonly-used techniques for model-based dependability analysis of computer systems.

One of the advantages of model-based analysis is sensitivity analysis, which can identify both dependability bottlenecks and critical parameters to improve system dependability. The sensitivity analysis plays an important role in the optimization of system in the design phase. In particular, the sensitivity analysis is effective to detect the critical components in the system. Generally, the parametric sensitivity and component importance (i.e., component-wise sensitivity) analysis are widely used sensitivity approaches to detect the design sensitivity of system. In addition, some extensive sensitivity analyses are devoted to evaluate the environmental sensitivity such as the survival probabilities in fault-tolerant systems, indicating how expected survivability would change with varying model parameters.

This thesis considers the sensitivity approaches for dependable systems design. Concretely, we consider the design sensitivity for virtualized systems (in Chapters 3 and 4) and real-time computing systems (in Chapter 5). For evaluating the environmental sensitivity, virtual machine (VM)-based intrusion tolerant systems (in Chapter 6) are taken into account. The thesis is organized as follows.

Chapter 2 presents the preliminaries of the commonly-used techniques in model-based dependability analysis and the sensitivity analysis.

In Chapter 3, we focus on the component importance analysis regarding system availability of virtualized system design and develop a method to evaluate the importance of components. In the past literature, most of people focused on estimating the performance of an entire virtualized system such as the system availability. One of the important things in the system design is how to allocate system resources to components. To the best of our knowledge, there are a few papers to deal with such design problems of virtualized system. Our analysis can provide quantitative importance of all the components for system availability thereby formulating a resource allocation problem to improve system availability.

Chapter 4 is devoted to a novel state-of-the-art Markov-based component-wise sensitivity analysis. We apply it to the CTMC model of the live migration in a virtualized system, and reveal the component importance of live migration without using structure function.

In Chapter 5, we turn our attention to the component importance analysis of a real-time computing system in the presence of common-cause failures (CCFs). The CCFs are known as a risk factor of the degradation of system reliability in practice, and make it difficult to evaluate the component importance measures analytically. Thus it is important to evaluate the effect of CCFs especially in the real-time computing systems.

Chapter 6 discusses the survivability analysis of a VM-based intrusion tolerant system in the presence of intrusion. The survivability is the capability of a system to provide its services in a timely manner even after intrusion and compromise occur, which is the sensitivity of environmental changes.

Finally this thesis is summarized with some remarks and future works in Chapter 7.

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

FT	Fault Tree
RBD	Reliability Block Diagram
DTMC	Discrete-Time Markov Chain
CTMC	Continuous-Time Markov Chain
SPN	Stochastic Petri Net
VM	Virtual Machine
CIA	Confidentiality, Integrity and Availability
QoS	Quality-of-Service
DoS	Deny-of-Service
SMR	State Machine Replication
ABS	Anti-lock Braking System
CCF	Common-Cause Failure
MRM	Markov Reward Model
ODE	Ordinary Differential Equation
OS	Operating System
VMM	Virtual Machine Manager
SAN	Storage Area Network
MTTF	Mean Time To Failure
MTTR	Mean Time To Repair
PM	Processing Module
SM	Shared Memory
DS	Digital Switch

I/O	Input/Output
A/D	Analog-to-Digital
D/A	Digital-to-Analog
AS	Agreement Service
CRCE	Common Root Cause Event
i.i.d.	independent and identically distributed
c.d.f.	cumulative distribution function
p.d.f.	probability density function
p.m.f.	probability mass function

SYMBOLS

Chapter 2 Preliminaries

$X(t)$: a time-homogeneous CTMC.
$\mathcal{S}$: denumerable state space.
$\boldsymbol{\pi}(t)$: state probability (row) vector.
$\boldsymbol{Q}$: an n-by-n square matrix called the infinitesimal generator.
q_{ij} : the transition rate from state i to state j.
$\mathbf{1}$: a column vector whose elements are 1.
$\mathbf{0}$: a column vector whose elements are 0.
$\boldsymbol{\pi}_0$: the initial state probability vector.
$\boldsymbol{A}$: a matrix.
$\boldsymbol{I}$: an identity matrix.
$\boldsymbol{\pi}_{ss}$: the steady-state probability vector.
ρ : a reward function in an MRM.
$Y(t)$: a reward process.
$\boldsymbol{r}$: a reward (column) vector.
θ : a model parameter of MRM.
$\boldsymbol{s}_{ss}(\theta)$: sensitivity function of steady-state probability vector with respect to a model parameter θ.
$\boldsymbol{s}(t, \theta)$: sensitivity function of transient probability vector with respect to a model parameter θ.
$\boldsymbol{S}(\theta)$: the derivative of $\boldsymbol{Q}$ with respect to a model parameter θ.

Chapter 3 Component Importance Measures for Virtualized System

$\mathcal{U}$: the set of up state.
$\mathcal{D}$: the set of down state.
λ : failure rate of component.
μ : repair rate of component.
α : the rate of summoning inherent in the component.

χ_{SAN} : the rate of copying data.

β : the rate of rebooting a component.

b : the coverage of transient failures.

A_i : the steady-state availability of component i.

π_k : the steady-state probability of state k.

λ_i : the failure rate of component i.

μ_i : the repair rate of component i.

$I_{\lambda,i}$: importance measure with respect to failure rate.

$I_{\mu,i}$: importance measure with respect to repair rate.

$t_{i,j}$: the transition rate from state i to j.

$\tilde{\lambda}$: the equivalent failure rate of component.

$\tilde{\mu}$: the equivalent repair rate of component.

Chapter 4 Importance Measures for Virtualized System with Live Migration

$\boldsymbol{Q}$: the infinitesimal generator of CTMC availability model for live migration.

$\boldsymbol{\pi}_{ss}$: the steady-state probability vector of CTMC model for live migration.

$\boldsymbol{\xi}_i$: a 0-1 vector whose elements are 1 in the state where component i is up.

$\boldsymbol{J}$: a matrix whose elements are the sensitivities of availabilities of components with respect to model parameters.

$\boldsymbol{z}$: a column vector whose elements are the sensitivities of system availability with respect to each model parameter.

Chapter 5 Component Importance Measures for Real-time Computing Systems

λ_i : the rate of independent failure killing single component.

λ_d : the rate of dependent failures killing all components.

λ : the overall failure rate of a particular component.

β : the probability that a failure in a specific component causes all component to fail.

$\boldsymbol{x}$: the state vector of system.

x_k : a binary variable which represents the condition of component k.

$\phi(\boldsymbol{x})$: structure function of system.

$P_{\boldsymbol{x}}(t)$: a certain probability mass function of the system being in state $\boldsymbol{x}$ at time t.

Ω : the state space of system.

$R(t)$: reliability function of system.

$R_k(t)$: reliability function of component k at time t.

$\boldsymbol{Q}_S$: the infinitesimal generator of system.

$\delta_k(\boldsymbol{x})$: the first derivative of structure function with respect to the state condition of component k.

$IB_k(t)$: Birnbaum importance of component k at time t.

AIB_k : Birnbaum availability importance measure of component k.

$RIB_k(t)$: Birnbaum reliability importance measure of component k at time t.

$ICF_k(t)$: criticality importance of component k at time t from the unreliability point of view.

$ICR_k(t)$: criticality importance of component k at time t from the reliability point of view.

$AICR_k$: criticality availability importance measure of component k.

$RICR_k(t)$: criticality reliability importance measure of component k at time t.

$AIU_{k,\lambda}$: availability upgrading function for component k regarding to failure rate.

$AIU_{k,\mu}$: availability upgrading function for component k regarding to repair rate.

$RIU_{k,\lambda}(t)$: reliability upgrading function for component k at time t.

$\lambda_k(t)$: time-dependent failure rate of component k.

Chapter 6 Survivability Analysis of VM-based Intrusion Tolerant Systems

n : the number of initially activated VMs.

r : the number of additionally activated VMs.

m : the maximum number of rounds.

f : the tolerance level of a system.

n_k : the number of VMs at the k-th round.

f_k : the tolerance level at the k-th round.

γ : the arrival rate of intrusion for each VM.

S : an random variable representing the processing time for a request in one VM.

$G(t)$: the cumulative distribution function for S.

p_I : the probability that a VM is intruded during it processes one request.

$G^*(\gamma)$: the Laplace-Stieltjes transform of $G(t)$.

N_k : the number of normal VMs at the end of the k-th round.

$N_k; k \geq 1$: a discrete-time Markov chain.

(v, k) : a state of DTMC indicating that there are v normal VMs at the end of the k-th round.

$\boldsymbol{\pi}_k$: the probability vector at the end of the k-th round.

$\boldsymbol{P}_{k-1}$: the transition matrix from states in $k-1$-round to states in k-th round.

f_m : the tolerance level of system with maximum number of rounds, m.

q : a randomization parameter satisfying $q = \sup_{i \in S} |q_{ii}|$.

$\boldsymbol{P}$: the transition probability matrix of DTMC.

Appendices A and B

U : the upper (right) bound.

ε : the error tolerance.

I_S : the performance index of system.

I_k : the performance index of component k.

$\boldsymbol{r}_S$: the reward vector of system.

$\boldsymbol{r}_k$: the reward vector of component k.

$\boldsymbol{\pi}$: a state probability vector of the underlying CTMC at arbitrary time point.

$\boldsymbol{\theta}$: model parameter vector.

δ_j : the deviation of I_S with respect to θ_j which are not correlated to the deviations of $I_1, \ldots, I_K$.

$\boldsymbol{z}$: a column vector whose elements are the sensitivities of system performance index with respect to each model parameter.

$\boldsymbol{u}$: a column vector whose elements are the sensitivities of system performance index with respect to each performance index of component.

$\boldsymbol{\delta}$: a column vector whose elements are the deviation of I_S with respect to each model parameter.

$\boldsymbol{J}$: a matrix whose elements are the sensitivities of performance indices of components with respect to model parameters.

$||\boldsymbol{\delta}||_2$: a 2-norm of vector $\boldsymbol{\delta}$.

T : the transpose operator.

CHAPTER 1

INTRODUCTION

In this chapter, we first give an introduction to dependability and dependable system designs. We then introduce the model-based dependability analysis and sensitivity analysis. Also, two commonly-used dependable system applications are mentioned; virtualized systems and real-time computing systems. In particular, the intrusion due to malicious attacks and dependent failures among components in such systems are taken into account.

1.1 Background

Dependability is an all-encompassing definition for reliability, availability, safety and security, and is required in computer applications such as safety-critical control systems for road vehicles, airplanes and medical devices, and business-critical systems for e-commerce and financial transactions. To assure high dependability of systems, redundancy has been widely applied and plays an important role in enhancing system reliability. In general, there are two commonly-used types of system designs; component (or subsystem) redundancy and environmental redundancy. The component redundancy is the use of additional components or subsystems beyond the number actually required for the system to operate reliably, such as k-out-of-n redundant systems with spares [1]. In the environmental redundancy, the system re-executes some of its initialization procedures to obtain a fresh environment that might in turn make the system less prone to failures [2], such as rejuvenation techniques that reboot the system before failures occur. In fact, redundancy increases not only the complexity of a system, but also the complexity of associated problems such as common-mode error. Thus, in order to detect the optimal design of systems, model-based analysis is important in the system design. Fault trees (FTs), reliability block diagrams (RBDs), Markov chains (e.g., discrete-time Markov chain (DTMC) and continuous-time Markov chain (CTMC)) and stochastic Petri nets (SPNs) are commonly-used techniques for model-based dependability analysis of computer systems.

One of the advantages of model-based analysis is sensitivity analysis, which can identify both dependability bottlenecks and critical parameters to improve system dependability. The sensitivity analysis plays an important role in the optimization of system in the design phase. In particular, the sensitivity analysis is effective to detect the critical components in the system and helps to select the best system configuration. Generally, the parametric sensitivity and component importance analysis are widely used sensitivity approaches to detect the design sensitivity of system. The parametric sensitivity is defined as the first derivatives of dependability indices with respect to model parameters and applied to optimizing system dependability by combining the mathematical programming as well as the evaluation of effects on parameters. The component importance analysis is more preferred than the parametric sensitivity analysis in the dependability engineering. The component importance analysis, called the component-wise sensitivity analysis, is to estimate the first derivatives of dependability measures of system with respect to dependability measures of components. Thus the component importance analysis can detect the critical components from the dependability point of view directly. In addition, some extensive sensitivity analyses are devoted to evaluate the environmental sensitivity such as the survival probabilities in fault-tolerant systems, indicating how expected survivability would change with varying model parameters.

In recent years, some technologies have been developed to improve the system dependability. For example, cloud computing has emerged as an important trend in software industry. Cloud computing is a promising system architecture to ensure the high dependability, and has widely spread in many of computer applications. In general, cloud computing is defined as a style of computing in which dynamically scalable and virtualized resources are provided as a service over the Internet [3]. Along with ubiquitous computing, cloud computing drastically grows as a key technology of next generation of computing recently. One of the core technologies that support cloud computing for high dependable system is virtualization of system resources. Roughly speaking, virtualization is to create software components that emulate behavior of hardware units and platform, and is to control them in a software platform. The most popular virtualization is to create a virtual machine (VM) that behaves an actual computer on the platform such as VMware, Xen and Hyper-V. One of the advantages of virtualization is that physical servers can be integrated into a platform that manages their VMs. This is one of techniques to enhance system utilization by integrating physical servers with low utilization. Such integration leads to saving energy of server operations. Moreover, if two physical servers have the same platform that can drive VMs, we exploit the live migration between them [4]. The live migration is a technique that allows a server administrator to move a running VM of application between different physical machines without disconnecting the client or application. In a virtualized system, the live migration can improve the system dependability by migrating a failed VM on a platform to another platform. The virtualization has many advantages of managing the system in practical situation, and the configuration of virtualized system is more flexible than the ordinary server architecture that has physical constraints. However, due to such diversified computing environments, it is more difficult to determine the best of virtualized system design and to predict virtualized system performance in its operation compared to conventional non-virtualized system designs.

On the other hand, with the rapid development of computer systems, the systems face the threat of intrusion due to malicious attacks that exploit security holes or vulnerabilities. Typically, these attacks are caused by malicious codes, for example, viruses, worms, Trojan horses, and attack scripts. Since security holes and vulnerabilities are essentially software bugs, they can be detected and removed in system testing. However, system and applications are released while their security holes or vulnerabilities still remain in practice, even if developers carefully execute system testing. In fact, the number of security incidents is increasing as the software system is used in wide application area. Generally, the attributes of security are categorized to confidentiality, integrity and availability (CIA). The system is evaluated with CIA criteria from the viewpoint of security, and highly-secure system should be designed to keep high levels of these criteria. Nowadays, security is treated as a QoS (Quality of Service) attribute at par with other QoS attributes such as performance. Although there exist many types of security incidents such as website defacement and deny-of-service (DoS) attack, all of them are related to CIA. Generally, the security failure is defined by the system failure that causes the degradation of CIA. This thesis focuses on the intrusion as a typical security failure that is a main cause of degradation of confidentiality and integrity in the system. In addition, the intrusion is regarded as a Byzantine failure of system, which is defined as an arbitrary failure, namely, a process with Byzantine failure causes deviation from its normal behavior created by an algorithm [5]. To counteract the intrusion, we need to take care of every action by a failed and intruded process such as sending fake messages, not sending any messages, and disrupting other processes. Security solutions such as intrusion detection systems, intrusion prevention systems, and firewalls are designed to protect against the malicious attacks [6]. However, these solutions are not always possible to prevent all kinds of attacks. For example, intrusion detection systems require the signatures of attacks that have been reported. Thus it is difficult to prevent the intrusion completely. Therefore, to ensure that systems are correctly and safely available even in the presence of intrusion, it is necessary to develop a mechanism to tolerate intrusions. Intrusion tolerance is the ability of a system to continuously provide correct, but possibly degraded, services even if the system is intruded [7]. Recently, several researchers pay attention to virtualization technology to build more trusted computing environments, based on the state machine replication (SMR) [8], where each service replica runs on a different VM. In virtualized computing environments, attackers search for exploitable security holes or vulnerabilities in deployed virtualization environments to compromise one of its software processes. The common attack patterns are virtualized botnets, virtual code injection attacks, and hypervisor traversal attacks [9].

In this thesis, we also consider the real-time computing systems which are widely used in our daily lives, e.g., anti-lock braking system (ABS) in cars, telephone networks, and patient care systems. A real-time computing system is a system in which timeliness is as important as correctness of its outputs [10]. A delayed output in real-time systems is not acceptable even if it has a correct value. Thus, the Dependability of these systems is more important. Generally, to guarantee the high Dependability of real-time computing systems, redundancy has been commonly applied. However, redundancy increases not only the complexity of a system but also the complexity of associated problems such as common-cause failures (CCFs). The CCF is also called the dependent failure, which is defined as any condition or event that affects several components inducing their simultaneous failure or malfunction [11], and is synonymous with the simultaneous failures and multiple failures. The CCF is known as a risk factor of the degradation of system reliability in practice, and makes it difficult to evaluate the component importance measures analytically.

1.2 Related Works

In virtualized system, the flexibility often causes the difficulty to determine the best configuration of the virtualized system. For such issues, researchers tried to evaluate the performance of system configuration of the virtualized system quantitatively by using probabilistic models.

On the performance index, Kundu et al. [12] presented statistical models using regression and artificial Neural networks. Also, Okamura et al. [13] proposed a queueing model to evaluate energy efficiency of virtualized system design. On the system index for reliability and availability, Cully et al. [14] and Farr et al. [15] built and evaluated their schemes to enhance system availability in virtualized system design. Myint and Thein [16] also evaluated a system architecture combining virtualization and rejuvenation. Vishwanath and Nagappan [17] collected operation data of virtualized system and performed statistical analysis to reveal a causal relationship between server failures and hardware repairs. Kim et al. [18] focused on failure modes of virtualized system and presented availability evaluation using fault trees and continuous-time Markov chains. Also Matos et al. [19] developed the CTMC model representing the dynamic behaviors of live migration in the virtualized system.

Recently, several researchers pay attention to virtualization technology to build more trusted computing environments. For example, Junior et al. [20] proposed a shared memory based intrusion tolerant system in which each service replica runs in a different VM and the communication among the replicas is performed through an abstraction of a shared memory. In this system, the required number of replicas (VMs) is reduced from $3f + 1$ to $2f + 1$. Note that, f is the tolerance level of system, namely, the maximum number of intruded replicas (VMs) that are tolerated to the extent that the system behaves normal. Moreover, Lau et al. [21] considered a similar approach to [20]. Concretely, they described the algorithms for processing requests sent by clients in the presence of malicious attacks and succeeded to reduce the minimum number of replicas to $f + 1$ by using an agreement service that provides a voting process and puts a signature into an agreed response. However, once Byzantine failure occurs, $2f + 1$ VMs are needed. Thus there is the situation where Lau's method is very wasteful, and it has not been clear what situation Lau's method functioned well.

On the other hand, some researches considered the real-time computing systems with failure dependencies. For example, Fricks et al. [11] studied the effect of failure dependencies in a real-time computing system using SPNs and CTMCs. Also, they classified some different types of failure dependencies that can arise in the reliability model of real-time computing system, and illustrate how several of the failure dependencies can be incorporated in SPN model. Based on their research, it is realized that failure dependencies highly influence the system reliability and that failure dependencies therefore never should be ignored.

Also, in [22], Fricks et al. considered three kinds of component importance measures for Markov reward model (MRM), in contrast to the common method of computing importance measures using combinatorial models (e.g., fault tree (FT) and reliability block diagram (RBD)) and structure function which represents the relationship between components failures and system failure, and can be obtained using symbolic analytical logic techniques such as FT and RBD analysis. Pan et al. [23] presented a quantitative method to evaluate the importance of each CCF event. More precisely, they divided the CCFs into two groups; one with a clear relationship between the causes and effects and the other with no such relationship. For the first group of CCFs, they evaluate the structure function importance and probability importance of the common root cause events modeled using FT. On the other hand, they considered the Birnbaum importance for the second group of CCFs which are achieved by using parametric model.

CHAPTER 2

PRELIMINARIES

This chapter presents the preliminaries of the commonly-used techniques including non-state-space (fault tree and reliability block diagram) and state-space models (continuous-time Markov chain) in model-based dependability analysis and the sensitivity analysis of Markov models.

2.1 Symbolic Analytical Logic Techniques

In reliability engineering, fault trees (FTs) and reliability block diagrams (RBDs) are both symbolic analytical logic techniques that can be applied to analyze system reliability and related characteristics. In the system modeling, the FTs and RBDs are non-state-space models and used to describe the static relationships between component failures and system failure. Although the symbols and structures in the diagrams of two techniques are different, most of the logical constructs in a FT diagram can also be modeled with a RBD. In this section, we present a brief introduction to FT and RBD analysis.

2.1.1 Fault Tree Analysis

Fault trees (FTs) are a graphical representation of a system hazard depicting the underlying causal events using Boolean logic gates (e.g., OR/AND

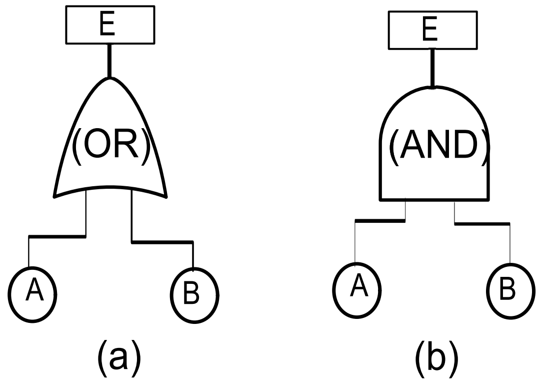

Figure 2.1 OR/AND gates.

gates) and are used to reason about and/or quantifiably estimate the potential cause(s) of a system failure [24, 25]. Fault tree analysis is a backward search technique that starts from a system failure and works towards the initiating events (i.e., the causing events that may lead to the system hazard/failure) [25]. Fig. 2.1 (a) shows OR gate, it defines that output event occurs if any one of the input events occurs. And AND gate is shown in Fig. 2.1 (b), it defines that output event occurs if all input events occur. In the figure, circle A and B mean basic event with sufficient data, and rectangle E means event represented by a gate.

2.1.2 Reliability Block Diagram

Reliability block diagrams (RBDs) are an adequate technique for describing systems, when considering the reliability and availability of systems. Concretely, the RBD is top level description for the system that illustrates how components and subsystems reliabilities contribute to the success or failure of a system, and allows us to model the failure relationships of complex systems. It is a well-accepted method in industrial practice.

However, like the FT, the RBD cannot be used to describe the dynamic behavior of systems. When using the either of FTs and RBDs, we need the following assumptions; (i) all failure and repair events in the system are stochastically independent, and (ii) each component can be in two states only, that is, active and failed. In contrast, Markov reward models (MRMs) provide a powerful mathematical framework for computing system state probabilities and thus quantifying a system under study. The modeling power of MRMs is much higher than that of FTs and RBDs; each component can be described by an arbitrary number of states (e.g., active, passive, and several failed states), and arbitrary inter-component dependencies (such as failure propagation, failures with a common cause, or limited repair capacities) can be specified. Consequently, MRMs are an adequate formal model for analyzing in particular industrial critical systems [26].

2.2 Markov Reward Models

2.2.1 Continuous-Time Markov Chain

Continuous-Time Markov Chain (CTMC) is a stochastic process with discrete state space on continuous time domain, which can capture the dynamic behavior of system. The CTMC is convenient to represent the state transition of system such as normal and failure states, and thus is frequently used for reliability evaluation of system. In this paper, we consider the parameter sensitivity of CTMC in both steady-state and transient analysis.

Let $\{X(t);\ t > 0\}$ be a time-homogeneous CTMC with the denumerable state space $\mathcal{S} = \{1, 2, \ldots, n\}$ and $\boldsymbol{\pi}(t)$ is the state probability (row) vector whose i-th element is the probability $P(X(t) = i)$. According to the fundamental CTMC analysis, we have

$$\frac{d}{dt}\boldsymbol{\pi}(t) = \boldsymbol{\pi}(t)\boldsymbol{Q}, \tag{2.1}$$

where $\boldsymbol{Q} = [q_{ij}]$ is an n-by-n square matrix called the infinitesimal generator. The (i, j)-entry of $\boldsymbol{Q}$ means the transition rate from state i to state j. Also, when $\mathbf{1}$ and $\mathbf{0}$ are column vectors where all the elements are 1 and 0, respectively, the diagonal entries of $\boldsymbol{Q}$ are given by the negative values such that $\boldsymbol{Q}\mathbf{1} = \mathbf{0}$, namely, $q_{ii} = -\sum_{j \neq i} q_{ij}$.

In the transient analysis of CTMC, we focus on the probability vector $\boldsymbol{\pi}(t)$ at arbitrary time under a given initial probability $\boldsymbol{\pi}(0) = \boldsymbol{\pi}_0$. In other words, the transient analysis is to solve the initial value problem of the ordinary differential equation (ODE) (2.1). In general, by using the matrix exponential, the transient state probability vector can also be expressed by

$$\boldsymbol{\pi}(t) = \boldsymbol{\pi}_0 \exp(\boldsymbol{Q}t), \tag{2.2}$$

where the matrix exponential is defined by $\exp(\boldsymbol{A}) = \boldsymbol{I} + \boldsymbol{A} + \boldsymbol{A}^2/2! + \cdots$ and $\boldsymbol{I}$ is an identity matrix. The uniformization is well known as one of the most effective methods to solve the transient state probability vector [27].

On the other hand, the steady-state analysis of CTMC is to derive the steady-state probability vector;

$$\boldsymbol{\pi}_{ss}\boldsymbol{Q} = \boldsymbol{0}, \quad \boldsymbol{\pi}_{ss}\boldsymbol{1} = 1. \tag{2.3}$$

Intuitively, the steady-state probability vector $\boldsymbol{\pi}_{ss}$ corresponds to the state probability vector when $t \to \infty$. The steady-state probability vector can be obtained by solving the linear equation with GTH algorithm [28], Gauss-Seidel and SOR.

2.2.2 Markov Reward Model

Markov reward models (MRMs) are a model-based approach for evaluating the system performance. In general, The MRM is defined by a CTMC and a reward function which maps the finite state space $\mathcal{S}$ to a real value. Let ρ be a reward function in an MRM. Then a reward process is given by $Y(t) = \rho(X(t))$ for the underlying CTMC process $X(t)$.

In fact, several kinds of reward functions have been discussed in the past literatures [29, 46]. Since this paper deals with reliability and availability measures, we consider instantaneous reward functions in MRM. Let $\boldsymbol{r}$ be a column vector that maps each of CTMC state $i \in \mathcal{S}$ to a corresponding real-valued reward r_i at time instance. Thus the expected instantaneous reward at time t is given by

$$\mathrm{E}[Y(t)] = \boldsymbol{\pi}(t)\boldsymbol{r}. \tag{2.4}$$

Likewise, the expected instantaneous reward in the steady state is calculated by $\boldsymbol{\pi}_{ss}\boldsymbol{r}$.

2.3 Sensitivity Analysis

Sensitivity analysis is a method to estimate the magnitude of deviations of performance indices when system configuration changes such as the change in the component failure rate, thereby determining the factors that are most influential on model output measure. Thus the sensitivity analysis is effective to detect the critical component in the system. There are two kinds of sensitivity analysis: non-parametric and parametric approaches. Non-parametric approach is a method of evaluating sensitivity by statistical techniques, without any models. Typical examples are FAST and Sobol' methods [30]. Parametric approach is a method which examines the variations of output with respect to changes of the input parameters under given models. Generally, the parametric sensitivity is considered, which is defined by the first derivatives of performance indices with respect to model parameters. By using parametric sensitivity analysis, we can identify the model parameters that mostly affect the quantitative reliability measures. The parametric sensitivity can also be applied to optimizing system performance by combining the mathematical programming as well as the evaluation of effects on model parameters.

2.3.1 Parametric Sensitivity of MRMs

The sensitivity analysis is to estimate the magnitude of deviations of performance indices when some parameters change. In particular, the parametric sensitivity is the first or more derivatives of performance indices with respect to model parameters. The parametric sensitivity can also be applied to optimizing system performance by combing the mathematical programming as well as the evaluation of effects on parameters. In this section, we introduce the parametric sensitivity of general MRMs with instantaneous rewards. Similar to MRM analysis, the parametric sensitivity analysis is also divided into the steady-state and transient cases.

Let θ be a model parameter of MRM. The parametric sensitivity analysis starts with computing the following sensitivity functions:

$$\boldsymbol{s}_{ss}(\theta) = \tfrac{\partial \boldsymbol{\pi}_{ss}}{\partial \theta}, \tag{2.5}$$

$$\boldsymbol{s}(t,\theta) = \tfrac{\partial \boldsymbol{\pi}(t)}{\partial \theta}. \tag{2.6}$$

If these sensitivity functions are obtained, the sensitivity of performance index with instantaneous reward is given by

$$\tfrac{\partial}{\partial \theta}\boldsymbol{\pi}_{ss}\boldsymbol{r} = \boldsymbol{s}_{ss}(\theta)\boldsymbol{r} + \boldsymbol{\pi}_{ss}\tfrac{\partial}{\partial \theta}\boldsymbol{r}, \tag{2.7}$$

$$\tfrac{\partial}{\partial \theta}\mathrm{E}[Y(t)] = \boldsymbol{s}(t,\theta)\boldsymbol{r} + \boldsymbol{\pi}(t)\tfrac{\partial}{\partial \theta}\boldsymbol{r}. \tag{2.8}$$

Note that the above sensitivity functions of performance indices become simple when the reward vector is not sensitive to the parameter θ.

To obtain the sensitivity function in the case of steady-state probability vector, we take the first derivative of Eq. (2.3);

$$\boldsymbol{s}_{ss}(\theta)\boldsymbol{Q} + \boldsymbol{\pi}_{ss}\boldsymbol{S}(\theta) = \boldsymbol{0}, \quad \boldsymbol{s}_{ss}(\theta)\boldsymbol{1} = 0, \tag{2.9}$$

where $\boldsymbol{S}(\theta) = \partial \boldsymbol{Q}/\partial \theta$. If the steady-state probability vector $\boldsymbol{\pi}_{ss}$ is already given, the sensitivity function $\boldsymbol{s}_{ss}(\theta)$ can be solved as the linear equation.

In the transient case, from Eq. (2.1), the sensitivity function holds the following ODE:

$$\frac{d}{dt}\boldsymbol{s}(t,\theta) = \boldsymbol{s}(t,\theta)\boldsymbol{Q} + \boldsymbol{\pi}(t)\boldsymbol{S}(\theta). \tag{2.10}$$

By integrating Eq. (2.1) into the above ODE, we have

$$\frac{d}{dt}\tilde{\boldsymbol{\pi}}(t,\theta) = \tilde{\boldsymbol{\pi}}(t,\theta)\tilde{\boldsymbol{Q}}(\theta), \tag{2.11}$$

where

$$\tilde{\boldsymbol{\pi}}(t,\theta) = \left(\boldsymbol{\pi}(t), \boldsymbol{s}(t,\theta)\right), \quad \tilde{\boldsymbol{Q}}(\theta) = \begin{pmatrix} \boldsymbol{Q} & \boldsymbol{S}(\theta) \\ & \boldsymbol{Q} \end{pmatrix}. \tag{2.12}$$

Since the diagonal elements of $\tilde{\boldsymbol{Q}}(\theta)$ are same as those of $\boldsymbol{Q}$, we can apply the uniformization (see Appendix A) to the following matrix exponential form:

$$\tilde{\boldsymbol{\pi}}(t,\theta) = \tilde{\boldsymbol{\pi}}(0,\theta)\exp\left(\tilde{\boldsymbol{Q}}(\theta)t\right). \tag{2.13}$$

In fact, it is not enough to investigate the system deviation by using parametric sensitivity analysis, thus component importance analysis is considered.

2.3.2 Component Importance Analysis

In the reliability engineering, the component importance analysis is more preferred than the parametric sensitivity analysis. The component importance analysis, called the component-wise sensitivity analysis, is to estimate the first derivatives of reliability measures of system with respect to reliability measures of components, in other words, the component importance analysis measures the effect on system reliability of component reliabilities. Thus the component importance analysis can detect the critical components from the reliability point of view directly, and can be used to the system design.

Nevertheless, except for some of specific cases such as the independent components systems with explicit structure function, it is difficult to obtain the sensitivities of component reliability on system reliability analytically.

The component importance analysis, as well as the parametric sensitivity, is widely used sensitivity approach to detect the design sensitivity of system. In addition, some extensive sensitivity analyses are devoted to evaluate the environmental sensitivity such as the survival probabilities in fault-tolerant systems, indicating how expected survivability would change with varying model parameters.

CHAPTER 3

COMPONENT IMPORTANCE MEASURES FOR VIRTUALIZED SYSTEM

This chapter presents component importance analysis of virtualized system design. The component importance analysis is significant to develop a trusted system in its design phase. In this chapter, we discuss the importance of components from the viewpoint of availability. Specifically, based on the hybrid model (i.e., fault tree (FT) and continuous-time Markov chain (CTMC) models) for virtualized system in Kim et al. [18], we present a new method to evaluate the component importance and detect the critical components that contribute the most to the system availability. In numerical examples, we illustrate the quantitative importance of components and compare the availabilities of non-virtualized and virtualized system designs.

3.1 Model Description

3.1.1 Fault tree models

Consider the availability models presented in [18]. Concretely, the system provides two services such as Web and SQL servers to clients. When one of two services stops, the system fails (i.e., system failure occurs). Each of hosts is assumed to equip hardware and software components; CPU, memory

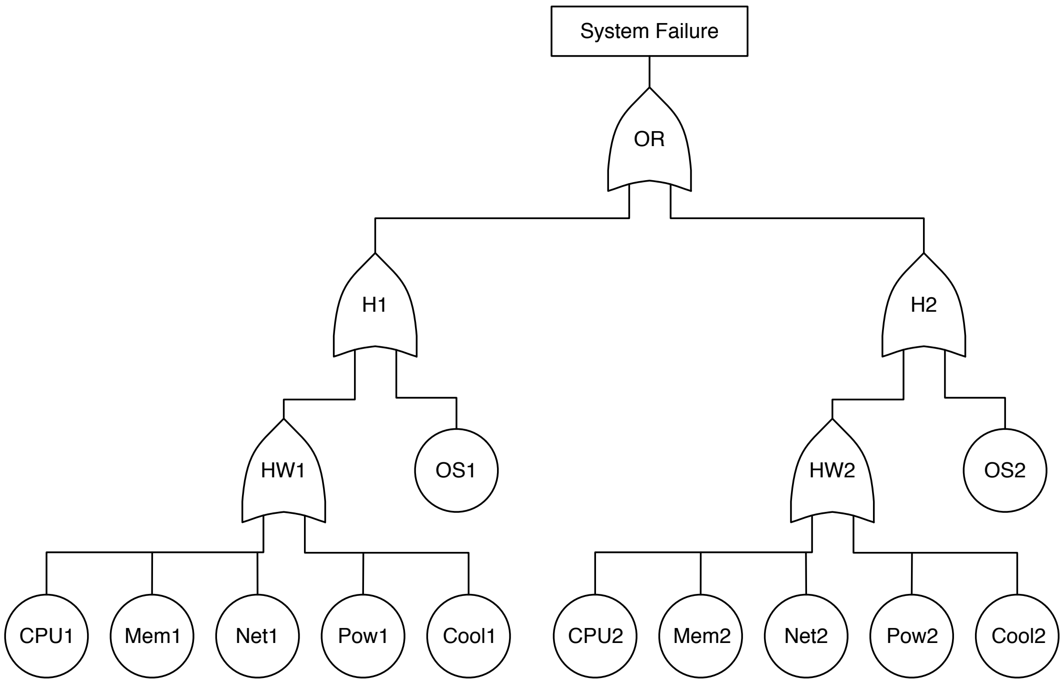

Figure 3.1 The FT diagram of non-virtualized system design.

(Mem), power subsystem (Pow), network device (Net), cooling subsystem (Cool) and operating system (OS). Here we present two system designs; non-virtualized and virtualized system designs.

In the non-virtualized system design, the system is composed of two hosts, and each of the hosts provides a specific service. Then the relationship between the system failure and all the component failures can be written as the FT diagram in Fig. 3.1. In the figure, the top event means the system failure and the leaf nodes correspond to the events that respective components are failed. The nodes, H1 (H2) and HW1 (HW2) represent the events that the host 1 (host 2) is failed and the hardware failure occurs in the host 1 (host 2), respectively. They are given by OR gates in the FT representation. Also, since the system failure occurs when one of two hosts is down, the top event is also given by an OR gate.

In the virtualized system design, the system is composed of two hosts, but the hosts are supposed to install the same virtual machine manager (VMM). The VMs run and provide the service on the VMM. One of the important features provided by the VMM is the live migration [4]. The live migration is a technique that can enhance the system availability. When a physical host is stopped, all the VMs running on the host can migrate to another physical

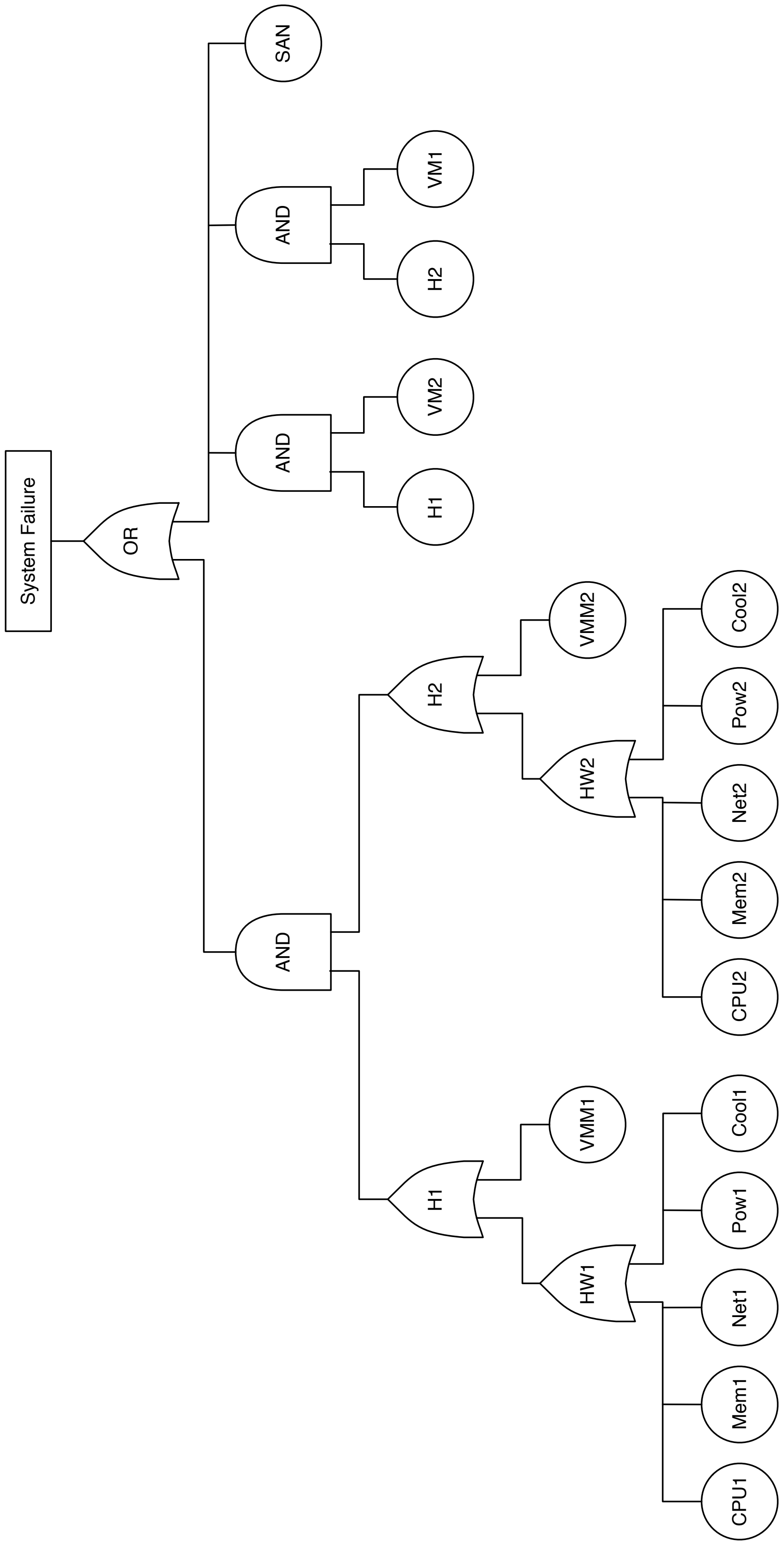

Figure 3.2 The FT diagram of virtualized system design.

host without the down time. In fact, most of the VMM products such as Xen, VMware and Hyper-V provide the live migration. However, in order to use the live migration, the two hosts are required to share a common storage area network (SAN). The SAN is a service to provide hard disk drives through a high-speed network using Fiber Channel or iSCSI technologies. Fig. 3.2 illustrates the FT in the virtualized system design. Although the hosts have the same structures as the FT in the non-virtualized system design, the failure of virtualized system is given by an AND gate because of the live migration. In addition, even if the VM is failed on one VMM, it can migrate to another VMM. Thus the VM failure (VM1 or VM2) is connected to the failure of another host (H2 or H1) with an AND gate. On the other hand, the failure of SAN causes the system failure directly, and therefore the top event is given by an OR gate connected to these events.

3.1.2 Continuous-time Markov chain models

In [18], Kim et al. defined the CTMC models to represent the behavior of hardware and software components. This section briefly introduces the CTMC models presented in [18].

Generally, in the availability modeling, the states of system can be classified into two sets: $\mathcal{U}$, the set of up (operational) states in which the system is available; and $\mathcal{D}$, the set of down (or failure) states in which the system is unavailable. Fig. 3.3 shows the 3-state CTMC model for the availability of CPU and Mem components proposed in [18]. In the figure, the states UP, DN and RP mean that the component is available, the component is failed, and the component is under repair, respectively. Hence the states DN and RP are classified into the down state in the availability model. Moreover, λ and μ denote failure and repair rates of the component. If the host equips 2-way CPUs, the failure rate is given by $\lambda = 2\lambda_{CPU}$ by using the failure rate of a single CPU because both processors are needed for the operation. Also, the transition from DN to RP corresponds to the event that a repair person is summoned and its mean time is given by $1/\alpha$ using the rate of summoning inherent in the component.

For the components, Pow and Net, Kim et al. [18] applied the 5-state availability model. This can be used for evaluating the component described as a 2-unit redundant (parallel) system. Fig. 3.4 presents the state transition diagram of the CTMC used for Pow and Net. In the figure, white and gray

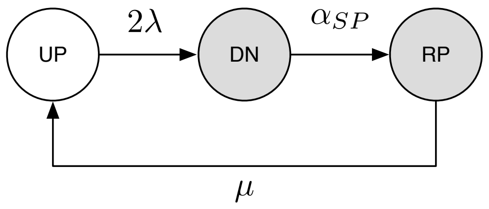

Figure 3.3 State transition diagram of the 3-state availability model.

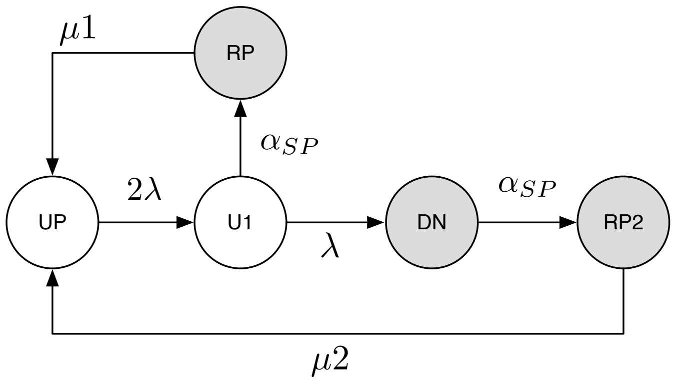

Figure 3.4 State transition diagram of the 5-state availability model.

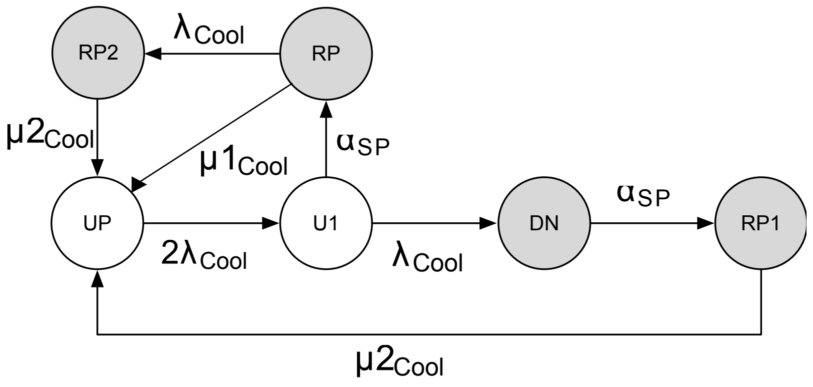

Figure 3.5 State transition diagram of the cooling system availability model.

nodes separately represent the up and down states in the availability model. The main difference from the 3-state availability model is to add the state that only one unit is failed, since the component failure is caused when both of two units are failed. Moreover, the model adds a repair state where two units are failed. The CTMC models for Cool and SAN are extended from the 5-state availability model. Concretely, in the CTMC for Cool (see Fig. 3.5), they added a transition from RP to RP2, namely, the Cool availability model allows the event occurrence that one unit fails while another unit is under re-

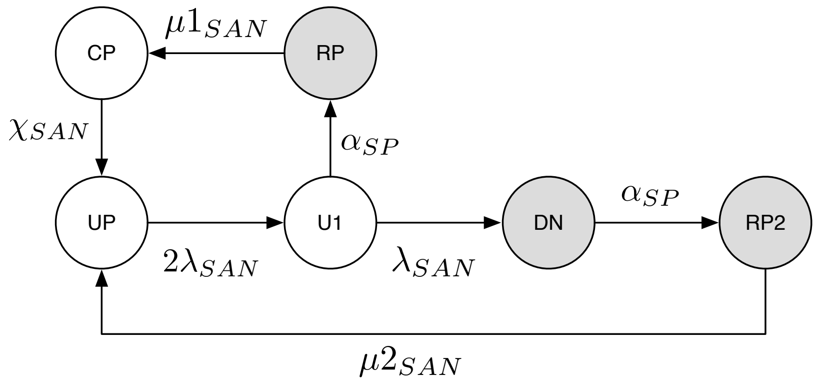

Figure 3.6 State transition diagram of the SAN availability model.

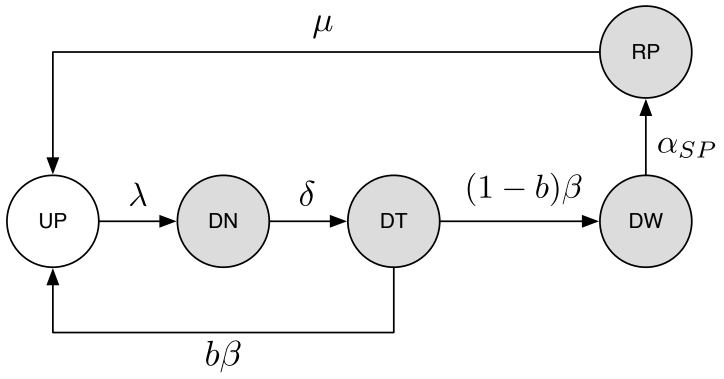

Figure 3.7 State transition diagram of the OS/VMM/VM availability model.

pair. In the CTMC for SAN (see Fig. 3.6), a state CP is put to the transition between the states UP and RP, which means the mirrored data is copied from a working disk unit to the repaired disk unit under RAID1 design. The transition rate from CP to UP is given by χ_{SAN}. Additionally, since the working disk unit may fail in the CP state, they added a transition from CP to RP2 with the failure rate of a disk unit λ_{SAN}.

The CTMC model for OS and VMM is given by Fig. 3.7. The state DT means that the failure is detected, since the software failure cannot be detected immediately. In [18], after the failure detection, the system takes an action to reboot OS or VMM witn mean time $1/\beta$. It is empirically known that most of transient failures in software can be recovered by the system reboot [31]. The state DN2 indicates that the failure is not recovered by a system reboot, i.e., the failure is caused by a kind of design faults, and thus b is the coverage of transient failures. In this case where the failure is caused by a design fault, a repair person is summoned.

In [18], based on the CTMC model in Fig. 3.7, they built the CTMC model for VM which takes account of the dynamic behavior of the live migration. Thus the CTMC model for VM was quite complicated so that the system failure in the virtualized system design cannot be represented by the FT. Since this paper describes the correlation between the failures of VM and host by the AND gate in the FT representation, the CTMC model simply becomes the same model as OS and VMM, i.e., the model of Fig. 3.7 can also represent the availability for VM.

Based on these CTMC models, the steady-state availability for a component x can be calculated as follows.

$$\begin{aligned} A_x &= \lim_{t\to\infty} \frac{\text{the cumulative available time during } [0,t)}{t} \\ &= \textstyle\sum_{k\in\mathcal{U}} \pi_k, \end{aligned} \tag{3.1}$$

where π_k is a steady-state probability of state k in the availability model and $\mathcal{U}$ is the set of up state. The steady-state probability π_k are computed by numerical methods such as power method and Gauss-Seidel method [32].

3.2 Availability Importance Analysis

3.2.1 Importance measures

According to the FT analysis, letting A_i be the steady-state availability of a component i, we have the following steady-state availabilities for a host in the non-virtualized and virtualized system designs:

$$A_H = A_{OS} \textstyle\prod_{i\in HW} A_i, \tag{3.2}$$

$$A_H = A_{VMM} \textstyle\prod_{i\in HW} A_i, \tag{3.3}$$

where HW is a set of $\{CPU, Mem, Net, Pow, Cool\}$. Then the system availability in the non-virtualized system design is given by

$$A_S = A_{H1}A_{H2}, \tag{3.4}$$

where A_{H1} and A_{H2} are the availabilities for the hosts 1 and 2 which can be obtained by A_H. In the case of the virtualized system design, since the events H1 and H2 are used twice in the FT, the system availability in the virtualized system design can be obtained

$$\begin{aligned} A_S = \; & 1 - \big(\overline{A}_{H1}\overline{A}_{H2} + \overline{A}_{H1}\overline{A}_{VM2} + \overline{A}_{H2}\overline{A}_{VM1} \\ & - \overline{A}_{H1}\overline{A}_{H2}(\overline{A}_{VM2} + \overline{A}_{VM1})\big)(1 - A_{SAN}), \end{aligned} \tag{3.5}$$

where $\overline{A}_i = 1 - A_i$.

In [33], Cassady et al. proposed the importance measures of components in terms of availability. They assumed the FT model with the events that are described by the 2-state availability model. The 2-state availability is a CTMC model with only two states; up and down. In such modeling, Cassady et al. [33] defined two importance measures as the derivatives of the system availability:

$$I_{\lambda,i} = \frac{1}{A_S} \left| \frac{\partial A_S}{\partial \lambda_i} \right|, \tag{3.6}$$

$$I_{\mu,i} = \frac{1}{A_S} \left| \frac{\partial A_S}{\partial \mu_i} \right|, \tag{3.7}$$

where λ_i and μ_i are the failure and repair rates of the component i, i.e., the transition rates from up to down and from down to up in the 2-state availability model, respectively. These measures come form the idea behind the Birnbaum measure [34].

In this chapter, since we do not treat the 2-state availability model to represent the component availability, the importance measures proposed in [33] cannot directly be applied to evaluating the non-virtualized and virtualized system designs. Thus we propose a preprocessing based on the aggregation of CTMC-based availability model [35] before applying the availability importance measures.

The aggregation is a technique to reduce CTMC-based availability models to the 2-state availability model which has the same availability as the original model. As mentioned before, the states of CTMC-based availability models can be classified into up and down groups. The aggregation technique converts the up and down groups to the up and down states of the 2-state availability model. The essential problem of the aggregation is to find the transition rates; failure and repair rates that ensure the steady-state probability of the up (down) group in the original model equals that of the up (down) state in the 2-state model. From the argument of CTMC, such failure and repair rates can be computed as follows.

$$\tilde{\lambda} = \frac{\sum_{(i,j)\in\mathcal{U}\times\mathcal{D}} \pi_i t_{i,j}}{\sum_{i\in\mathcal{U}} \pi_i}, \tag{3.8}$$

$$\tilde{\mu} = \frac{\sum_{(i,j)\in\mathcal{D}\times\mathcal{U}} \pi_i t_{i,j}}{\sum_{i\in\mathcal{D}} \pi_i}, \tag{3.9}$$

where $\mathcal{U}$ and $\mathcal{D}$ are sets of states belonging to the up and down groups, respectively. Then the set $\mathcal{U} \times \mathcal{D}$ indicates the transitions from up to down in the original model. Also, $t_{i,j}$ denotes the transition rate from i to j in the original model. For simplification, $t_{i,j} = 0$ if there is no transition from i to j. The calculated failure and repair rates $\tilde{\lambda}$ and $\tilde{\mu}$ in the 2-state availability model are called the equivalent failure and repair rates [35].

By applying the aggregation to the component availability models as pre-processing, the availability importance measures of the component i can be rewritten by

$$I_{\tilde{\lambda},i} = \frac{1}{A_S} \left| \frac{\partial A_S}{\partial \tilde{\lambda}_i} \right|, \tag{3.10}$$

$$I_{\tilde{\mu},i} = \frac{1}{A_S} \left| \frac{\partial A_S}{\partial \tilde{\mu}_i} \right|, \tag{3.11}$$

where $\tilde{\lambda}_i$ and $\tilde{\mu}_i$ are the equivalent failure and repair rates of the component i.

3.3 Numerical Illustration

This section illustrates the quantitative importance analysis of components based on the non-virtualized and virtualized system designs. Table 3.1 presents MTTF (mean time to failure) and MTTR (mean time to repair) for the components which are used in [18]. Also, other model parameters are given in Table 3.2.

Based on these parameters, we first compute the equivalent failure and repair rates for all the components when the original availability models are reduced to the 2-state availability models. Table 3.3 shows the equivalent failure and repair rates and the component availabilities. From the table, it is found that the availabilities of hardware units are relatively high, compared to the availabilities of software components. In particular, the availability of SAN is quite high.

From the component availabilities, we compute the system availabilities by using Eqs. (3.2)–(3.5). Table 3.4 presents the availability of a hardware unit, the availability of a host and the system availabilities in the non-virtualized and virtualized system designs. Although there is no remarkable difference of the host availabilities in the non-virtualized and virtualized system, the system availability of the virtualized system is further improved from the system availability of the non-virtualized system. This implies that the live migration is considerably effective to enhance the system availability.

Next we derive the importance measures of component in both non-virtualized and virtualized system designs. Since the equivalent failure and repair rates

Table 3.1 MTTF/MTTR of components.

Params	Description	Value (hours)
$1/\lambda_{CPU}$	MTTF of CPU	2,500,000
$1/\lambda_{Mem}$	MTTF of Mem	480,000
$1/\lambda_{Pow}$	MTTF of Pow	670,000
$1/\lambda_{Net}$	MTTF of Net	120,000
$1/\lambda_{Cool}$	MTTF of Cool	3,100,000
$1/\lambda_{SAN}$	MTTF of SAN	20,000,000
$1/\lambda_{OS}$	MTTF of OS	1440
$1/\lambda_{VMM}$	MTTF of VMM	2880
$1/\lambda_{VM}$	MTTF of VM	2880
$1/\mu_{CPU}$	MTTR of CPU	0.5
$1/\mu_{Mem}$	MTTR of Mem	0.5
$1/\mu 1_{Pow}$	MTTR of one power module	0.5
$1/\mu 2_{Pow}$	MTTR of two power modules	1
$1/\mu 1_{Net}$	MTTR of one network device	0.5
$1/\mu 2_{Net}$	MTTR of two network devices	1
$1/\mu 1_{Cool}$	MTTR of one cooler module	0.5
$1/\mu 2_{Cool}$	MTTR of two cooler modules	1
$1/\mu 1_{SAN}$	MTTR of one disk unit	0.5
$1/\mu 2_{SAN}$	MTTR of two disk units	1
$1/\mu_{OS}$	MTTR of OS	1
$1/\mu_{VMM}$	MTTR of VMM	1
$1/\mu_{VM}$	MTTR of VM	0.5

Table 3.2 Other model parameters.

Params	Description	Value
$1/\alpha_{SP}$	mean time to repair person summoned	30 minutes
$1/\chi_{SAN}$	mean time to copy data	20 minutes
$1/\delta_{OS}$	mean time for OS failure detection	30 seconds
$1/\delta_{VMM}$	mean time for VMM failure detection	30 seconds
$1/\delta_{VM}$	mean time for VM failure detection	30 seconds
$1/\beta_{OS}$	mean time to reboot OS	10 minutes
$1/\beta_{VMM}$	mean time to reboot VMM	10 minutes
$1/\beta_{VM}$	mean time to reboot VM	5 minutes
b_{OS}	coverage factor for OS reboot	0.9
b_{VMM}	coverage factor for VMM reboot	0.9
b_{VM}	coverage factor for VM reboot	0.95

Table 3.3 Equivalent failure and repair rates and component availabilities.

Component	$\tilde{\lambda}_i$	$\tilde{\mu}_i$	A_i
CPU	8.0000000e-7	1.0000000	0.99999920
Mem	8.3333333e-6	1.0000000	0.99999167
Net	1.6666528e-5	1.9999833	0.99999167
Pow	2.9850702e-6	1.9999970	0.99999851
Cool	6.4516108e-7	1.9999990	0.99999968
OS	6.9444444e-4	3.0769231	0.99977436
VMM	3.4722222e-4	3.0769231	0.99988717
VM	3.4722222e-4	7.0588235	0.99995081
SAN	9.9999992e-8	1.9999999	0.99999995

Table 3.4 Availabilities of hardware units, host and system.

System	Availability
HW1 and HW2	0.99998072
H1 and H2 in non-virtualized system	0.99975508
H1 and H2 in virtualized system	0.99986789
System availability in Non-virtualized system	0.99951022
System availability in virtualized system	0.99999992

have already been computed as shown in Table 3.3, we substitute them into Eqs. (3.10) and (3.11) after differentiating them analytically. Tables 3.5 and 3.6 show the importance measures of components in terms of the system availability for non-virtualized and virtualized systems, respectively. Since the components of the hosts 1 and 2 are same in both non-virtualized and virtualized system design, the importance measures of same components in the hosts 1 and 2 are identical. Thus the tables present the importance measures of components in a host.

In the result of non-virtualized system design, it can be found that the importance measures of CPU and Mem are higher than those of the other components in terms of failure rates, namely, $I_{\tilde{\lambda},i}$. The importance measure regarding failure rates indicates that how much the system availability decreases as the component failure rate increases. In Table 3.3, we find the repair rates of CPU and Mem are not so high, and the failures of CPU and Mem cause long down time. Hence their importance measures are higher than the others. The similar tendency can be observed in the case of OS. That is, although the failure rate of OS is higher, the importance measure $I_{\tilde{\lambda},OS}$ of OS is the smallest among those of others, because the repair rate of OS is high. Moreover, by comparing between the importance measures with respect to failure and repair rates, it can be seen that the importance measures with respect to failure rate is higher than the importance measures with respect to repair rate. Since the importance measures are defined by the first derivative of failure or repair rate, we can conclude that the improvement of failure rate is more important to enhance the system availability in the case of non-virtualized system design.

In the case of virtualized system design, both the importance measures with respect to failure and repair rates are changed from those of non-virtualized system design. Specifically, all the importance measures become smaller than those of non-virtualized system design. This implies that the improvement of failure and repair rates is not effective on the system availability, compared to the non-virtualized system design. This is caused by the fact that the system availability of virtualized system design is quite high. The system availability of virtualized system design is sufficiently high. Thus this implies that it spends much cost to require more higher system availability. On the other hand, there is room to improve the failure and repair rates in SAN. In other words, SAN is a bottleneck of availability, though its availability seems to be high. Also, from the result, we find that the importance measures of VM and VMM are not high. In fact, VM and VMM can be migrated when a failure occurs. Therefore, compared to SAN, VM and VMM are not critical components.

Table 3.5 Component importance measures in the non-virtualized system design.

Component	$I_{\tilde{\lambda},i}$	$I_{\tilde{\mu},i}$
CPU	0.99999920	7.9999936e-7
Mem	0.99999167	8.3332639e-6
Net	0.50000000	4.1666667e-6
Pow	0.50000000	7.4626866e-7
Cool	0.50000008	1.6129037e-7
OS	0.32492667	7.3334143e-5

Table 3.6 Component importance measures in the virtualized system design.

Component	$I_{\tilde{\lambda},i}$	$I_{\tilde{\mu},i}$
CPU	1.8126415e-4	1.4501132e-10
Mem	1.8126278e-4	1.5105232e-09
Net	9.0632147e-5	7.5526790e-10
Pow	9.0632147e-5	1.3527186e-10
Cool	9.0632162e-5	2.9236186e-11
VMM	5.8904249e-5	6.6471808e-09
VM	1.8711815e-5	9.2043069e-10
SAN	0.50000000	2.4999999e-08

3.4 Conclusion

In this chapter, we have revisited the FT and CTMC models for non-virtualized and virtualized system design in [18], and have proposed the generalized method of importance analysis of components from the viewpoint of availability. Concretely, our method is based on both the aggregation techniques of CTMC-based availability models [35] and the importance measures of components with respect to failure and repair rates [33]. The proposed method can be applied to any types of virtualized system design without changes. In numerical examples, based on model parameters in [18], we have exhibited the importance analysis of components and detected the critical components in both non-virtualized and virtualized system designs. Although the presented component analysis is simple, it is quite helpful in the system design phase to ensure the system availability.

CHAPTER 4

IMPORTANCE MEASURES FOR VIRTUALIZED SYSTEM WITH LIVE MIGRATION

This chapter is an extension work of Chapter 3, which presents component importance analysis for virtualized system with live migration. In Chapter 3, we have developed a generalized method to evaluate the importance of components for hybrid models. However, the hybrid model had a limitation for the model expression. For example, when two or more components have interactions between them, the structure function cannot always be explicitly expressed. In such cases, we cannot use the hybrid model. Instead of using the hybrid model, we should use a CTMC describing whole the system behavior. In the component analysis of virtualized system, the behavior of live migration is this case. In fact, Matos et al. [19] presented only a CTMC for the live migration. Since the structure function cannot be obtained from the CTMC, we cannot also apply the component importance analysis in Chapter 3 to the live migration model. In this chapter, we introduce the state-of-the-art Markov-based component-wise sensitivity analysis and apply it to the CTMC-based live migration model to reveal the component importance in the context of live migration without using structure function.

4.1 Availability Importance Analysis for Hybrid Model

4.1.1 Fault tree model

Consider the FT model for the virtualized system shown as Fig. 3.2 in Chapter 3. In general, the virtualized system is composed of three elements: physical server, VM, and SAN. For instance, a service such as Web or SQL is provided by a VM running on a physical server. The system under consideration has two physical hosts; H1 and H2, two VMs; VM1 and VM2, and one SAN. Two VMs provide different services, and H1 and H2 have the same virtualization platform. That is, if a physical host fails, the VM running on the host can be migrated to another physical host. This is called the live migration, which is one of the important functions to ensure the high availability in the virtualized platform.

4.1.2 Continuous-time Markov chain models

In this section, we revisit the CTMC models for components and describe them in detail. Fig. 4.1 shows the 3-state CTMC availability models of CPU and Mem components proposed in [18]. In the figure, the states UP, DN and RP mean that the component is available, the component is failed, and the component is under repair, respectively. Hence the states DN and RP are classified into $\mathcal{D}$ set in the availability model. Moreover, λ and μ denote failure and repair rates of the component. For example, if the host equips 2-way CPUs, the failure rate is given by $\lambda = 2\lambda_{CPU}$ by using the failure rate of a single CPU because both processors are needed for the operation. Also, the transition from DN to RP corresponds to the event that a repair person is summoned and its mean time is given by $1/\alpha$ using the rate of summoning inherent in the component.

As seen in Fig. 4.2, Kim et al. [18] applied the 5-state availability model to describe the dynamic behaviors of components Pow and Net which are described as 2-unit redundant (parallel) subsystems. In the figure, white and gray nodes represent up and down states respectively. The main difference from the 3-state availability model is to add the state U1 representing that only one unit is failed, since the component failure is caused when both of two units are failed. Moreover, the model adds a repair state RP2 where two units are failed. For the components, Cool and SAN, the CTMC models are extended from the 5-state availability model. Concretely, in the CTMC for

Cool as shown in Fig. 4.3, they added a transition from RP to RP2, namely, the Cool availability model allows the event occurrence that one unit fails while another unit is under repair. In the CTMC for SAN as shown in Fig. 4.4, a state CP is put to the transition between the states UP and RP, which means the mirrored data is copied from a working disk unit to the repaired disk unit under RAID1 design. The transition rate from CP to UP is given by χ_{SAN}. Additionally, since the working disk unit may fail in the CP state, they added a transition from CP to RP2 with the failure rate of a disk unit λ_{SAN}.

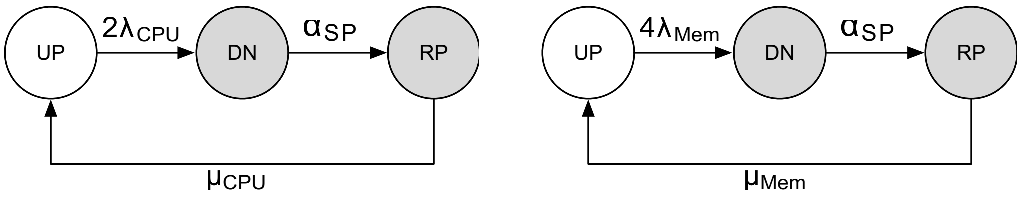

Figure 4.1 State transition diagram of the CPU and memory availability models.

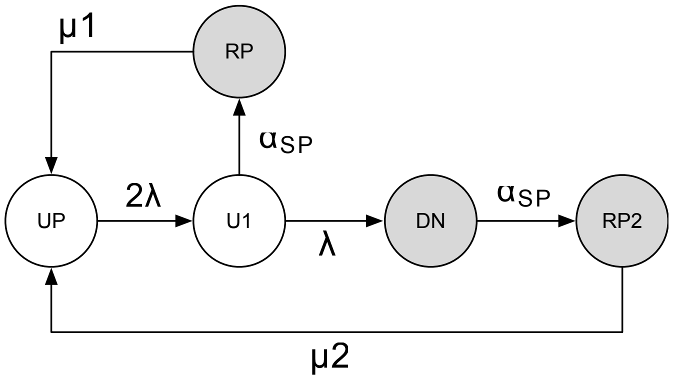

Figure 4.2 State transition diagram of the power (or network card) availability model.

The CTMC model for VMM is given by Fig. 4.5. As seen in this figure, since the software failure cannot be detected immediately, the state DT is added, which means the failure is detected. In [18], after the failure detection, the system takes an action to reboot VMM with mean time $1/\beta$. It is empirically known that most of transient failures in software can be recovered by the system reboot [31]. In this CTMC model, the reboot will be unsuccessful with probability $(1 - b)$. Hence the state DW indicates that the failure is not recovered by a failed system reboot, and a repair person is summoned.

In [18], based on the CTMC model in Fig. 4.5, they built the CTMC model for VM which takes account of the dynamic behaviors of the live migration.

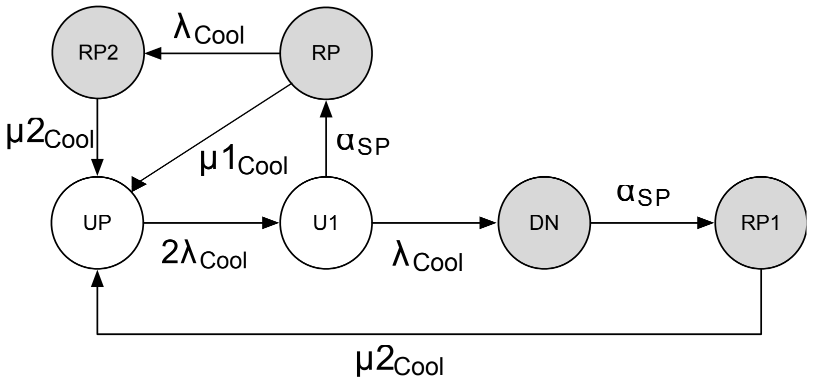

Figure 4.3 State transition diagram of the cooling system availability model.

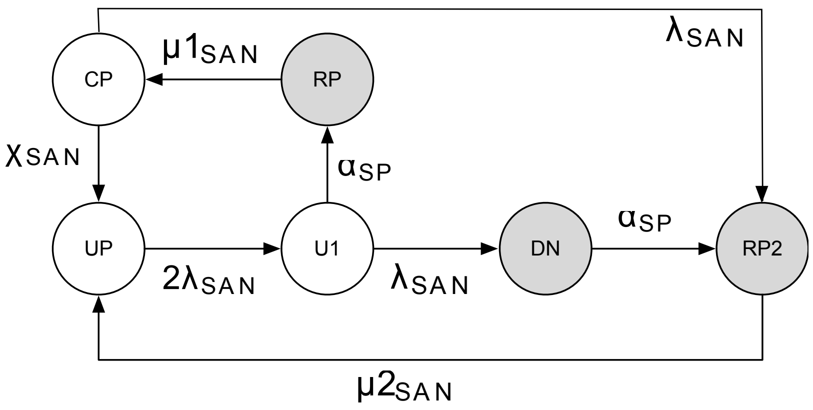

Figure 4.4 State transition diagram of the SAN availability model.

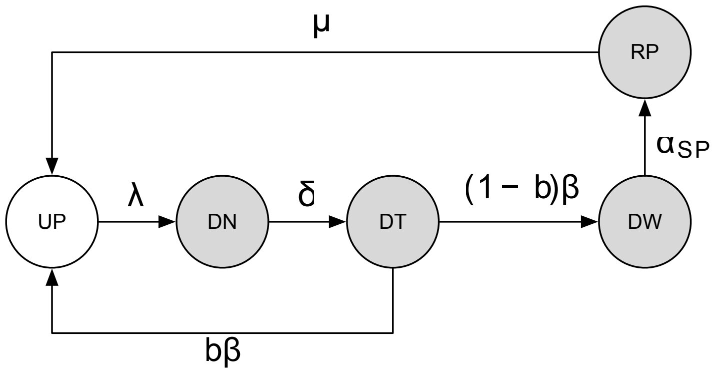

Figure 4.5 State transition diagram of the VMM/VM availability model.

Thus the CTMC model for VM was quite complicated so that the system failure in the virtualized system cannot be represented by the FT. Since this paper describes the correlation between the failures of VM and host by the AND gate in the FT representation, the CTMC model simply becomes the same model as VMM, i.e., the model of Fig. 4.5 can also represent the availability for VM.

4.1.3 Importance measures

We also consider the importance measures as those as in Chapter 3. That is,

$$I_{\tilde{\lambda},i} = \frac{1}{A_S}\left|\frac{\partial A_S}{\partial \tilde{\lambda}_i}\right|, \quad I_{\tilde{\mu},i} = \frac{1}{A_S}\left|\frac{\partial A_S}{\partial \tilde{\mu}_i}\right|, \tag{4.1}$$

where $\tilde{\lambda}_i$ and $\tilde{\mu}_i$ are the effective failure and repair rates of component i.

4.2 Component Importance for Live Migration

In the previous section, we have introduced the component importance for the structure function given by the FT model. The model considered the live migration as a static structure. However, since the live migration is essentially described by a dynamic behavior, the previous method cannot analyze how effect of components on the dynamic behaviors of live migration. Thus in this section, we consider the component importance on live migration from the viewpoint of dynamic behaviors, that is, we apply the component importance analysis for a CTMC representing the dynamic behaviors of live migration presented in [19].

4.2.1 Model description

Matos et al. [19] presented the CTMC for live migration in the virtualized system. This availability model does not consider the detailed behavior of hardware components (e.g., CPU, Mem, Pow) and the VMM, but only the components of VMs (VM1 and VM2), hosts (H1 and H2) and applications (App1 and App2).

Table 4.1 shows notations for the state of system which are based on the current conditions of components. Concretely, each state is indicated by six characters. The first character means the state of H1. The notations ‘U’, ’F’ and ‘D’ correspond to the conditions where H1 is up, H1 fails and the failure is detected, respectively. The second character represents the state of

Table 4.1 The states of system.

State	Description
UUXUUX	VM1 is running on H1, VM2 is running on H2.
FXXUUX	H1 is failed, VM1 is failed due to the failure of H1. VM2 is running on H2.
DXXUUR	H1 failure is detected, VM1 is restarting on H2.
DXXUUU	H1 is down, VM1 and VM2 are running on H2.
UXXUUU	H1 is up, VM1 and VM2 are running on H2.
UXXFXX	H1 is up, H2 is failed. VM1 and VM2 are failed due to the failure of H2.
URXDXX	H2 failure is detected. VM1 is restarting on H1.
DXXFXX	H1 is down, H2 is failed.
DXXDXX	H1 is down, H2 failure is detected.
DXXURX	H1 is down, H2 is up, VM2 is restarting on H2.
UXXURX	H1 is up, H2 is up, VM2 is restarting on H2.
UXXUUR	H1 is up, VM2 is running on H2. VM1 is restarting on H2.
UFaXUUX	App1 is failed, both VMs and Hosts are up.
UDaXUUX	App1 failure is detected.
UPaXUUX	App1 failure is not covered. Additional recovery step is started.
UFvXUUX	H1 is up, VM1 is failed, VM2 is running on H2.
UDvXUUX	VM1 failure is detected.
UPvXUUX	VM1 failure is not covered. Manual repair is started.

Table 4.2 Model parameters.

Params	Description
$1/\lambda_h$	Mean time to host failure
$1/\lambda_v$	Mean time to VM failure
$1/\lambda_a$	Mean time to Application failure
$1/\delta_h$	Mean time for host failure detection
$1/\delta_v$	Mean time for VM failure detection
$1/\delta_a$	Mean time for App failure detection
$1/m_v$	Mean time to migrate a VM
$1/r_v$	Mean time to restart a VM
$1/\mu_h$	Mean time to repair a host
$1/\mu_v$	Mean time to repair a VM
$1/\mu1_a$	Mean time to App first repair (covered case)
$1/\mu2_a$	Mean time to App second repair (not covered case)
c_v	coverage factor for VM repair
c_a	coverage factor for application repair

VM1 and its application (App1). When both are up, the character is given by ‘U’. If VM1 fails, it is ‘Fv’. When the failure is detected, the character becomes ‘Dv’. Also, when a manual repair is applied, the character is ‘Pv’. If App1 fails, it is ‘Fa’. When the failure of App1 is detected, the state of system is represented by ‘Da’. If App1 requires an additional repair in the case where the application restart cannot solve the problem, the character is given by ‘Pa’. Also, when VM1 and App1 are restarting, the state is given by ‘R’. If VM1 and App1 are not running on the H1, then the character is ‘X’. The third character represents whether or not VM2 and App2 are running on H1. If VM2 and App2 run on H1, the character is given by ‘U’. If they are restarting on H1, the character is ‘R’. Otherwise, if they are not running on H1, the character is ‘X’. The fourth through sixth characters represent the state of H2 in the same manner as the first through third characters. Fig. 4.6 shows the state transition diagram for live migration in the virtualized system which is described by the CTMC model in [19]. Also, Table 4.2 presents the parameters of the CTMC model. For example, $1/\lambda_h$ is MTTF of host H1 and H2, and then λ_h is a failure rate which is a transition rate in the CTMC.

4.2.2 Importance analysis

Dissimilar to the case of FT model, we do not know the structure function in the CTMC. We consider the component importance analysis by only using the parameter sensitivity analysis.

Let $\boldsymbol{Q}$ be the infinitesimal generator of CTMC described in Fig. 4.6. Then the steady-state probability vector $\boldsymbol{\pi}_{ss}$ is given by the linear equations;

$$\boldsymbol{\pi}_{ss}\boldsymbol{Q} = \mathbf{0}, \quad \boldsymbol{\pi}_{ss}\mathbf{1} = 1, \tag{4.2}$$

where $\mathbf{1}$ is a column vector whose elements are 1. Also we define the following vectors:

- $\boldsymbol{\xi}_{hi,i\in\{1,2\}}$: a 0-1 vector whose elements are 1 in the state where H1 or H2 is up.
- $\boldsymbol{\xi}_{vi,i\in\{1,2\}}$: a 0-1 vector whose elements are 1 in the state where VM1 or VM2 is up.
- $\boldsymbol{\xi}_{ai,i\in\{1,2\}}$: a 0-1 vector whose elements are 1 in the state where App1 or App2 is up.

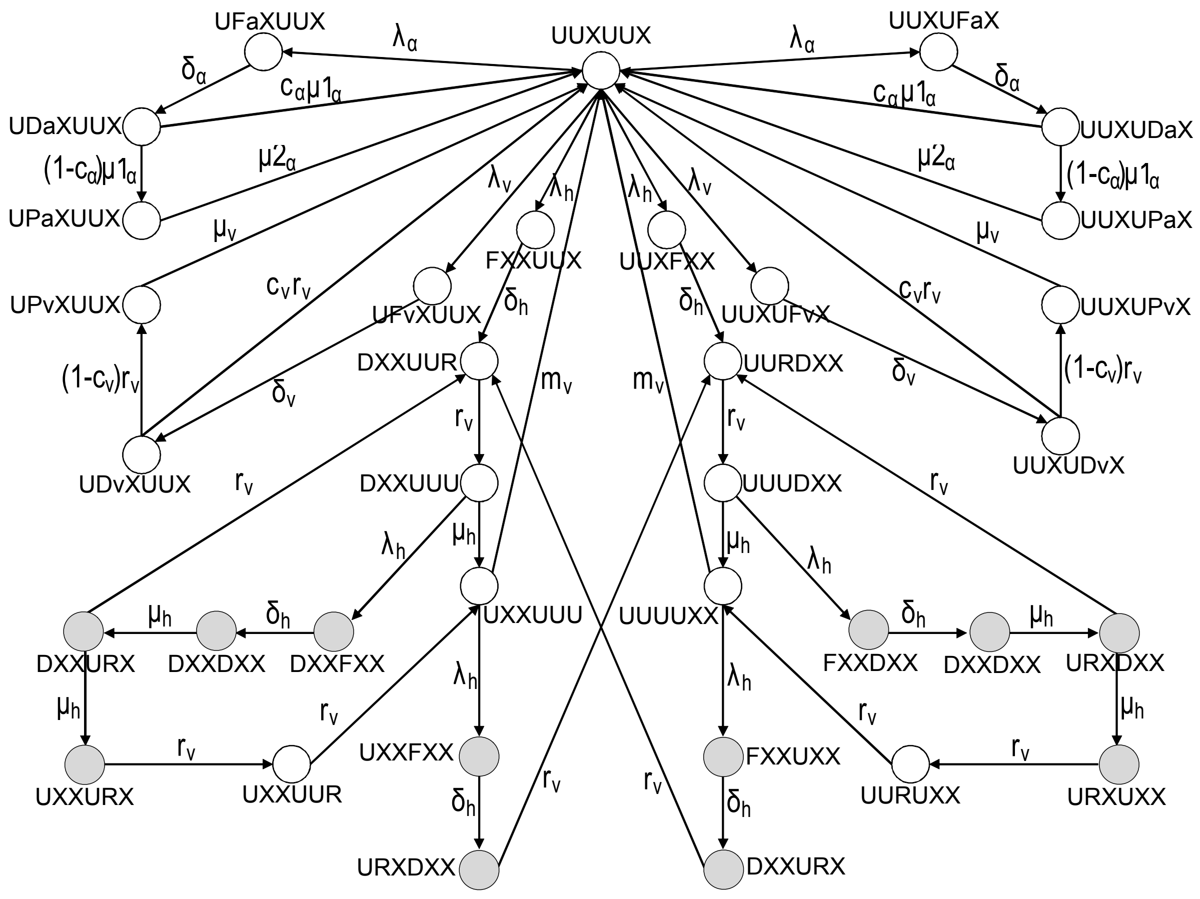

Figure 4.6 CTMC availability model for live migration.

- $\boldsymbol{\xi}_{sys}$: a 0-1 vector whose elements are 1 in the state where the system is up.

Then the component availability is given by a inner product of $\boldsymbol{\pi}_{ss}$ and $\boldsymbol{\xi}_{\cdot}$; for example, the component availability of H1 becomes

$$A_{h1} = \boldsymbol{\pi}_{ss}\boldsymbol{\xi}_{h1}. \tag{4.3}$$

On the other hand, the system availability can be obtained by

$$A_S = \boldsymbol{\pi}_{ss}\boldsymbol{\xi}_{sys}. \tag{4.4}$$

Similar to the case of FT model, we define the importance measures of component i as follows.

$$I_{\tilde{\lambda},i} = \frac{1}{A_S}\left|\frac{\partial A_S}{\partial \tilde{\lambda}_i}\right|, \quad I_{\tilde{\mu},i} = \frac{1}{A_S}\left|\frac{\partial A_S}{\partial \tilde{\mu}_i}\right|, \tag{4.5}$$

where $\tilde{\lambda}_i$ and $\tilde{\mu}_i$ are the effective failure and repair rates of component i. They can be computed by the aggregation technique introduced in Section 4.1.3. Also, we have

$$I_{\tilde{\lambda},i} = \frac{1}{A_S}\left|\frac{\partial A_S}{\partial \tilde{\lambda}_i}\right| = \frac{1}{A_S}\left|\frac{\partial A_S}{\partial A_i}\right|\left|\frac{\partial A_i}{\partial \tilde{\lambda}_i}\right| = \frac{1}{A_S}\left|\frac{\partial A_S}{\partial A_i}\right|\frac{\tilde{\mu}_i}{(\tilde{\lambda}_i+\tilde{\mu}_i)^2}. \tag{4.6}$$

Similarly, the importance measure with respect to repair rate is given by

$$I_{\tilde{\mu},i} = \frac{1}{A_S}\left|\frac{\partial A_S}{\partial A_i}\right|\frac{\tilde{\lambda}_i}{(\tilde{\lambda}_i+\tilde{\mu}_i)^2}. \tag{4.7}$$

Thus the problem is to estimate the sensitivity $\partial A_S/\partial A_i$ without the structure function.

To estimate the sensitivities for all the component availabilities, we consider the sensitivities of system and component availabilities with respect to model parameters. Suppose that $\theta_1, \ldots, \theta_m$ are model parameters of the underlying CTMC. Here we define a matrix $\boldsymbol{J}$ and a column vector $\boldsymbol{z}$ whose elements are the sensitivities for all the component availabilities and the system availability with respect to the model parameters, i.e.,

$$\boldsymbol{J} = \begin{pmatrix} \frac{\partial A_1}{\partial \theta_1} & \frac{\partial A_2}{\partial \theta_1} & \cdots & \frac{\partial A_n}{\partial \theta_1} \\ \frac{\partial A_1}{\partial \theta_2} & \frac{\partial A_2}{\partial \theta_2} & \cdots & \frac{\partial A_n}{\partial \theta_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial A_1}{\partial \theta_m} & \frac{\partial A_2}{\partial \theta_m} & \cdots & \frac{\partial A_n}{\partial \theta_m} \end{pmatrix}, \quad \boldsymbol{z} = \begin{pmatrix} \frac{\partial A_S}{\partial \theta_1} \\ \frac{\partial A_S}{\partial \theta_2} \\ \vdots \\ \frac{\partial A_S}{\partial \theta_m} \end{pmatrix}, \tag{4.8}$$

where $A_1, \ldots, A_n$ represent component availabilities for all the components. These sensitivities can be obtained by solving the following linear equations:

$$\boldsymbol{s}(\theta_j) = \frac{\partial}{\partial \theta_j}\boldsymbol{\pi}_{ss}, \quad \boldsymbol{s}(\theta_j)\boldsymbol{Q} = -\boldsymbol{\pi}_{ss}\frac{\partial}{\partial \theta_j}\boldsymbol{Q}, \quad \boldsymbol{s}(\theta_j)\boldsymbol{1} = 0. \tag{4.9}$$

By using the vector $\boldsymbol{s}(\theta_j)$, the sensitivities are given by

$$\frac{\partial A_i}{\partial \theta_j} = \boldsymbol{s}(\theta_j)\boldsymbol{\xi}_i, \quad \frac{\partial A_S}{\partial \theta_j} = \boldsymbol{s}(\theta_j)\boldsymbol{\xi}_{sys} \tag{4.10}$$

According to [36], the estimates of $\partial A_S/\partial A_i$ can be obtained by (see Appendix B)

$$\begin{pmatrix} \frac{\partial A_S}{\partial A_1} & \frac{\partial A_S}{\partial A_2} & \cdots & \frac{\partial A_S}{\partial A_n} \end{pmatrix}^T = (\boldsymbol{J}^T\boldsymbol{J})^{-1}\boldsymbol{J}^T\boldsymbol{z}, \tag{4.11}$$

where T is the transpose operator. By substituting the estimates of the sensitivities into Eqs. (4.6) and (4.7), we have the component importance measures for live migration.

4.3 Numerical Illustration

4.3.1 Hybrid models

In this section, we illustrate the quantitative component importance analysis of hybrid model for virtualized system. Table 4.3 presents the parameters of the CTMC models for all components. For example, $1/\lambda_{CPU}$ is mean time for CPU failure, and $1/\mu_{Mem}$ is mean time to repair one memory (i.e., MTTR of one memory). Also we give other model parameters in Table 4.4.

Table 4.3 MTTF/MTTR of components.

Params	Description	Value (hours)
$1/\lambda_{CPU}$	MTTF of CPU	2,500,000
$1/\lambda_{Mem}$	MTTF of Mem	480,000
$1/\lambda_{Pow}$	MTTF of Pow	670,000
$1/\lambda_{Net}$	MTTF of Net	120,000
$1/\lambda_{Cool}$	MTTF of Cool	3,100,000
$1/\lambda_{SAN}$	MTTF of SAN	20,000,000
$1/\lambda_{VMM}$	MTTF of VMM	2880
$1/\lambda_{VM}$	MTTF of VM	2880
$1/\mu_{CPU}$	MTTR of CPU	0.5
$1/\mu_{Mem}$	MTTR of Mem	0.5
$1/\mu1_{Pow}$	MTTR of one power module	0.5
$1/\mu2_{Pow}$	MTTR of two power modules	1
$1/\mu1_{Net}$	MTTR of one network device	0.5
$1/\mu2_{Net}$	MTTR of two network devices	1
$1/\mu1_{Cool}$	MTTR of one cooler module	0.5
$1/\mu2_{Cool}$	MTTR of two cooler modules	1
$1/\mu1_{SAN}$	MTTR of one disk unit	0.5
$1/\mu2_{SAN}$	MTTR of two disk units	1
$1/\mu_{VMM}$	MTTR of VMM	1
$1/\mu_{VM}$	MTTR of VM	0.5

Table 4.4 Other model parameters.

Params	Description	Value
$1/\alpha_{SP}$	mean time to repair person summoned	30 minutes
$1/\chi_{SAN}$	mean time to copy data	20 minutes
$1/\delta_{VMM}$	mean time for VMM failure detection	30 seconds
$1/\delta_{VM}$	mean time for VM failure detection	30 seconds
$1/\beta_{VMM}$	mean time to reboot VMM	10 minutes
$1/\beta_{VM}$	mean time to reboot VM	5 minutes
b_{VMM}	coverage factor for VMM reboot	0.9
b_{VM}	coverage factor for VM reboot	0.95

Table 4.5 Effective failure and repair rates and component availabilities.

Component	$\tilde{\lambda}_i$	$\tilde{\mu}_i$	A_i
CPU	8.0000000e-7	1.0000000	0.99999920
Mem	8.3333333e-6	1.0000000	0.99999167
Net	1.6666528e-5	1.9999833	0.99999167
Pow	2.9850702e-6	1.9999970	0.99999851
Cool	6.4516108e-7	1.9999990	0.99999968
VMM	3.4722222e-4	3.0769231	0.99988717
VM	3.4722222e-4	7.0588235	0.99995081
SAN	9.9999992e-8	1.9999999	0.99999995

Using the aggregation technique, we first transform the availability models for all components into the equivalent 2-state, 2-transition models, then compute the effective failure and repair rates for components based on the model parameters. We also compute the component availabilities, and these results are shown in Table 4.5. From this table, we can see the availabilities of hardware units are relatively high by the comparison to the availabilities of software components, especially for SAN, the availability is quite high.

We then compute the system availabilities based on the structure functions and the component availabilities. The availabilities of a hardware unit and a host, and the system availability are presented in Table 4.6. From this table, the sufficiently high availability of the virtualized system implies that the live migration is considerably effective to enhance the system availability.

Table 4.6 Availabilities of hardware units, host and system.

System	Availability
HW1 and HW2	0.99998072
H1 and H2	0.99986789
System availability	0.99999992

Next we derive the importance measures of components in the virtualized system by using Eq. (4.1), and the effective failure and repair rates shown in Table 4.5. The importance measures of components in terms of the system availability are shown in Table 4.7. Note that this table presents the importance measures of components only in a host, because the components of the host 1 and 2 are assumed to be the same in the system design, and the importance measures of same components in the host 1 and 2 are identical.

Table 4.7 shows that the importance measure with respect to failure rate is higher than that with respect to repair rate for any component. The importance measure regarding failure rate, $I_{\tilde{\lambda},i}$, indicates the relative improvement in system availability resulting from a decrease to the component failure rate. Similarly, the importance measure regarding repair rate, $I_{\tilde{\mu},i}$, indicates the relative improvement in system availability resulting from an increase to the component repair rate. Thus, to improve the system availability, the more efficient way is to decrease the failure rates of components. Also, as seen in this table, it is easy to find that the importance measures of SAN are much higher than those of the other components, especially the importance measure with respect to failure rate, $I_{\tilde{\lambda},SAN}$. The highest importance of SAN indicates that the improvement of failure rate of SAN is the most efficient way to improve the system availability. In other words, SAN is a bottleneck of availability, though its availability seems to be high. Besides, from Table 4.5, we find the repair rates of CPU and Mem are not so high. This implies that the failures of CPU and Mem cause long down time. Hence their importance measures with respect to failure rate are relatively higher than the others except SAN. Moreover, we find that the importance measures of VM and VMM are not high in Table 4.7. This is caused by the fact that VM and VMM can be migrated when a failure of a host occurs. Therefore, VM and VMM are not critical components, compared to SAN.

Table 4.7 Component importance measures in the virtualized system.

Component	$I_{\tilde{\lambda},i}$	$I_{\tilde{\mu},i}$
CPU	1.8126415e-4	1.4501132e-10
Mem	1.8126278e-4	1.5105232e-09
Net	9.0632147e-5	7.5526790e-10
Pow	9.0632147e-5	1.3527186e-10
Cool	9.0632162e-5	2.9236186e-11
VMM	5.8904249e-5	6.6471808e-09
VM	1.8711815e-5	9.2043069e-10
SAN	0.5000000000	2.4999999e-08

4.3.2 Dynamic model for live migration

This section illustrates the quantitative component importance analysis of the CTMC for live migration in the virtualized system. Based on these parameters shown in Table 4.8, we first compute the availabilities for all components and system which are shown in Table 4.9. From this table, we find that the availability of VM is the highest among those of the other components because of the live migration.

Next we compute the effective failure and repair rates for all components based on the aggregation of CTMC model, and the results are shown in Table 4.10. From this table, it is found that the repair rate of VM are much higher than that in Table 4.5. As mentioned before, the FT model considered the live migration as a static structure which cannot represent the dynamic behaviors of system. However, since the live migration is essentially described by a dynamic behavior, the dynamic behaviors have been taken into account in the CTMC model for live migration. The higher repair rate of VM confirms the effectiveness of live migration in the virtualized system. Table 4.11 presents the importance measures for components in the virtualized system. As observed in Table 4.10 and 4.11, we find that, although the failure rate of VM is higher than that of host, the importance measures of VM are much lower than those of host. This is because the repair rate of VM is very high. Also, comparing Table 4.9 with Table 4.10, we can see that the availability of host is the lowest among those of others, because the repair rate of host is also the lowest. This indicates that, the component host is important, and

Table 4.8 Model parameters.

Params	Description	Value
$1/\lambda_h$	Mean time for host failure	2654 hr
$1/\lambda_v$	Mean time for VM failure	2893 hr
$1/\lambda_a$	Mean time to Application failure	175 hr
$1/\delta_h$	Mean time for host failure detection	30 sec
$1/\delta_v$	Mean time for VM failure detection	30 sec
$1/\delta_a$	Mean time for App failure detection	30 sec
$1/m_v$	Mean time to migrate a VM	330 sec
$1/r_v$	Mean time to restart a VM	50 sec
$1/\mu_h$	Mean time to repair a host	100 min
$1/\mu_v$	Mean time to repair a VM	30 min
$1/\mu 1_a$	Mean time to App first repair (covered case)	1 min
$1/\mu 2_a$	Mean time to App second repair (not covered case)	20 min
c_v	coverage factor for VM repair	0.95
c_a	coverage factor for application repair	0.8

Table 4.9 Availabilities of host, VM, application components and system.

System	Availability
H1 and H2	0.9993644
VM1 and VM2	0.9999746
App1 and App2	0.9994520
System availability	0.9999992

any change in its associated parameters will have a large effect on the system availability. And this conclusion also can be confirmed from Table 4.11.

Table 4.11 shows that the importance measures of host is the most highest. Moreover, by comparing between the importance measure with respect to failure and repair rates for each component, it is found that the importance measure with respect to failure rate is higher than that with respect to repair rate. Therefore, it indicates that the improvement of failure rate of host is more efficient to enhance the system availability.

Table 4.10 Effective failure and repair rates.

Component	$\tilde{\lambda}_i$	$\tilde{\mu}_i$
H1 and H2	3.763673e-4	0.5917368
VM1 and VM2	7.212219e-4	28.351750
App1 and App2	6.425198e-3	11.718790

Table 4.11 Component importance measures in the dynamic model for live migration.

Component	$I_{\tilde{\lambda},i}$	$I_{\tilde{\mu},i}$
H1 and H2	2.118715e-03	1.347584e-06
VM1 and VM2	1.675414e-12	4.261977e-17
App1 and App2	9.438502e-13	5.174957e-16

4.4 Conclusion

This chapter has dealt with the quantitative component importance analysis of virtualized system with live migration in terms of availability. In Chapter 3, we have developed a method to evaluate the importance of components for hybrid model which consists of FTs and CTMCs. However, the hybrid model had a limitation for the model expression in the situation where two or more components have interactions between them. Instead of using the hybrid model, we considered a CTMC model for live migration presented in [19]. This chapter introduced the state-of-the-art Markov-based component-wise sensitivity analysis and applied it to the CTMC-based live migration model to reveal the component importance in the context of live migration without using structure function. In numerical examples, we illustrated the quantitative component importance analysis of live migration model for virtualized system, and indicated that component SAN is the most critical component in virtualized system. That means, the improvement in the failure rate and repair rates of SAN is more efficient to improve the system availability.

CHAPTER 5

COMPONENT IMPORTANCE MEASURES FOR REAL-TME COMPUTING SYSTEMS

Component importance analysis is to measure the effect on system reliability of component reliabilities, and it can be used to the design of system from the reliability point of view. In this chapter, we consider the component importance analysis of real-time computing systems in the presence of common-cause failures (CCFs) (i.e., failure dependencies) from the viewpoints of availability and reliability. Although the CCFs are known as a risk factor of degradation of system reliability, it is difficult to evaluate the component importance measures in the presence of CCFs analytically. This chapter introduces CTMC models for real-time computing system, and applies the Markov-based component-wise sensitivity analysis based on CTMCs which can evaluate the component importance measures without any structure function of system. Also, in numerical experiments, we evaluate the effect of CCFs by the comparison of system performance measures and component importance in the case of system with CCFs with those in the case that there is no CCF in the system.

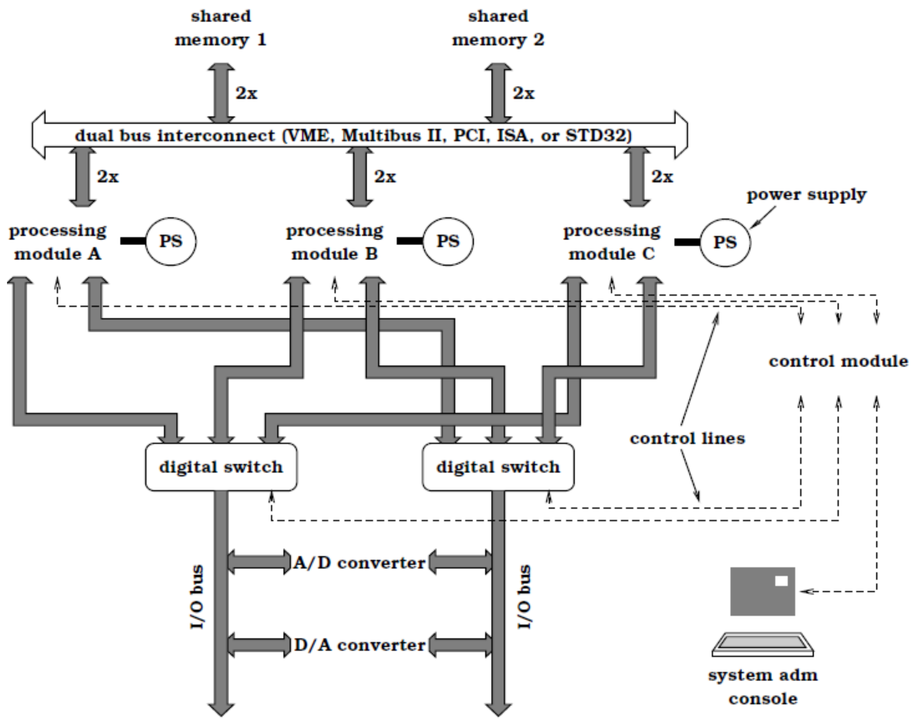

Figure 5.1 The architecture of real-time computing system.

5.1 Real-time Computing System

Consider the real-time computing system in [11] as shown in Fig. 5.1 . In the system, there are three processing modules (PMs), two shared memories (SMs), and two digital switches (DSs). The processing modules are implemented by a *pair-and-a-spare* fault-tolerant scheme [37], each consisting of processor(s), cache and local memories, power source, interface drives, and control circuitry. On the other hand, the parallel redundancy schemes are adopted to operate all the other critical system components (e.g., shared memories, input/output (I/O) bus, and digital switches). The communication among these processing modules is performed through shared memories where each processing module can read and store values, and the data transfer is achieved by using a parallel interconnecion bus. Additionally, the I/O bus interconnects the processing modules to external interface devices, and the Analog-to-Digital (A/D) and Digital-to-Analog (D/A) converters are connected directly to a dual I/O bus to provide redundant data/control path to the processing modules. Digital switches broadcast all data received from the I/O bus to all processing modules simultaneously.

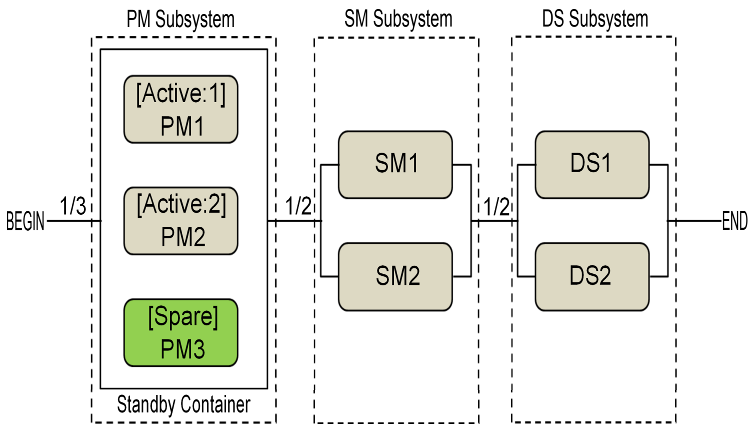

Figure 5.2 RBD of the real-time computing system.

Moreover, there is a control module in the system, which is responsible for selecting which of the online processor modules effectively controls the physical process. And the digital switches enforce the directives of the control module, that is, all processor modules may receive data from the physical process at any time, but only one can send control signals to the process.

We assume that the *pair-and-a-spare* configuration is implemented by a *hot standby sparing* scheme, where two processor modules operate online in synchrony, and a spare module runs simultaneously with the pair modules but will not process data or requests. However, data is mirrored in real time, thus both processor modules have identical data. Upon failure of the pair modules, the spare one immediately takes over, replacing the pair modules.

Fig. 5.2 shows the RBD of the real-time computing system. As seen in this figure, the system is divided into three subsystems: PM, SM, and DS subsystems. The system is considered operational as long as there is one operational critical component in each subsystem: processor module, shared memory, and digital switch. Also, we assume that there is a single infallible repair station for each component.

5.1.1 Subsystem Models

In [11], the behaviors of all subsystems are described by CTMCs which are commonly used to represent the variations caused by failures and repairs of components in the system structure. Particularly, since redundancy often increases common-mode errors, we consider the CCFs in PM subsystem.

5.1.1.1 Common-Cause Failure (CCF) As mentioned before, the CCF is defined as any condition or event that affects several components inducing their simultaneous failure or malfunction. Generally, there are three types of common-cause failures, that is (i) human errors, which can result in damage to equipment and property or disruption of scheduled operations of the system; (ii) system environment, including the characteristics of the environment where the system operates and the natural factors such as earthquake, fire, and flood; and (iii) intercomponent, which means that the failure of a component may affect adversely other components as a result of a chain reaction or domino effect. This chapter focuses on the intercomponent failure dependency model by using parametric approach. Concretely, we consider the *beta factor* introduced by Fleming [38]. The beta factor β gives the probability that a failure in a specific component causes all components to fail, and $1 - \beta$ gives the probability that the failure will involve just the component. Suppose that λ_i is the rate of independent failure killing single component, and λ_d is the rate of dependent failures killing all components. Then the overall failure rate λ of a particular component can be written as the sum of independent and dependent failure contributions:

$$\lambda = \lambda_i + \lambda_d. \tag{5.1}$$

Thus the β is defined as the fraction of the total failure rate attributable to dependent failures:

$$\beta = \frac{\lambda_d}{\lambda}. \tag{5.2}$$

Obviously, dssimilar components may have different failure rates and different beta factors.

5.1.1.2 PM Subsystem The CTMC of PM subsystem is depicted in Fig. 5.3. In this figure, white and gray nodes represent operational and failure states respectively. Table 5.1 shows the state notations which based on the current conditions of components. Concretely, each state is indicated by 3 characters. The first character means the state of component 1. When component 1 is operational, the character is given by ‘1’, if failed, it is ‘0’. The second and third characters represent the states of component 2 and component 3 respectively in the same manner as the first character. The model parameters are represented in Table 5.2. For example, $1/\lambda_{p1}$ is MTTF of PM1, and then λ_{p1} is a failure rate which is a transition rate in the CTMC. For three dissimilar components PM, we assume that they have the same beta factor β which gives

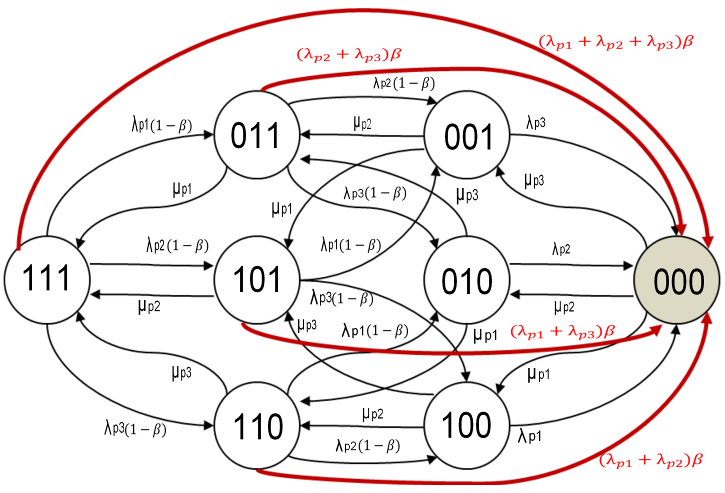

Figure 5.3 CTMC of PM subsystem.

the probability that a failure in one component causes all components to fail. As seen in Fig. 5.3, the simultaneous failures of all PMs and two PMs are highlighted by bold and red transitions.

Table 5.1 The states of PM subsystem.

State	Description
111	All PMs are operational.
011	PM1 is failed, PM2 and PM3 are operational.
101	PM2 is failed, PM1 and PM3 are operational.
110	PM3 is failed, PM1 and PM2 are operational.
001	PM1 and PM2 are failed, PM3 is operational.
010	PM1 and PM3 are failed, PM2 is operational.
100	PM2 and PM3 are failed, PM1 is operational.
000	PM subsystem is failed.

5.1.1.3 SM and DS Subsystems Assume that there is no CCF in SM and DS subsystems. Then we have the 4-state CTMC models represented in Fig. 5.4 for SM and DS subsystems. In this figure, the state notations are given in the same manner as the PM subsystem, and shown in Table 5.3. The parameters of the 4-state CTMC models are also given in Table 5.2.

Table 5.2 Model parameters.

Params	Description
$1/\lambda_{p1}$	Mean time to processing module 1 failure
$1/\lambda_{p2}$	Mean time to processing module 2 failure
$1/\lambda_{p3}$	Mean time to processing module 3 failure
$1/\lambda_{m1}$	Mean time to shared memory 1 failure
$1/\lambda_{m2}$	Mean time to shared memory 2 failure
$1/\lambda_{d1}$	Mean time to digital switch 1 failure
$1/\lambda_{d2}$	Mean time to digital switch 2 failure
$1/\mu_p$	Mean time to repair a processing module
$1/\mu_m$	Mean time to repair a shared memory
$1/\mu_d$	Mean time to repair a digital switch
β	Beta factor

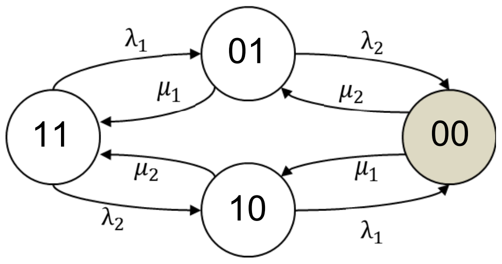

Figure 5.4 CTMC of SM/DS subsystem.

Table 5.3 The states of SM/DS subsystem.

State	Description
11	All SMs/DSs are operational.
01	SM1/DS1 is failed, SM2/DS2 is operational.
10	SM2/DS2 is failed, SM1/DS1 is operational.
00	SM/DS subsystem is failed.

5.2 Performance Evaluation

5.2.1 Structure Function

The structure function is a binary function that indicates the state of the system (success or failure) given the state of each component [39]. Given the structure function of a system, we can compute its reliability. In general, the structure function can be derived from FT and RBD.

Let $\boldsymbol{x} = (x_{PM1}, x_{PM2}, x_{PM3}, x_{SM1}, x_{SM2}, x_{DS1}, x_{DS2})$ be the state vector of real-time computing system, and the k-th element of $\boldsymbol{x}$ is a binary variable which represents the condition of component $k, k \in \{PM1,\ PM2,\ PM3,\ SM1,\ SM2,\ DS1,\ DS2\}$;

$$x_k = \begin{cases} 1, & \text{if component } k \text{ is operational,} \\ 0, & \text{if component } k \text{ is failed.} \end{cases} \tag{5.3}$$

The structure function represents the relationship between component failures and system failure. In general, the structure function is defined by

$$\phi(\boldsymbol{x}) = \begin{cases} 1, & \text{if system is operational,} \\ 0, & \text{if system is failed.} \end{cases} \tag{5.4}$$

For example, consider a system consisting of K components. If the system is a series system, namely, the system failure occurs when any component fails, the structure function is given by

$$\phi(\boldsymbol{x}) = x_1 x_2 \cdots x_K. \tag{5.5}$$

If the system failure occurs only when all the components fail, so-called parallel system, then the structure function is given below,

$$\phi(\boldsymbol{x}) = 1 - (1 - x_1)(1 - x_2) \cdots (1 - x_K). \tag{5.6}$$

According to the RBD in Fig. 5.2, we obtain the structure function of real-time computing system as follow,

$$\begin{aligned} \phi(\boldsymbol{x}) = \ & \big(1 - (1 - x_{PM1})(1 - x_{PM2})(1 - x_{PM3})\big) \\ & \cdot\big(1 - (1 - x_{SM1})(1 - x_{SM2})\big) \\ & \cdot\big(1 - (1 - x_{DS1})(1 - x_{DS2})\big). \end{aligned} \tag{5.7}$$

Let $P_{\boldsymbol{x}}(t)$ be a certain probability mass function of the system being in state $\boldsymbol{x}$ at time t. Then the reliability function of system can be computed by

$$R(t) = E[\phi(\boldsymbol{x})] = \sum_{\boldsymbol{x} \in \Omega} \phi(\boldsymbol{x}) P_{\boldsymbol{x}}(t), \tag{5.8}$$

where Ω is the state space of the system as shown in Table 5.4. Note that in this table, $R_k(t)$ indicates the reliability of component k at time t.

Table 5.4 Parallel-series reliability function.

$\boldsymbol{x}$	$\phi(\boldsymbol{x})$	$P_{\boldsymbol{x}}(t)$
1111111	1	$R_{PM1}(t)R_{PM2}(t)R_{PM3}(t)R_{SM1}(t)R_{SM2}(t)R_{DS1}(t)R_{DS2}(t)$
0111111	1	$[1-R_{PM1}(t)]R_{PM2}(t)R_{PM3}(t)R_{SM1}(t)R_{SM2}(t)R_{DS1}(t)R_{DS2}(t)$
0011111	1	$[1-R_{PM1}(t)][1-R_{PM2}(t)]R_{PM3}(t)R_{SM1}(t)R_{SM2}(t)R_{DS1}(t)R_{DS2}(t)$
0001111	0	$[1-R_{PM1}(t)][1-R_{PM2}(t)][1-R_{PM3}(t)]R_{SM1}(t)R_{SM2}(t)R_{DS1}(t)R_{DS2}(t)$
1110111	1	$R_{PM1}(t)R_{PM2}(t)R_{PM3}(t)[1-R_{SM1}(t)]R_{SM2}(t)R_{DS1}(t)R_{DS2}(t)$
1111011	1	$R_{PM1}(t)R_{PM2}(t)R_{PM3}(t)R_{SM1}(t)[1-R_{SM2}(t)]R_{DS1}(t)R_{DS2}(t)$
1110011	0	$R_{PM1}(t)R_{PM2}(t)R_{PM3}(t)[1-R_{SM1}(t)][1-R_{SM2}(t)]R_{DS1}(t)R_{DS2}(t)$
$\vdots$	$\vdots$	$\vdots$
0010101	1	$[1-R_{PM1}(t)][1-R_{PM2}(t)]R_{PM3}(t)[1-R_{SM1}(t)]R_{SM2}(t)[1-R_{DS1}(t)]R_{DS2}(t)$
$\vdots$	$\vdots$	$\vdots$
0000000	0	$[1-R_{PM1}(t)][1-R_{PM2}(t)][1-R_{PM3}(t)][1-R_{SM1}(t)][1-R_{SM2}(t)][1-R_{DS1}(t)][1-R_{DS2}(t)]$

On the other hand, the availability function of system is given by

$$A = E[\phi(\boldsymbol{x})] = \sum_{\boldsymbol{x}\in\Omega} \phi(\boldsymbol{x})P_{\boldsymbol{x}}, \tag{5.9}$$

where $P_{\boldsymbol{x}} = lim_{t->\infty}P_{\boldsymbol{x}}(t)$ when considering the repair for the component failures.

Using Eq. (5.8), we have the system reliability function $R_S(t)$;

$$\begin{aligned} R_S(t) &= \big(1-\overline{R}_{PM1}(t)\overline{R}_{PM2}(t)\overline{R}_{PM3}(t)\big) \\ &\quad \cdot\big(1-\overline{R}_{SM1}(t)\overline{R}_{SM2}(t)\big)\big(1-\overline{R}_{DS1}(t)\overline{R}_{DS2}(t)\big), \end{aligned} \tag{5.10}$$

where, in general $\overline{R}_k(t) = 1 - R_k(t)$. In practice, the above equation is often called the structure function which represents the effect of component reliability on the system reliability.

Then we define the steady-state availability of component k as A_k. Similarly, we obtain the structure function for the system availability A_S:

$$\begin{aligned} A_S &= (1-\overline{A}_{PM1}\overline{A}_{PM2}\overline{A}_{PM3})(1-\overline{A}_{SM1}\overline{A}_{SM2}) \\ &\quad \cdot(1-\overline{A}_{DS1}\overline{A}_{DS2}), \end{aligned} \tag{5.11}$$

where $\overline{A}_k = 1 - A_k$.

5.2.2 CTMC Analysis

Let Q_{PM}, Q_{SM}, Q_{DS} be the infinitesimal generators of CTMC for PM , SM, and DS subsystems, respectively. Then we have the composite CTMC generator for the real-time computing system by using the tensor sum of matrices as [40]

$$Q_S = Q_{PM} \oplus Q_{SM} \oplus Q_{DS}. \tag{5.12}$$

Generally, in the availability modeling of CTMC, the states of system can be classified into two sets; $\mathcal{U}$, the set of up (operational) states in which the system is available; and $\mathcal{D}$, the set of down (failed) states in which the system is unavailable. We define $\mathcal{U}_k$ and $\mathcal{D}_k$ as the sets of states where the component k is up or down, respectively. Also, $\mathcal{U}_S$ and $\mathcal{D}_S$ are the sets of states where the system is up or down. Then the reward vectors for component k and system can be defined by

$$[\boldsymbol{r}_k]_i = \begin{cases} 1, & i \in \mathcal{U}_k, \\ 0, & i \in \mathcal{D}_k, \end{cases} \tag{5.13}$$

and

$$[\boldsymbol{r}_S]_i = \begin{cases} 1, & i \in \mathcal{U}_S, \\ 0, & i \in \mathcal{D}_S, \end{cases} \tag{5.14}$$

respectively, where $[\cdot]_i$ means the i-th element of a vector. Using the reward vectors, the steady-state availabilities for component k and system are

$$A_S = \boldsymbol{\pi}_{ss}\boldsymbol{r}_S, \quad A_k = \boldsymbol{\pi}_{ss}\boldsymbol{r}_k, \tag{5.15}$$

where $\boldsymbol{\pi}_{ss}$ is the steady-state vector which can be computed by the following linear equations:

$$\boldsymbol{\pi}_{ss}\boldsymbol{Q}_S = \boldsymbol{0}, \quad \boldsymbol{\pi}_{ss}\boldsymbol{1} = 1, \tag{5.16}$$

where $\boldsymbol{1}$ is a column vector whose elements are 1.

On the other hand, if the underlying CTMC does not have transitions from $\mathcal{D}_k$ to $\mathcal{U}_k$ and from $\mathcal{D}_S$ to $\mathcal{U}_S$, the reliability functions of component k and system are given by

$$R_S(t) = \boldsymbol{\pi}(t)\boldsymbol{r}_S, \quad R_k(t) = \boldsymbol{\pi}(t)\boldsymbol{r}_k, \tag{5.17}$$

where $\boldsymbol{\pi}(t)$ is the state probability vector which is given by

$$\boldsymbol{\pi}(t) = \boldsymbol{\pi}(0)\exp(\boldsymbol{Q}_S t), \tag{5.18}$$

in which $\boldsymbol{\pi}(0)$ is a given initial probability vector.

5.3 Component Importance Analysis

5.3.1 Birnbaum Importance Measure

Birnbaum [41] defined the component importance from the reliability point of view. In [41], the component importance is defined by the first derivative of system reliability function with respect to the component reliability:

$$IB_k(t) = \frac{\partial R_S(t)}{\partial R_k(t)}. \tag{5.19}$$

Let $\delta_k(\boldsymbol{x})$ be the first derivative of structure function with respect to the state condition of component k:

$$\delta_k(\boldsymbol{x}) = \frac{\partial \phi(\boldsymbol{x})}{\partial x_k}. \tag{5.20}$$

Integrating Eqs. (5.8) and (5.20) into Eq. (5.19), then Birnbaum importance becomes

$$IB_k(t) = E[\delta_k(\boldsymbol{x})] = \sum_{\boldsymbol{x} \in \Omega} \delta_k(\boldsymbol{x}) P_{\boldsymbol{x}}(t). \tag{5.21}$$

Thus, we can compute the Birnbaum availability and reliability component importance measures (AIB and RIB) which are written by

$$AIB_k = \frac{\partial A_S}{\partial A_k}, \quad RIB_k(t) = \frac{\partial R_S(t)}{\partial R_k(t)}. \tag{5.22}$$

In general, we compute the RIB by using the first derivative of system reliability with respect to the component reliability after obtaining the system reliability structure function as in Eq. (5.10). The AIB can be computed by the same manner.

5.3.1.1 Case I: system without CCFs Suppose that there is no CCF in the real-time computing system, that is, all components are statistically independent, and we delete the bold and red transitions in the CTMC of PM subsystem. Thus the sensitivities of system performance index with respect to component performance indices can be obtained from the structure function analytically. For example, for the component PM1 in the system without CCFs, then using Eq. (5.11), the Birnbaum availability importance of component PM1 becomes

$$\begin{aligned} AIB_{PM1} &= \frac{\partial A_S}{\partial A_{PM1}} \\ &= (1 - A_{PM2} - A_{PM3} + A_{PM2}A_{PM3}) \\ &\quad \cdot(1 - \overline{A}_{SM1}\overline{A}_{SM2})(1 - \overline{A}_{DS1}\overline{A}_{DS2}), \end{aligned} \tag{5.23}$$

where the availability of each component can be computed by

$$A_k = \mu_k/(\mu_k + \lambda_k). \tag{5.24}$$

Similarly, the Birnbaum reliability component importance measure $RIB_k(t)$ can also be obtained from structure function. For example, the Birnbaum reliability importance of component PM1 is given by

$$\begin{aligned} RIB_{PM1}(t) &= \frac{\partial R_S(t)}{\partial R_{PM1}(t)} \\ &= (1 - R_{PM2}(t) - R_{PM3}(t) + R_{PM2}(t)R_{PM3}(t)) \\ &\quad \cdot(1 - \overline{R}_{SM1}(t)\overline{R}_{SM2}(t))(1 - \overline{R}_{DS1}(t)\overline{R}_{DS2}(t)), \end{aligned} \tag{5.25}$$

in which, the reliability of each component is computed from

$$R_k(t) = \exp(-\lambda_k t). \tag{5.26}$$

5.3.1.2 Case II: system with CCFs However, in practice, the system failure often occurs due to the CCFs. For example, the real-time computing system where the intercomponent dependent failures occur among the components in PM subsystem as seen in Fig. 5.3. In such case, we cannot obtain the above sensitivities from structure function analytically. Then we consider the component-wise sensitivity analysis presented in Chapter 3, which can be used to compute $\partial A_S/\partial A_k$ and $\partial R_S(t)/\partial R_k$ directly based on Markov chains. Concretely, for the real-time computing system with components PM1, PM2, PM3, SM1, SM2, DS1, and DS2, and model parameter vector

$\boldsymbol{\theta} = (\theta_1, \dots, \theta_m)$, we compute

$$\boldsymbol{z}_A = \begin{pmatrix} \frac{\partial A_S}{\partial \theta_1} \\ \vdots \\ \frac{\partial A_S}{\partial \theta_m} \end{pmatrix}, \quad \boldsymbol{J}_A = \begin{pmatrix} \frac{\partial A_{PM1}}{\partial \theta_1} & \dots & \frac{\partial A_{DS2}}{\partial \theta_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial A_{PM1}}{\partial \theta_m} & \dots & \frac{\partial A_{DS2}}{\partial \theta_m} \end{pmatrix}, \tag{5.27}$$

$$\boldsymbol{z}_R(t) = \begin{pmatrix} \frac{\partial R_S(t)}{\partial \theta_1} \\ \vdots \\ \frac{\partial R_S(t)}{\partial \theta_m} \end{pmatrix}, \quad \boldsymbol{J}_R(t) = \begin{pmatrix} \frac{\partial R_{PM1}(t)}{\partial \theta_1} & \dots & \frac{\partial R_{DS2}(t)}{\partial \theta_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial R_{PM1}(t)}{\partial \theta_m} & \dots & \frac{\partial R_{DS2}(t)}{\partial \theta_m} \end{pmatrix}. \tag{5.28}$$

In the case of steady-state analysis, these sensitivities can be obtained by solving the following linear equations:

$$\boldsymbol{s}_{ss}(\theta_j)\boldsymbol{Q}_S = -\boldsymbol{\pi}_{ss}\frac{\partial}{\partial \theta_j}\boldsymbol{Q}_S, \quad \boldsymbol{s}_{ss}(\theta_j)\boldsymbol{1} = 0, \tag{5.29}$$

where $\boldsymbol{s}_{ss}(\theta_j) = \frac{\partial}{\partial \theta_j}\boldsymbol{\pi}_{ss}$. By using the vector $\boldsymbol{s}_{ss}(\theta_j)$, the sensitivities are given by

$$\frac{\partial A_k}{\partial \theta_j} = \boldsymbol{s}_{ss}(\theta_j)\boldsymbol{r}_k, \quad \frac{\partial A_S}{\partial \theta_j} = \boldsymbol{s}_{ss}(\theta_j)\boldsymbol{r}_S \tag{5.30}$$

In the transient case, we have

$$\frac{d}{dt}\boldsymbol{\pi}(t) \ = \boldsymbol{\pi}(t)\boldsymbol{Q}_S, \tag{5.31}$$

$$\boldsymbol{s}(t, \theta_j) \ = \tfrac{\partial}{\partial \theta_j}\boldsymbol{\pi}(t). \tag{5.32}$$

Then the following ordinary differential equation (ODE) is obtained,

$$\frac{d}{dt}\boldsymbol{s}(t, \theta_j) = \boldsymbol{s}(t, \theta_j)\boldsymbol{Q}_S + \boldsymbol{\pi}(t)\frac{\partial}{\partial \theta_j}\boldsymbol{Q}_S. \tag{5.33}$$

By integrating Eq. (5.31) into the above ODE, we have

$$\frac{d}{dt}\tilde{\boldsymbol{\pi}}(t, \theta_j) = \tilde{\boldsymbol{\pi}}(t, \theta_j)\tilde{\boldsymbol{Q}}(\theta_j), \tag{5.34}$$

where

$$\tilde{\boldsymbol{\pi}}(t, \theta_j) = (\boldsymbol{\pi}(t), \boldsymbol{s}(t, \theta_j)), \quad \tilde{\boldsymbol{Q}}(\theta_j) = \begin{pmatrix} \boldsymbol{Q}_S & \frac{\partial}{\partial \theta_j}\boldsymbol{Q}_S \\ & \boldsymbol{Q}_S \end{pmatrix}. \tag{5.35}$$

Since the diagonal elements of $\tilde{\boldsymbol{Q}}(\theta_j)$ are same as those of $\boldsymbol{Q}_S$, we can apply the uniformization to the following matrix exponential form:

$$\tilde{\boldsymbol{\pi}}(t,\theta_j) = \tilde{\boldsymbol{\pi}}(0,\theta_j)\exp\left(\tilde{\boldsymbol{Q}}(\theta_j)t\right). \tag{5.36}$$

Then the estimates of $\boldsymbol{AIB} = (AIB_{PM1},\ldots,AIB_{DS2})^T$ and $\boldsymbol{RIB}(t) = (RIB_{PM1}(t),\ldots,RIB_{DS2}(t))^T$ are given by

$$\boldsymbol{AIB} = (\boldsymbol{J}_A^T\boldsymbol{J}_A)^{-1}\boldsymbol{J}_A^T\boldsymbol{z}_A, \tag{5.37}$$

$$\boldsymbol{RIB}(t) = \left(\boldsymbol{J}_R(t)^T\boldsymbol{J}_R(t)\right)^{-1}\boldsymbol{J}_R(t)^T\boldsymbol{z}_R(t). \tag{5.38}$$

5.3.2 Criticality Importance Measure

The criticality measure was proposed by Henley et al. [42] which means the probability that, when the system fails, the failure of component k becomes a cause of the system failure. They defined the criticality importance of component k as a fractional sensitivity given by

$$ICF_k(t) = \frac{F_k(t)}{F_S(t)}\frac{\partial F_S(t)}{\partial F_k(t)}, \tag{5.39}$$

where $F_k(t)$ and $F_S(t)$ are the unreliability functions of component k and system at time t respectively, and given by

$$\begin{aligned} F_k(t) &= 1 - R_k(t), \\ F_S(t) &= 1 - R_S(t). \end{aligned} \tag{5.40}$$

Similarly, according to [43], the Eq. (5.39) can be represented by the reliability functions of system and component, ie.,

$$ICR_k(t) = \frac{R_k(t)}{R_S(t)}\frac{\partial R_S(t)}{\partial R_k(t)}. \tag{5.41}$$

Thus, the criticality importance measures of availability and reliability (AICF and RICF) from the unreliability point of view can be derived by

$$AICF_k = \frac{F_k}{F_S}\frac{\partial F_S}{\partial F_k} = \frac{F_k}{F_S}\frac{\partial A_S}{\partial A_k}, \tag{5.42}$$

$$RICF_k(t) = \frac{F_k(t)}{F_S(t)}\frac{\partial F_S(t)}{\partial F_k(t)} = \frac{F_k(t)}{F_S(t)}\frac{\partial R_S(t)}{\partial R_k(t)}, \tag{5.43}$$

where $F_k = 1 - A_k$, and $F_S = 1 - A_S$.

Also, according to Eq. (5.41), we obtain the criticality importance measures of availability and reliability (AICR and RICR) from the viewpoint of reliability given by

$$AICR_k = \tfrac{A_k}{A_S}\tfrac{\partial A_S}{\partial A_k}, \tag{5.44}$$

$$RICR_k(t) = \tfrac{R_k(t)}{R_S(t)}\tfrac{\partial R_S(t)}{\partial R_k(t)}. \tag{5.45}$$

Essentially, these measures can be computed from AIB_k and $RIB_k(t)$, i.e.,

$$AICF_k = \frac{F_k}{F_S} AIB_k, \quad RICF_k(t) = \frac{F_k(t)}{F_S(t)} RIB_k(t), \tag{5.46}$$

and

$$AICR_k = \frac{A_k}{A_S} AIB_k, \quad RICR_k(t) = \frac{R_k(t)}{R_S(t)} RIB_k(t). \tag{5.47}$$

5.3.3 Upgrading function

The upgrading function is the parametric sensitivity function with respect to a failure rate [22]. According to the definition, we have the availability and reliability upgrading functions (AIU and RIU) for component k:

$$AIU_{k,\lambda} = \tfrac{\lambda_k}{A_S}\tfrac{\partial A_S}{\partial \lambda_k}, \tag{5.48}$$

$$AIU_{k,\mu} = \tfrac{\mu_k}{A_S}\tfrac{\partial A_S}{\partial \mu_k}, \tag{5.49}$$

$$RIU_{k,\lambda}(t) = \tfrac{\lambda_k}{R_S(t)}\tfrac{\partial R_S(t)}{\partial \lambda_k}, \tag{5.50}$$

where λ_k and μ_k are failure and repair rates of component k. Note that AIU can also be defined for the repair rate. The AIU is essentially same as the availability importance measures discussed by Cassady et al. [33].

In [33] and [32], they assumed that components are independent and that the component has only two states, up and down on the underlying CTMC. On the other hand, in Chapter 3, we have introduced the method to derive the availability upgrading functions under which components are described by general CTMCs. The idea of our approach is to apply the aggregation technique [35]. However, even in Chapter 3, components are assumed to be independent. Then by applying the sensitivity estimation (see Appendix B) and aggregation, we derive AIUs and RIU for MRMs.

First we consider AIUs. The aggregation is a technique to reduce MRM-based availability models to the 2-state model which has the same availability as the original model. When we focus on the state of one component, the states can be classified to $\mathcal{U}_k$ and $\mathcal{D}_k$. The aggregation technique converts the original model to the 2-state model with transitions from up to down states and up to down state. By applying this technique, we obtain failure and repair rates in steady state that ensure the steady-state probabilities of the up (down) states are the same as those in the original model. In this paper, these failure and repair rates are called *equivalent failure and repair rates* [35].

From the argument of CTMC, equivalent failure and repair rates of component k can be computed as follows.

$$\lambda_k = \frac{\sum_{(i,j)\in\mathcal{U}_k\times\mathcal{D}_k}[\boldsymbol{\pi}_{ss}]_i[\boldsymbol{Q}]_{i,j}}{\sum_{i\in\mathcal{U}_k}[\boldsymbol{\pi}_{ss}]_i}, \tag{5.51}$$

$$\mu_k = \frac{\sum_{(j,i)\in\mathcal{D}_k\times\mathcal{U}_k}[\boldsymbol{\pi}_{ss}]_j[\boldsymbol{Q}]_{j,i}}{\sum_{j\in\mathcal{D}_k}[\boldsymbol{\pi}_{ss}]_j}, \tag{5.52}$$

where $[\cdot]_{i,j}$ is an (i,j)-entry of $\boldsymbol{Q}$. By taking account of $A_k = \mu_k/(\lambda_k+\mu_k)$, $AIU_{k,\lambda}$ can be rewritten by

$$\begin{aligned} AIU_{k,\lambda} &= \frac{\lambda_k}{A_S}\frac{\partial A_S}{\partial \lambda_k} \\ &= \frac{\lambda_k}{A_S}\sum_{l=1}^{K}\frac{\partial A_S}{\partial A_l}\frac{\partial A_l}{\partial \lambda_k} \\ &= \frac{\lambda_k}{A_S}\frac{\partial A_S}{\partial A_k}\frac{\partial A_k}{\partial \lambda_k} \\ &= \frac{\lambda_k}{A_S}\frac{\partial A_S}{\partial A_k}\left(-\frac{\mu_k}{(\lambda_k+\mu_k)^2}\right) \\ &= -\frac{A_k}{A_S}\frac{\partial A_S}{\partial A_k}\frac{\lambda_k}{\lambda_k+\mu_k} \\ &= -(1-A_k)AICR_k. \end{aligned} \tag{5.53}$$

Similarly, $AIU_{k,\mu}$ becomes

$$\begin{aligned}
AIU_{k,\mu} &= \frac{\mu_k}{A_S}\frac{\partial A_S}{\partial \mu_k} \\
&= \frac{\mu_k}{A_S}\frac{\partial A_S}{\partial A_k}\frac{\partial A_k}{\partial \mu_k} \\
&= \frac{\mu_k}{A_S}\frac{\partial A_S}{\partial A_k}\frac{\lambda_k}{(\lambda_k+\mu_k)^2} \\
&= \frac{A_k}{A_S}\frac{\partial A_S}{\partial A_k}\frac{\lambda_k}{\lambda_k+\mu_k} \\
&= (1-A_k)AICR_k. \qquad (5.54)
\end{aligned}$$

Next we consider RIU. In this case, the equivalent failure rate cannot be computed. Instead of the equivalent failure rate, we use the time-dependent failure rate, i.e., $\lambda_k(t) = -(dR_k(t)/dt)/(1-R_k(t))$. Generally, the relationship between reliability function and failure rate is given by

$$R_k(t) = e^{-\int_0^t \lambda_k(s)ds}. \qquad (5.55)$$

Based on this failure rate, RIU can be obtained by

$$\begin{aligned}
RIU_{k,\lambda} &= \frac{\lambda_k(t)}{R_S(t)}\frac{\partial R_S(t)}{\partial \lambda_k(t)} \\
&= \frac{\lambda_k(t)}{R_S(t)}\sum_{l=1}^{K}\frac{\partial R_S(t)}{\partial R_l(t)}\frac{\partial R_l(t)}{\partial \lambda_k(t)} \\
&= \frac{\lambda_k(t)}{R_S(t)}\frac{\partial R_S(t)}{\partial R_k(t)}\frac{\partial R_k(t)}{\partial \lambda_k(t)} \\
&= \frac{\lambda_k(t)}{R_S(t)}\frac{\partial R_S(t)}{\partial R_k(t)}(-tR_k(t)) \\
&= -t\lambda_k(t)\frac{R_k(t)}{R_S(t)}\frac{\partial R_S(t)}{\partial R_k(t)} \\
&= -t\lambda_k(t)RICR_k(t). \qquad (5.56)
\end{aligned}$$

In the Markov chain, the failure rate of component k is

$$\lambda_k(t) = -\frac{\boldsymbol{\pi}(t)\boldsymbol{Q}\boldsymbol{r}_k}{\boldsymbol{\pi}(t)\boldsymbol{r}_k}. \qquad (5.57)$$

5.4 Numerical Illustration

In this section, we illustrate the quantitative component analysis of real-time computing system. Concretely, we compute the importance measures using structure function in the case where there is no CCF in the system. On other hand, for the system with CCFs case, the CTMC-based component-wise sensitivity analysis is applied. Based on the results, we evaluate the effect of CCFs on system performance measures and component importance. Table 5.5 shows the model parameters.

Table 5.5 Model parameters.

Params	Description	Value
$1/\lambda_{p1}$	Mean time to processing module 1 failure	1250 hr
$1/\lambda_{p2}$	Mean time to processing module 2 failure	1000 hr
$1/\lambda_{p3}$	Mean time to processing module 3 failure	800 hr
$1/\lambda_{m1}$	Mean time to shared memory 1 failure	200 hr
$1/\lambda_{m2}$	Mean time to shared memory 2 failure	155 hr
$1/\lambda_{d1}$	Mean time to digital switch 1 failure	500 hr
$1/\lambda_{d2}$	Mean time to digital switch 2 failure	425 hr
$1/\mu_p$	Mean time to repair a processing module	30 min
$1/\mu_m$	Mean time to repair a shared memory	30 min
$1/\mu_d$	Mean time to repair a digital switch	30 min
β	Beta factor	0.02

5.4.1 System without CCFs

Before considering the real-time computing system with CCFs, we first evaluate the component importance measures of the system without CCFs by using structure function. That is, we consider the CTMC of PM subsystem without bold and red transitions. According to the RBD analysis, we obtain the same structure functions for the system availability and reliability as in Eqs. (5.11) and (5.10), and the availabilities and reliabilities of subsystems and system at time $t = 60$ hours are given in Table 5.6. From the table, we find that both the availability and reliability of PM subsystem are the highest, and approximately equals 1 because of the *pair-and-a-spare* fault-tolerant scheme.

Table 5.6 Availabilities and reliabilities without CCFs.

Subsystem	Availability	Reliability ($t = 60$ hrs)
PM subsystem	0.99999	0.99980
SM subsystem	0.99987	0.91680
DS subsystem	0.99998	0.98511
System	0.99986	0.90298

Table 5.7 Availabilities and importance measures without CCFs.

Comp.	Availability	AIB	$AICR$	$AICF$	AIU_λ	AIU_μ
PM1	0.99840	4.976855e-06	4.969625e-06	5.490746e-05	-7.938698e-09	7.938698e-09
PM2	0.99800	3.983074e-06	3.975700e-06	5.490746e-05	-7.935528e-09	7.935528e-09
PM3	0.99750	3.188050e-06	3.180560e-06	5.490746e-05	-7.931571e-09	7.931571e-09
SM1	0.99010	1.273862e-02	1.261432e-02	8.710670e-01	-1.248942e-04	1.248942e-04
SM2	0.98726	9.900805e-03	9.776096e-03	8.710670e-01	-1.245363e-04	1.245363e-04
DS1	0.99602	4.683250e-03	4.665267e-03	1.288618e-01	-1.858672e-05	1.858672e-05
DS2	0.99532	3.983561e-03	3.965477e-03	1.288618e-01	-1.857366e-05	1.857366e-05

Table 5.8 Reliabilities and importance measures at $t = 60$ hours without CCFs.

Comp.	Reliability	RIB	$RICR$	$RICF$	RIU_λ
PM1	0.95313	3.800397e-03	4.011475e-03	1.835833e-03	-1.925508e-04
PM2	0.94176	3.058449e-03	3.189810e-03	1.835832e-03	-1.913886e-04
PM3	0.92774	2.464971e-03	2.532567e-03	1.835832e-03	-1.899425e-04
SM1	0.74082	3.161334e-01	2.593602e-01	8.445377e-01	-7.780807e-02
SM2	0.67903	2.552726e-01	1.919603e-01	8.445376e-01	-7.430723e-02
DS1	0.88692	1.206869e-01	1.185403e-01	1.406658e-01	-1.422484e-02
DS2	0.86834	1.036519e-01	9.967508e-02	1.406658e-01	-1.407178e-02

Also, Tables 5.7 and 5.8 show the availabilities and availability importance measures (AIB, AICR, AICF, and AIU), and reliabilities and reliability importance measures (RIB, RICR, RICF, and RIU) at time $t = 60$ hours of each component, respectively. As seen in these tables, the availabilities and reliabilities of PMs are relatively higher, compared to the other components, since the failure rates of PMs are relatively lower. From Table 5.7, we find that in each subsystem, the component with lower failure rate has higher importance measures AIB, AICR, and AIU. The same result appears in the reliability importance measures RIB, RICR, and RIU shown in Table 5.8. This is caused by the fact that the failure of a high-reliability component always decreases the system reliability largely in the redundancy system in a parallel configuration. Note that $ICF_k(t)$ is another criticality measure defined by the unreliability function that quantifies the probability of component k being responsible for system failure before t hours. For example, in Table 5.8, $RICF_{SM1}(60)$ says that there is a 84.45% probability of component SM1 being responsible for system failure before 60 hours. Moreover, $RICF_{PM1}(60)$ indicates that the probability of component PM1 being responsible for system failure before 60 hours is only 00.18%. This implies that component SM1 is more important than component PM1, since a component that is frequently critical should be considered important. Moreover, we find that the ICFs for components in the same subsystem are almost the same.

Tables 5.9 and 5.10 illustrate the importance ranking of system components according to distinct measures. From these tables, we find that the importance rankings in steady-state and transition state ($t = 60$ hours) are the same, and there is consistency in the ranking of components for three measures (IB, ICR, and IU). The components SM1 and SM2 are more important than the other components, because their importance measures are relatively higher (see Tables 5.7 and 5.8). This indicates that the improvement of failure rates of SMs is more efficient to enhance the system availability and reliability, since the component most susceptible to failures is the natural candidate for improvement. In addition, these tables imply that the component PM is not critical, compared to the other components, benefiting from the *hot standby sparing* configuration and long MTTF (see Table 5.5).

5.4.2 System with CCFs

Next we focus on the case where there are CCFs in PM subsystem (see Fig. 5.3). The component-wise sensitivity analysis is applied to compute the importance measures based on CTMCs, and the results are shown in from Table 5.11 to 5.13. Table 5.11 presents the availabilities and reliabilities at time $t = 60$ hours. Compared to the availabilities and reliabilities in Table 5.6, the system availability and reliability become smaller, because the availabil-

Table 5.9 Importance ranking of system components with respect to availability without CCFs.

Comp.	AIB	$AICR$	AIU
PM1	5	5	5
PM2	6	6	6
PM3	7	7	7
SM1	1	1	1
SM2	2	2	2
DS1	3	3	3
DS2	4	4	4

Table 5.10 Importance ranking of system components with respect to reliability without CCFs.

Comp.	RIB	$RICR$	RIU
PM1	5	5	5
PM2	6	6	6
PM3	7	7	7
SM1	1	1	1
SM2	2	2	2
DS1	3	3	3
DS2	4	4	4

ity and reliability of PM subsystem decrease due to the dependent failures among PMs.

Table 5.11 Availabilities and reliabilities with CCFs.

Subsystem	Availability	Reliability (t = 60 hrs)
PM subsystem	0.99996	0.99627
SM subsystem	0.99987	0.91681
DS subsystem	0.99998	0.98511
System	0.99981	0.89979

Moreover, Tables 5.12 and 5.13 present the availabilities and availability importance measures (AIB, AICR, AICF, and AIU), and reliabilities and reliability importance measures (RIB, RICR, RICF, and RIU) at time $t = 60$

Table 5.12 Availabilities and importance measures with CCFs.

Comp.	Availability	AIB	$AICR$	$AICF$	AIU_λ	AIU_μ
PM1	0.99831	6.429934e-03	6.420277e-03	5.850368e-02	-1.083064e-05	1.083064e-05
PM2	0.99792	6.428668e-03	6.416502e-03	7.203403e-02	-1.333027e-05	1.333027e-05
PM3	0.99743	6.427512e-03	6.412213e-03	8.893078e-02	-1.644906e-05	1.644906e-05
SM1	0.99010	1.273810e-02	1.261432e-02	6.802356e-01	-1.248942e-04	1.248942e-04
SM2	0.98726	9.900403e-03	9.776096e-03	6.802356e-01	-1.245363e-04	1.245363e-04
DS1	0.99602	4.683060e-03	4.665267e-03	1.006310e-01	-1.858672e-05	1.858672e-05
DS2	0.99531	3.983399e-03	3.965477e-03	1.006310e-01	-1.857366e-05	1.857366e-05

Table 5.13 Reliabilities and importance measures at $t = 60$ hours with CCFs.

Comp.	Reliability	RIB	$RICR$	$RICF$	RIU_λ
PM1	0.95064	2.090991e-02	2.209173e-02	1.029804e-02	-1.116186e-03
PM2	0.93952	2.019041e-02	2.108192e-02	1.218537e-02	-1.313604e-03
PM3	0.92579	1.961298e-02	2.017976e-02	1.452370e-02	-1.554796e-03
SM1	0.74082	3.150168e-01	2.593602e-01	8.147694e-01	-7.780807e-02
SM2	0.67903	2.543709e-01	1.919603e-01	8.147694e-01	-7.430723e-02
DS1	0.88692	1.202606e-01	1.185403e-01	1.357076e-01	-1.422484e-02
DS2	0.86833	1.032858e-01	9.967508e-02	1.357076e-01	-1.407178e-02

hours of each component in the case where there are some CCFs in the system. From these tables, it is found that the importance of each component PM increases sharply due to the CCFs, especially the importance measures regarding the system availability. In fact, the dependent failures also decrease the availability and reliability of each component PM, since the time-dependent failure rate of each component PM increases caused by the dependent failures (see Table 5.14). On the other hand, the availabilities and reliabilities of the other components (e.g., SMs and DSs) remain the same because any failure in the PM subsystem cannot cause the simultaneous failure of components in the other subsystems. However, these tables also indicate that the importance of components SMs and DSs is slightly affected by the dependent failures among PMs, and decreases. For example, from Table 5.13, $RICF_{SM1}(60)$ says that the probability of component SM1 being responsible for system failure before

60 hours is 81.48%, which is smaller than that in the case of system without CCFs (see Table 5.8).

Table 5.14 Time-dependent failure rates ($t = 60$ hours).

Comp.	Failure rate	Time-dependent failure rate	
		Case I: system without CCFs	Case II: system with CCFs
PM1	0.000800000	0.000800000	0.000842084
PM2	0.001000000	0.001000000	0.001038492
PM3	0.001250000	0.001250000	0.001284122
SM1	0.005000000	0.005000000	0.005000000
SM2	0.006451613	0.006451613	0.006451613
DS1	0.002000000	0.002000000	0.002000000
DS2	0.002352941	0.002352941	0.002352941

Table 5.15 Importance ranking of system components with respect to availability with CCFs.

Comp.	AIB	$AICR$	AIU
PM1	3	3	7
PM2	4	4	6
PM3	5	5	5
SM1	1	1	1
SM2	2	2	2
DS1	6	6	3
DS2	7	7	4

We then investigate the importance ranking of components in the case where simultaneous failures occur in the PM subsystem. Tables 5.15 and 5.16 show the importance rankings of system components in steady-state and transition state ($t = 60$ hours), respectively. From Table 5.15, we find that the ranking of components SMs still remains the same, however the importance (AIB and AICR) of each component PM increases and becomes larger than that of components DSs in the case of system with CCFs, compared to Table 5.9. When considering AIU, we find that in the PM subsystem, the component with lower failure rate has lower importance, that is, component PM3 is the most important one, followed by component PM2, and finally component PM1. This

Table 5.16 Importance ranking of system components with respect to reliability with CCFs.

Comp.	RIB	$RICR$	RIU
PM1	5	5	7
PM2	6	6	6
PM3	7	7	5
SM1	1	1	1
SM2	2	2	2
DS1	3	3	3
DS2	4	4	4

can be explained by the fact that there is the smallest reduction in the failure probability of the system when the failure rate of component PM1 is reduced fractionally, in other words, components PM2 and PM3 are more critical to system failure by the comparison to component PM1, since there are two reasons; (i) the failure of each component PM probably causes all components to fail (i.e., system failure); and (ii) the failure rate of PM1 is the smallest in three PMs, thus the probability that the system failure is caused by component PM1 is also the smallest. From the viewpoint of reliability, we can conclude the same mark (see RIU in Table 5.16). In addition, Table 5.16 indicates that the dependent failures do not affect the importance ranking of components in cases of RIB and RICR from the reliability point of view.

5.5 Conclusion

In this chapter, we have quantitatively evaluated the component importance measures of a real-time computing system [11] where CCFs occur. Concretely, we evaluated three kinds of importance measures in the case of system without CCFs by the common method using hybrid model (RBD and CTMCs) and structure function of system, and in the case of system with CCFs using Markov-based component-wise sensitivity analysis, respectively. When considering the system without CCFs, the above two methods can be applied to compute the important measures. However, the common method does not hold in the case where components are dependent. In such case, the Markov-based component-wise sensitivity analysis is adopted. In numerical experiments, we illustrated the quantitative importance measures of all components, and ranked the components according to distinct importance measures. Also, we considered the effect of failure dependencies. Our numerical experiments show that components SMs are the most critical components in the real-time computing system, thus the improvement of failure rates of SMs is more efficient to enhance the system performance.

CHAPTER 6

SURVIVABILITY ANALYSIS OF VM-BASED INTRUSION TOLERANT SYSTEMS

This chapter discusses the environmental sensitivity that measures the effect of environmental changes on system dependability. Survivability is considered, which is the capability of a system to provide its services in a timely manner even after intrusion and compromise occur. In fact, survivability analysis helps to find the best design of system. In this chapter, we focus on the quantitative analysis of survivability of VM-based intrusion tolerant system in the presence of Byzantine failures due to malicious attacks. Intrusion tolerant system has the ability of a system to continuously provide correct services even if the system is intruded. This chapter presents a generalized scheme of the intrusion tolerant system with virtualization, and build a stochastic model by Markov chain. Based on the model, we compute two system performance indices: the success probability for one request from clients under malicious attacks, and the conditional success probability provided that the system has been intruded. The latter is a survivability metric to evaluate the system ability to mask the intrusion.

6.1 VM-Based Intrusion Tolerant System

Consider the intrusion tolerant system in the virtualized environment [21] as shown in Fig. 6.1. In the system, we suppose that there are n (≥ 2) VMs and component agreement service (AS) running on a hypervisor. The VMs provide identical services to clients. The communication among VMs is performed through a shared memory in the physical host where any VMs can read and store values. The AS provides a voting process and puts a signature into an agreed response, is kept isolated, and assumed to be reliable in the system. When the system receives a request from a client, the request is copied and sent to all the VMs. Each of the VMs processes the request independently and sends the response to AS. After receiving all the responses from all VMs, AS votes to make a final response based on all the responses, and puts a signature into the final response, then sends it to the client.

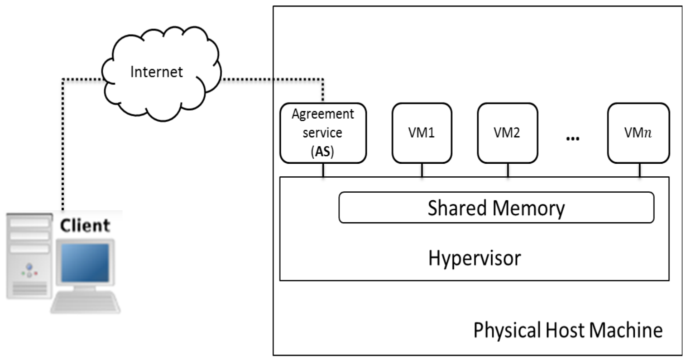

Figure 6.1 Architecture of VM-based intrusion tolerant system.

If all the VMs are normal (not intruded), all the responses are identical. Thus it is possible to make the agreement of the final response in AS. However, VMs are likely to be maliciously intruded when processing requests. In general, intruded VMs can be regarded as VMs with Byzantine failures. In other words, we cannot forecast the behavior of intruded VMs. Even in such case, AS can make the agreement when the number of intruded VMs is less than $n/2$. Let f be the tolerance level, namely, the maximum number of intruded VMs that are tolerated to the extent that the system behaves normal. From the argument of Byzantine fault tolerance, the inequalities $n \geq 2f + 1$ and $f \leq (n-1)/2$, hold.

Lau et al. [21] proposed a scheme of intrusion tolerance with two rounds of processing. Their idea is to achieve the tolerance level f at the second round of processing. On the other hand, to save resource usage in the system, the number of VMs is reduced at the first round. Concretely, for a given tolerance level f, they assumed that $f+1$ VMs were initially activated, and $f+1$ VMs process a request at the first round. In their scheme, AS can make the agreement after finishing all the processes in VMs at the first round if and only if the following two conditions hold; (i) all the VMs have sent responses to AS and (ii) all of the responses are coincident. It should be noted that the tolerance level at the first round in Lau's scheme is 0, i.e., no intrusion is tolerated at the first round. If the agreement is not made, AS resends the request to VMs including already-intruded VMs, at the second round after activating f suspended VMs. At the second round, AS makes the agreement by a majority voting, i.e., AS makes the agreement when at least $f+1$ responses from VMs are coincident. At that time, the tolerance level f is achieved in the system.

Lau's scheme can be extended from two points. One is to allow to assign arbitrary integers to the numbers of initially and additionally activated VMs. In Lau's scheme, since the tolerance level at the first round is $f = 0$, the minimum number of initially activated VMs is $n = 2$. In this case, the number of additionally activated VMs after the first round is $r = 2f - 1$. From the viewpoint of resource usage, this configuration seems to be better than the original one. Another point is to allow to use 3 or more rounds for the agreement. In Lau's scheme, the tolerance level is fixed as f. However, if the agreement was not made at the second round, it would be better to make the agreement as the tolerance level increases at the third round.

According to these two points, this paper considers a generalized scheme including Lau's scheme. Concretely, AS try to make the agreement within m rounds using n initially activated VMs and r additionally activated VMs at each round. In this case, the maximum tolerance level of system is given by $n + r(m-1) = 2f + 1$ and $f = \big(n + r(m-1) - 1\big)/2$. In addition, we allow that AS can make the agreement at the first round if the number of intruded VMs is less than $n/2$, i.e., the tolerance level at the first round is $f = n/2$ which is more severe than $f = \big(n + r(m-1) - 1\big)/2$. That is, in the generalized scheme, the intrusion tolerance is varied with respect to the round and it becomes relaxed as the number of rounds increases.

We illustrate the behavior of the system achieved by the generalized scheme with $n = 3$ and $r = 2$ for one request in Fig. 6.2. Suppose that the number of initially activated VMs is 3, denoted by VM1, VM2, and VM3. All the VMs are normal when the system receives a request form a client. The request is copied and sent to all the VMs, then each of the VMs processes the request independently and sends the response to AS. Then AS votes to make the agreement based on all the responses. As seen in the figure, the system is failed to process the request (i.e., the agreement is failed) at the first round because VM1 and VM3 are intruded by malicious attacks (the number of normal VM (VM2) $<$ the number of intruded VMs (VM1 and VM3)). Thus two VMs are additionally activated and the request is sent to all the VMs. All the VMs continually process the request, and AS achieved the agreement at the second round (the number of normal VMs (VM2, VM4, and VM5) $>$ the number of intruded VMs (VM1 and VM3)), that is, the request processing is succeeded, and a final response will be sent to the client. If AS cannot get the majority of normal VMs, new VMs will be activated, then the request will be processed at the third round. Note that, after sending the response to the client, all the VMs are initialized and the number of VMs becomes $n = 3$ again. Also, the security failure occurs when AS cannot make the agreement even at the m-th round.

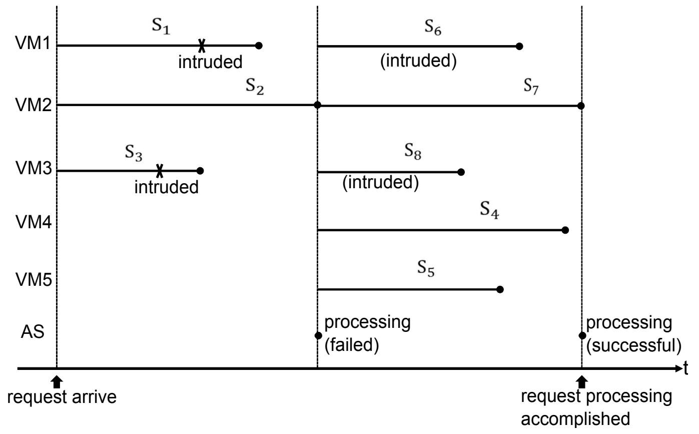

Figure 6.2 An illustrative behavior of the system for one request.

6.2 Survivability Analysis

6.2.1 Formulation of Success Probability

In this chapter, we focus on the success probability for one request under the environment where VMs may be intruded due to malicious attacks by exploiting security holes of system. The success probability is the probability that one request is processed within the maximum number of rounds under malicious attacks. For notational convenience, n_k is the number of VMs at the k-th round. Also f_k is the tolerance level at the k-th round. They are given by

$$n_k = n_1 + r(k-1), \quad k = 1, \ldots, m, \tag{6.1}$$

and

$$f_k = \lfloor \frac{n_k - 1}{2} \rfloor, \quad k = 1, \ldots, m, \tag{6.2}$$

respectively.

Suppose that the intrusion occurs according to a Poisson process with arrival rate γ (> 0) for each VM, namely, the failure rate of each VM is γ. Also let S be the processing time for a request in one VM which is an independent and identically distributed (i.i.d.) nonnegative random variable having the cumulative distribution function (c.d.f.) $G(t)$. Then the probability that a VM is intruded during it processes one request is given by

$$p_I = 1 - \int_0^\infty e^{-\gamma t} dG(t) = 1 - G^*(\gamma), \tag{6.3}$$

where $G^*(\gamma)$ is the Laplace-Stieltjes transform [44] of $G(t)$. For example, when $G(t)$ is a gamma distribution with mean a/b and standard deviation $\sqrt{a}/b$, the probability density function (p.d.f.) of the gamma distribution can be defined as follows,

$$g(t; a, b) = \frac{b^a t^{a-1} e^{-bt}}{\Gamma(a)}, \tag{6.4}$$

where

$$\Gamma(a) = \int_0^\infty e^{-t} t^{a-1} dt. \tag{6.5}$$

By applying the Laplace-Stieltjes transform, we obtain

$$G^*(\gamma) = \int_0^\infty e^{-\gamma t} g(t; a, b) dt = \left(\frac{b}{b+\gamma} \right)^a. \tag{6.6}$$

Thus in this case, we can compute the probability that a VM is intruded when processing one request by

$$p_I = 1 - \left(\frac{b}{b+\gamma} \right)^a. \tag{6.7}$$

Let N_k be the number of normal VMs at the end of the k-th round. Since the number of normal VMs is the number of VMs that are not intruded, the probability mass function (p.m.f.) of N_1 is given by

$$P(N_1 = x) = \binom{n_1}{x}(1-p_I)^x p_I^{n_1-x}, \quad \text{for } x = 0, \dots, n_1. \tag{6.8}$$

In the second and later round, the number of normal VMs depends on the number of normal VMs at the end of the previous round. Since r VMs are additionally activated when the number of normal VMs is less than $n_{k-1} - f_{k-1}$, the conditional p.m.f. is given by

$$P(N_k = x | N_{k-1} = y) = \tbinom{r+y}{x}(1-p_I)^x p_I^{r+y-x}, \tag{6.9}$$
$$\text{for } x = 0, \dots, r+y \text{ and } y = 0, \dots, n_{k-1} - f_{k-1} - 1.$$

According to the above probabilities, the request processing process can be described by a two dimensional discrete-time Markov chain (DTMC) $\{N_k; k \geq 1\}$ with state transition probability as in Eq. (6.9), which records the number of normal VMs at the end of one round in the case of disagreement, as well as the current round on which the request processing has been taken place. Note that the state space of the DTMC is defined by $\{0, \dots, n_k - f_k - 1\}$ because the agreement is succeeded when the number of normal VMs becomes greater than or equal to $n_k - f_k$ and that the state space depends on the number of rounds. Thus, each state of Markov chain is labeled as (x, k), indicating that there are x normal VMs at the end of the k-th round. For example, when considering that $m = 2$, the DTMC is depicted in Fig. 6.3.

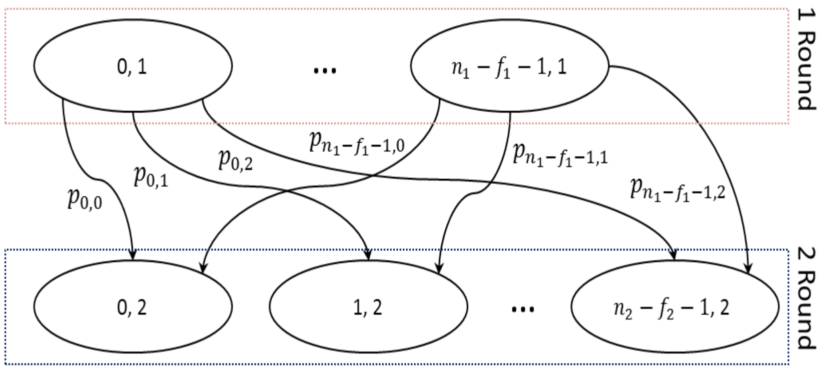

Figure 6.3 Markov model of request processing in the case of $m = 2$.

Let $\boldsymbol{\pi}_k$ be the probability vector at the end of the k-th round. Then we have

$$\boldsymbol{\pi}_k = \boldsymbol{\pi}_{k-1}\boldsymbol{P}_{k-1}, \tag{6.10}$$

where

$$\boldsymbol{P}_{k-1} = \begin{pmatrix} p_{0,0} & p_{0,1} & \cdots & p_{0,n_k-f_k-1} \\ p_{1,0} & p_{1,1} & \cdots & p_{1,n_k-f_k-1} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n_{k-1}-f_{k-1}-1,0} & p_{n_{k-1}-f_{k-1}-1,1} & \cdots & p_{n_{k-1}-f_{k-1}-1,n_k-f_k-1} \end{pmatrix} \tag{6.11}$$

and the (i, j)-element of transition matrix $\boldsymbol{P}_{k-1}$ is given by

$$\begin{aligned} p_{i,j} &= P(N_k = j | N_{k-1} = i) \\ &= \binom{r+i}{j}(1-p_I)^j p_I^{r+i-j}, \\ \text{for } i &= 0, \ldots, n_{k-1} - f_{k-1} - 1 \\ \text{and } j &= 0, \ldots, n_k - f_k - 1. \end{aligned} \tag{6.12}$$

Also the i-th element of the initial vector $\boldsymbol{\pi}_1$ is given by

$$\begin{aligned} [\boldsymbol{\pi}_1]_i &= \binom{n_1}{i}(1-p_I)^i p_I^{n_1-i}, \\ i &= 0, \ldots, n_1 - f_1 - 1. \end{aligned} \tag{6.13}$$

Since the state space changes at each round, it should be noted that $\boldsymbol{P}_k$ is not a square matrix. It is assumed that the maximum number of rounds is m, because of the lack of the resources on the physical machine to maintain many virtual machines. Therefore, in the worst-case scenario, the total number of VMs needed to increase to $n_1 + r(m-1)$ at the last round. The tolerance level of the system is $\lfloor \frac{n_1+r(m-1)-1}{2} \rfloor$, defined as f_m. That is, the system is able to tolerate maximally f_m Byzantine failures under malicious attacks. Moreover, the success probability for one request under malicious attack condition becomes

$$p_s = 1 - \boldsymbol{\pi}_m\mathbf{1} = 1 - \boldsymbol{\pi}_1\boldsymbol{P}_1\boldsymbol{P}_2\cdots\boldsymbol{P}_{m-1}\mathbf{1}, \tag{6.14}$$

where $\mathbf{1}$ is the column vector whose elements are all 1.

Table 6.1 Success probabilities with respect to n_1 and m in the case of $r = 1$.

	$m = 1$	$m = 2$	$m = 3$	$m = 4$
$n_1 = 2$	0.9994446372921	0.9999996144867	0.9999996144867	0.9999999994008
$n_1 = 3$	0.9999997686578	0.9999997686578	0.9999999996575	0.9999999996575
$n_1 = 4$	0.9999995374012	0.9999999995291	0.9999999995291	0.9999999999989
$n_1 = 5$	0.9999999997859	0.9999999997859	0.9999999999996	0.9999999999996

Table 6.2 Success probabilities with respect to n_1 and m in the case of $r = 2$.

	$m = 1$	$m = 2$	$m = 3$	$m = 4$
$n_1 = 2$	0.9994446372921	0.9999994603585	0.9999999992297	0.9999999999985
$n_1 = 3$	0.9999997686578	0.9999999997859	0.9999999999997	1.0000000000000
$n_1 = 4$	0.9999995374012	0.9999999994008	0.9999999999989	1.0000000000000
$n_1 = 5$	0.9999999997859	0.9999999999997	1.0000000000000	1.0000000000000

6.2.2 Survivability Metric

As mentioned before, the survivability is the capability of a system to provide its services in a timely manner even after intrusion and compromise occur. It has been quantified by using multidimensional Markov chains to consider simultaneous failures [45].

In this paper, we define the survivability metric as the conditional success probability for one request provided that the agreement is failed at the first round. It can be obtained as follows.

$$\tilde{p_s} = 1 - \tilde{\boldsymbol{\pi}}_1 \boldsymbol{P}_1 \boldsymbol{P}_2 \cdots \boldsymbol{P}_{m-1} \mathbf{1}, \tag{6.15}$$

where the i-th element of the initial vector $\tilde{\boldsymbol{\pi}}_1$ is given by

$$[\tilde{\boldsymbol{\pi}}_1]_i = \frac{\binom{n_1}{i}(1-p_I)^i p_I^{n_1-i}}{\sum_{j=0}^{n_1-f_1-1}\binom{n_1}{j}(1-p_I)^j p_I^{n_1-j}}, \quad i = 0, \ldots, n_1 - f_1 - 1. \tag{6.16}$$

6.3 Numerical Illustration

In this section, we evaluate the success probabilities for one request and discuss the survivability of VM-based intrusion tolerant system. Suppose that the failure (intrusion) rate of one VM is given by 10 times per hour, i.e., $\gamma = 1/360000[1/ms]$. In addition, the distribution for the processing time for a request in one VM is assumed to be the gamma distribution with mean 100 and standard deviation $50\sqrt{2}$. The p.d.f. of the gamma distribution is defined as

$$g(t; a, b) = \frac{b^a t^{a-1} e^{-bt}}{\Gamma(a)}, \tag{6.17}$$

where $a = 2$, and $b = 1/50$. Therefore, the probability that a VM is intruded when it processes one request can be computed by using Eq. (6.7).

Suppose that the number of initially activated VMs, n_1, varies from 2 to 5. If the agreement is failed due to some intruded VMs, AS tries to do it at the next round, and r VMs are additionally activated to continually process the request, in this chapter, we consider two situations of the number of VMs which is additionally activated at the next round, that is, $r = 1$ and $r = 2$. Besides, we consider the cases where the maximum number of rounds is $m = 2, 3, 4$.

6.3.1 Success Probability

This section illustrates the success probabilities for four different cases. Tables 6.1 and 6.2 show the success probabilities for the respective number of initially activated VMs and m rounds when $r = 1$ and $r = 2$, respectively. These tables indicate that the success probabilities become higher as the maximum number of rounds increases. However, as seen in Table 6.1, when $r = 1$, i.e., the number of additionally activated VMs is 1, there is the case where the success probability cannot be improved even if the number of rounds increases by one. For example, in the case of $r = 1$ and $n_1 = 2$, the success probability of $m = 2$ is the same as the probability of $m = 3$. Similarly, in the case of $r = 1$ and $n_1 = 3$, the success probabilities of $m = 1$ and $m = 2$ are identically 0.9999997686578. This is caused by the fact that there is the case where the intrusion tolerance level does not increase even if r increases by one in Eq. (6.2). On the other hand, by comparing Table 6.2 with Table 6.1, we find that the success probability monotonically increases when $r = 2$.

Next we investigate the success probabilities with respect to the number of initially activated VMs n_1. From Tables 6.1 and 6.2, we find that there are the cases where the success probability decreases even though n_1 increases. Remarkably, in the case of $r = 2$, the success probabilities with $n_1 = 4$ are less than those with $n_1 = 3$ for $m = 1, 2, 3$. This result is not intuitive.

Table 6.3 Success probabilities under Lau's and generalized schemes.

Scheme	Success probability	Success probability at the first round
Lau's scheme	0.9999999999991	0.9988895830120
Generalized scheme	0.9999999999997	0.9999997686578

The reasons are twofold; (i) the intrusion tolerance level with $n_1 = 3$ is the same as the one with $n_1 = 4$ in the case of $r = 2$; (ii) the risk of intrusion is higher as the number of VMs is larger. That is, in the case of $r = 2$, although the intrusion tolerance levels with $n_1 = 3$ and $n_1 = 4$ are the same, the intrusion risk of $n_1 = 4$ is higher than that of $n_1 = 3$. As a result, the success probability is degraded in the case of $n_1 = 4$. In fact, by looking at the success probabilities in the even numbers of initially activated VMs, $n_1 = 2, 4$, we find that they monotonically increase because the tolerance level surely increases. The same result appears in the odd numbers of initially activated VMs. As shown before, in the case of $r = 1$, since there are the cases where the tolerance level does not increase as the number of rounds increases, such degradation of success probabilities with respect to n_1 appears only in the cases of $m = 1$ and $m = 3$.

Finally we compare the success probabilities under Lau's scheme and the generalized scheme. In this experiment, the intrusion tolerance levels in both schemes are set as $f = 3$. Then under Lau's scheme, the numbers of initially and additionally activated VMs are $n = 4$ and $r = 3$, respectively. On the other hand, to achieve the same tolerance level at the maximum number of rounds $m = 3$, we set $n_1 = 3$ and $r = 2$ under the generalized scheme. Table 6.3 shows the success probabilities for one request under Lau's and generalized schemes. The last column in the table indicates the success probability at the first round. From the table, it can be seen that the success probabilities under Lau's and generalized schemes are almost the same, but the success probability at the first round under the generalized scheme is much higher than that under Lau's scheme. This implies that the generalized scheme more rarely goes to the second round, and thus the generalized scheme is expected to be more effective to reduce resource usage than Lau's scheme.

6.3.2 Survivability Analysis

In this section, we discuss the survivability of VM-based intrusion tolerant system under the generalized scheme. More precisely, we consider the survivability in terms of the conditional success probability $\tilde{p}_s$ under the environment where the agreement is failed at the first round. In general, in the intrusion tolerance scheme, the number of intruded VMs at the first round is unknown even when AS does not make the agreement at the first round, but the number of intruded VMs (alternatively, the number of normal VMs at the end of the first round) strongly affects the conditional success probability in the later rounds.

First we consider the case where the number of normal VMs at the end of the first round is known. Let N_1 be the number of normal VMs at the end of the first round. It has possible values $N_1 = 0, \dots, n_1 - f_1 - 1$ where f_1 is the tolerance level at the first round given by Eq. (6.2). Tables 6.4 and 6.5 present the conditional success probabilities for all the possible values of N_1 when $m = 2, 3, 4$ and $r = 1, 2$. Note that the conditional success probability is exactly 0 when there is no possible to obtain the majority of normal VMs at the end of the m-th round. Similar to the case of success probability, from Table 6.4, it is found that there are the cases where the conditional probabilities do not increase with respect to m in the case of $r = 1$. On the other hand, in the case of $r = 2$, the conditional success probabilities monotonically increase as the number of rounds becomes large. Also, there are the cases where the conditional success probabilities decrease with the number of initially activated VMs due to the tolerance level and the risk of intrusion.

Next we consider the case where the number of normal VMs is unknown. The conditional success probabilities can be computed by using the initial vector of Eq. (6.16). Tables 6.6 and 6.7 present the conditional success probabilities $\tilde{p}_s$ in the cases of $m = 2, 3, 4$ and $r = 1, 2$. Compared to the success probabilities in Tables 6.1 and 6.2, the conditional success probabilities become smaller, because the conditional success probability considers the situation where some VMs have already been intruded at the first round. Moreover, in the case of $r = 2$, the conditional success probability becomes higher as the number of rounds increases. This implies that it is important to check the conditional success probability as well as the success probability to enhance the survivability when the intrusion tolerance system is designed. On the other hand, we find that the conditional success probability does not increase mono-

Table 6.4 Conditional success probabilities with respect to n_1 and m in the case of $N_1 = 0, \ldots, n_1 - f_1 - 1$ and $r = 1$.

		$m = 2$	$m = 3$	$m = 4$
$n_1 = 2$	$N_1 = 1$	0.9994446372921	0.9994446372921	0.9999991522305
	$N_1 = 0$	0	0	0.9983348369882
$n_1 = 3$	$N_1 = 1$	0	0.9986121714787	0.9986121714787
	$N_1 = 0$	0	0	0
$n_1 = 4$	$N_1 = 2$	0.9991670716093	0.9991670716093	0.9999981512100
	$N_1 = 1$	0	0	0.9975032955591
	$N_1 = 0$	0	0	0
$n_1 = 5$	$N_1 = 2$	0	0.9980575795191	0.9980575795191
	$N_1 = 1$	0	0	0
	$N_1 = 0$	0	0	0

Table 6.5 Conditional success probabilities with respect to n_1 and m in the case of $N_1 = 0, \ldots, n_1 - f_1 - 1$ and $r = 2$.

		$m = 2$	$m = 3$	$m = 4$
$n_1 = 2$	$N_1 = 1$	0.9991670716093	0.9999988440168	0.9999999977972
	$N_1 = 0$	0	0.9983348369882	0.9999966884465
$n_1 = 3$	$N_1 = 1$	0.9991670716093	0.9999988440168	0.9999999977972
	$N_1 = 0$	0	0.9983348369882	0.9999966884465
$n_1 = 4$	$N_1 = 2$	0.9988895830120	0.9999979969749	0.9999999952122
	$N_1 = 1$	0	0.9977803990498	0.9999943810137
	$N_1 = 0$	0	0	0.9966724467444
$n_1 = 5$	$N_1 = 2$	0.9988895830120	0.9999979969749	0.9999999952122
	$N_1 = 1$	0	0.9977803990498	0.9999943810137
	$N_1 = 0$	0	0	0.9966724467444

tonically even if we focus only on the even numbers of n_1. For example, in the case of $r = 2$, the conditional success probabilities with $n_1 = 4$ are smaller than those with $n_1 = 2$. This result is remarkable in the case of survivability analysis of intrusion tolerance system, and also the same result is found when we focus on the odd numbers of n_1. When the agreement is failed at the first round, the number of intruded VMs is proportional to the number of initially activated VMs. In addition, under the situation where the number of intruded

Table 6.6 Conditional success probabilities with respect to n_1 and m in the case of $r = 1$.

	$m = 2$	$m = 3$	$m = 4$
$n_1 = 2$	0.9993058351766	0.9993058351766	0.9999989210916
$n_1 = 3$	0	0.9985197095296	0.9985197095296
$n_1 = 4$	0.9989820492377	0.9989820492377	0.9999976763937
$n_1 = 5$	0	0.9979189623371	0.9979189623371

Table 6.7 Conditional success probabilities with respect to n_1 and m in the case of $r = 2$.

	$m = 2$	$m = 3$	$m = 4$
$n_1 = 2$	0.9990283080419	0.9999986129208	0.9999999973376
$n_1 = 3$	0.9990745582816	0.9999986899457	0.9999999974908
$n_1 = 4$	0.9987046120248	0.9999975734967	0.9999999941299
$n_1 = 5$	0.9987508502756	0.9999976812805	0.9999999944068

Table 6.8 Conditional success probabilities $\tilde{p_s}$ in Lau's vs. generalized schemes under the failure of first round.

Scheme	Conditional success probability
Lau's scheme	0.9999999991600
Generalized scheme	0.9999986899457

VMs is large, it is difficult to make the agreement even if several VMs are additionally activated. That is, from the viewpoint of survivability, the small number of initially activated VMs is better.

Finally, we compare Lau's and generalized schemes in terms of conditional success probability. According to the same setting in the previous section, we compute the conditional success probabilities under Lau's and generalized schemes (see Table 6.8). From the table, the conditional success probability under Lau's scheme is higher than that under the generalized scheme. It is the evidence that the ability to tolerate the intrusion at the second round is high under Lau's scheme. Therefore, it is important to choose the scheme based on the purpose of system.

Moreover, we investigate the sensitivity of r on the success probability p_s and conditional success probability $\tilde{p_s}$. Table 6.9 shows the success probabil-

Table 6.9 Success probabilities and conditional success probabilities with respect to $r = 1, 2, 3, 4, 5$ in the case of $n_1 = 3$ and $m = 2$.

	Success probability p_s	Conditional success probability $\tilde{p_s}$
$r = 1$	0.9999997686578	0
$r = 2$	0.9999999997859	0.9990745582816
$r = 3$	0.9999999997217	0.9987970953771
$r = 4$	0.9999999999998	0.9999991264021
$r = 5$	0.9999999999997	0.9999987155386

ities and conditional success probabilities with respect to $r = 1, 2, 3, 4, 5$ in the case of $n_1 = 3$ and $m = 2$. Similar to the results on n_1, both success probability and conditional success probability do not increase monotonically with respect to r. When we focus on the even numbers of r, they monotonically increase due to the characteristics of tolerance level. In the case of the odd numbers of r, we can conclude the same remark.

6.3.3 Parameter Effects

Finally, we consider the effects of failure rate of VM on the success probability and conditional success probability in the situation where the number of initially activated VMs, n_1, is 3, and the maximum number of rounds, m, is 3. For the activation number of VMs, we assume that $r = 1$ and $r = 2$, respectively as previously mentioned. Also we suppose that the failure rate of one VM, γ, varies from 1 to 100 times per hour, i.e., γ varies from $1/3600000[1/ms]$ to $1/36000[1/ms]$. The numerical results are depicted from Figs. 6.4 to 6.7.

Figs. 6.4 and 6.5 show the effects of the failure of VM on the success probabilities and conditional success probabilities in the case of $r = 1$ and $r = 2$, respectively. From Fig. 6.4, we can see that the success probability in the case of $r = 1$ is quite high, and almost equal to the success probability in the case of $r = 2$ when the failure rate of VM is less than $\lfloor 0.000014 \times 3600000 \rfloor = 50$ times per hour. When the failure rate increases and becomes higher than 50 times per hour, the success probability in the case of $r = 2$ is almost unchanged and sufficiently high. However, the success probability in the case of $r = 1$ decreases as the failure rate increases. In Fig. 6.5, we find that the conditional success probability in the case of $r = 1$ decreases rapidly when

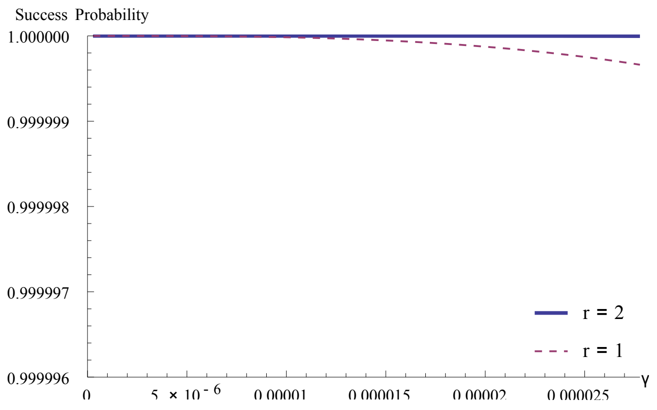

Figure 6.4 Effects of VM failure rate on the success probabilities in the case of $r = 1$ and $r = 2$.

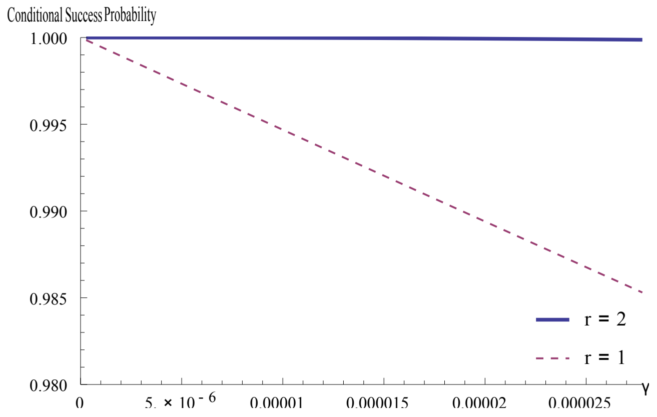

Figure 6.5 Effects of VM failure rate on the conditional success probabilities in the case of $r = 1$ and $r = 2$.

the failure rate increases. On the other hand, the conditional success probability in the case of $r = 2$ does not change much. These figures imply that any change in the failure rate of VM has a great impact on the success probability for one request in the case of $r = 1$. However, activating 2 VMs at the next round has a large effect on the improvement of the success probability which have been validated by comparing Table 6.1 with Table 6.2, or Table 6.6 with Table 6.7.

The effects of the failure rate of VM on the success probability and conditional success probability in both of two cases are shown in Figs. 6.6 and 6.7. From Fig. 6.6, it is observed that the conditional success probability decreases rapidly when the failure rate increases. This indicates that any change in the failure rate of VM also has a great impact on the conditional success probability $\tilde{p_s}$. Therefore, to improve the conditional success probability, the most

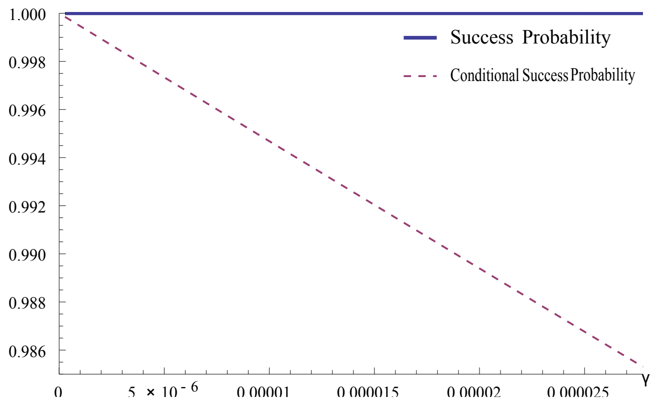

Figure 6.6 Effects of VM failure rate on the success probability and conditional success probability in the case of $r = 1$.

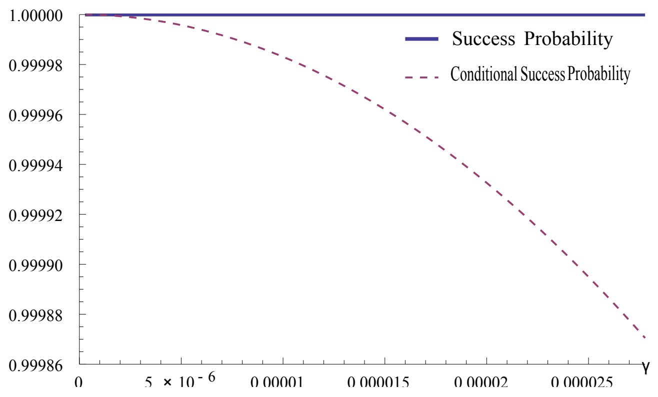

Figure 6.7 Effects of VM failure rate on the success probability and conditional success probability in the case of $r = 2$.

efficient way is to decrease the failure rate of VM. However, the curve for the success probability implies that any change in the failure rate has less effect on the success probability of VM-based intrusion tolerant system. The same conclusions can also be obtained from Fig. 6.7. As seen in Fig. 6.7, it is found that both of the success probability and conditional success probability in the case of $r = 2$ become high by the comparison to Fig. 6.6 with $r = 1$. Moreover, the main difference between two figures is that, the conditional success probability in the case of $r = 2$ decreases slowly as the failure rate increases, compared to the curve for the conditional success probability in the case of $r = 1$.

6.4 Conclusion

In this chapter, we have dealt with survivability quantification of VM-based intrusion tolerant system proposed by Lau et al. [21]. Concretely, we presented the generalized scheme which is extended from Lau's scheme, and built a DTMC-based stochastic model from the system behavior. We then formulated the success probability for one request, and the conditional success probability provided that the agreement is failed at the first round as a survivability metric. In numerical experiments, we have investigated the characteristics of success probability and conditional success probability under both Lau's and generalized schemes. As a result, their performance cannot be evaluated only by the tolerance level. In addition, we have investigated the situation where the success probability and the conditional success probability are degraded even though the number of activated VMs increases. Also, we have found that the generalized scheme is superior to the reduction of resource usage and the enhancement of success probability, and that Lau's scheme has high survivability. In addition, our numerical experiments indicate that the best design of the VM-based intrusion tolerant system is the initial number of VMs (n_1) is 3, the activation number of VMs (r) is 2, and the maximum number of rounds (m) is given according to the system resources.

CHAPTER 7

CONCLUSIONS

Dependability is an all-encompassing definition for reliability, availability, safety and security, and is required in computer applications such as safety-critical control systems for road vehicles, airplanes and medical devices, and business-critical systems for e-commerce and financial transactions. In general, two commonly-used types of system designs; component redundancy and environmental redundancy are developed to assure high dependability of systems. However, redundancy increases not only the complexity of a system, but also the complexity of associated problems such as common-mode error. Thus, in order to detect the optimal design of systems, model-based analysis is important in the system design. One of the advantages of model-based analysis is sensitivity analysis, which is a method to detect the most important factors to the system dependability and plays an important role in the optimization of system in the design phase. In this thesis, we have focused on the design sensitivity (i.e., component importance) for virtualized systems (in Chapters 3 and 4) and real-time computing systems (in Chapter 5). In particular, we considered the environmental sensitivity that measures the effect of environmental changes on system dependability, that is, the survivability of VM-based intrusion tolerant systems (in Chapter 6).

Concretely, in Chapter 3, we have revisited the hybrid model (FT and CTMC) for non-virtualized and virtualized system design in [18], and proposed a generalized method to the availability importance analysis of components. The proposed method is based on both the aggregation techniques of CTMC-based availability models [35] and the importance measures of components with respect to failure and repair rates [33]. Although the presented component analysis is simple, it is quite helpful in the system design phase to ensure the system availability. In Chapter 4, we focused on the CTMC model of live migration, and introduced the Markov-based component-wise sensitivity analysis to evaluate the component importance in the context of live migration without using structure function. According to the component importance analysis, the critical components in non-virtualized and virtualized system designs are detected.

In Chapter 5, we have quantitatively evaluated the component importance measures of a real-time computing system [11] where CCFs occur. Concretely, we evaluated three kinds of importance measures in the case of system without CCFs by the common method using hybrid model and structure function, and in the case of system with CCFs using Markov-based component-wise sensitivity analysis, respectively. When considering the system without CCFs, the above two methods can applied to compute the important measures. However, the common method does not hold in the case where components are dependent. In such case, the Markov-based component-wise sensitivity analysis is adopted. In numerical experiments, we illustrated the quantitative importance measures of all components, and ranked the components according to distinct importance measures. Also, the effect of CCFs is considered. Our numerical experiments indicate that the CCFs can affect not only the reliabilities of components and system, but also their importance and importance ranking.

Moreover, in Chapter 6, we have dealt with survivability quantification of VM-based intrusion tolerant system proposed by Lau et al. [21]. Concretely, based on Lau's scheme, we presented the generalized scheme, and built a DTMC-based stochastic model from the system behavior. We then formulated the success probability for one request, and the conditional success probability provided that the agreement is failed at the first round as a survivability metric. In numerical experiments, we have investigated the characteristics of success probability and conditional success probability under both Lau's and generalized schemes. As a result, their performance cannot be evaluated only by the tolerance level. Also, we have investigated the situation where the success probability and the conditional success probability are degraded even though the number of activated VMs increases. Moreover, we have found that the generalized scheme is superior to the reduction of resource usage and the enhancement of success probability, and that Lau's scheme has high survivability.

Generally speaking, the main contribution of this thesis is twofold. From the viewpoint of model analysis, we have presented the generalized methods of importance analysis to help detecting the critical components in the computer systems even in the case that system components are failure dependent. In such cases, the common method of computing importance measures using hybrid models and structure function cannot be applied. Also, a generalized scheme for VM-based intrusion tolerant system are proposed, which is superior to the reduction of resource usage and the enhancement of success probability, in contrast to the existing schemes; From the viewpoint of system applications, we have detected the critical components for computer systems and obtained the best system design by using proposed methods.

In the system design, the engineer is often faced with the task of developing a design that will achieve the desired reliability of the system while performing all of the intended functions of system at a minimum cost. This involves a balancing act of determining how to allocate system resources (e.g., reliability) to the components in the system so the system will meet its reliability goal while at the same time ensuring that the system meets all of the other associated performance specifications [50]. The generalized methods of importance analysis are significant and effective ways to solve above system resource allocation problem and reliability and cost optimization problem. For example, based on component importance analysis, the most critical components can be obtained. If the reliability of the system is to be improved, then the efforts can best be concentrated on improving the reliability of the critical component first.

It is worth noting that there are still some challenges to be addressed in our future work. First, the error appearing in the calculation of component importance in large Markov chains using proposed method will be reduced by improving the algorithms. Second, investigating the importance of common root cause event (CRCE) in the system where CCFs occur to find efficient defense strategies against CCFs will be considered. Finally, the optimization policies for maximizing the system dependability improvements based on the obtained importance and relative ranking of components will be studied in our future work.

APPENDIX A
UNIFORMIZATION-BASED ALGORITHMS

Ramesh and Trivedi [46] derived the integral form of solution $\boldsymbol{s}(t,\theta)$ as a convolution:

$$\boldsymbol{s}(t,\theta) = \boldsymbol{\pi}_0 \int_0^t \exp(\boldsymbol{Q}s)\boldsymbol{S}(\theta)\exp(\boldsymbol{Q}(t-s))ds. \tag{A.1}$$

For the above integral form, we consider the uniformization technique which reduce a CTMC to a discrete-time Markov chain (DTMC) subordinated to a Poisson process [47], and present two efficient algorithms based on uniformization.

A.0.0.1 Uniformization-based Algorithms In fact, it is known that the uniformization is an efficient method to compute transient solutions of CTMCs, and is superior to general-purpose differential equation methods in terms of scalability and accuracy [48]. The formula of the uniformization is given by

$$\exp(\boldsymbol{Q}t) = \sum_{k=0}^{\infty} e^{-qt}\frac{(qt)^k}{k!}\boldsymbol{P}^k, \tag{A.2}$$

where q is a randomization parameter satisfying $q = \sup_{i\in S}|q_{ii}|$ and the transition probability matrix of DTMC is $\boldsymbol{P} = \mathbf{I} + \mathbf{Q}/q$. By truncating the infinite sum with an approximate integer, we have

$$\exp(\boldsymbol{Q}t) \approx \sum_{k=0}^{U} e^{-qt}\frac{(qt)^k}{k!}\boldsymbol{P}^k, \tag{A.3}$$

where the upper (right) bound U is determined as

$$U = \inf\left\{u \geq 0; \sum_{k=0}^{u} e^{-qt}\frac{(qt)^k}{k!} \geq 1-\varepsilon\right\}. \tag{A.4}$$

In the above equation, ε is the error tolerance.

This section proposes the uniformization-based computation for the sensitivity function. By applying the uniformization technique, the convolution integral of Eq. (A.1) can be reduced to

$$s_\theta(t) \approx \tilde{s}_\theta(t) = \boldsymbol{\pi}(0)\frac{1}{q}\sum_{k=0}^{U}\mathrm{Poi}(k+1;qt)\sum_{u=0}^{k}\mathbf{P}^u\mathbf{A}\boldsymbol{P}^{k-u}\mathbf{r}, \tag{A.5}$$

where $\mathrm{Poi}(k;qt) = e^{-qt}\frac{qt^k}{k!}$. Then $\tilde{s}_\theta(t)$ can be obtained by the following recurrence formula:

$$\boldsymbol{f}(0) = \boldsymbol{\pi}(0), \tag{A.6}$$

$$\boldsymbol{f}(k) = \boldsymbol{f}(k-1)\boldsymbol{P}, \quad \text{for } k = 1,2,\ldots,U, \tag{A.7}$$

$$\boldsymbol{b}(U+1) = \mathbf{0}, \tag{A.8}$$

$$\boldsymbol{b}(k) = \boldsymbol{P}\boldsymbol{b}(k+1) + \mathrm{Poi}(k+1;qt)\mathbf{r}, \quad \text{for } k = U, U-1, \ldots, 0, \tag{A.9}$$

$$\tilde{s}_\theta(t) = \frac{1}{q}\sum_{k=0}^{U}\boldsymbol{f}(k)\mathbf{A}\boldsymbol{b}(k). \tag{A.10}$$

Therefore, we obtain the computation algorithm for the sensitivity function as follows.

```
1: $\boldsymbol{f}(0) \leftarrow \boldsymbol{\pi}(0)$
2: for $k = 1 : U$ do
3:     $\boldsymbol{f}(k) \leftarrow \boldsymbol{f}(k-1)\boldsymbol{P}$
4: end for
5: $\boldsymbol{b}(U+1) \leftarrow \mathbf{0}$
6: for $k = U : 0$ do
7:     $\boldsymbol{b}(k) \leftarrow \boldsymbol{P}\boldsymbol{b}(k+1) + \mathrm{Poi}(k+1;qt)\mathbf{r}$
8: end for
9: $s \leftarrow 0$
```

```
10: for $k = 0 : U$ do
11:     $s \leftarrow s + \boldsymbol{f}(k)\boldsymbol{A}\boldsymbol{b}(k)$
12: end for
```

The time complexity of the above algorithm is given by $O(qtn^2)$, since U is proportional to qt. The advantage of this method is to omit the computation of $\boldsymbol{f}(\cdot)$ and $\boldsymbol{b}(\cdot)$ when we wish to compute the sensitivity with respect to another parameter. That is, if v is the number of parameters whose the sensitivity functions are computed, the time complexity becomes $O(qtn^2v)$.

The second uniformization algorithm is derived from the following matrix:

$$\boldsymbol{Q}' = \begin{pmatrix} \mathbf{Q} & \mathbf{A} \\ \mathbf{O} & \mathbf{Q} \end{pmatrix}. \tag{A.11}$$

From the elementary analysis of matrix exponential, we have

$$\exp(\boldsymbol{Q}'t) = \begin{pmatrix} \exp(\boldsymbol{Q}t) & \int_0^t \exp(\boldsymbol{Q}s)\mathbf{A}\exp(\boldsymbol{Q}(t-s))ds \\ \mathbf{O} & \exp(\boldsymbol{Q}t) \end{pmatrix}. \tag{A.12}$$

The right top element equals the desired sensitivity function. Also, for any matrix $\boldsymbol{A}$, we can apply the uniformization to $\boldsymbol{Q}'$ with the randomization parameter $q = \sup_{i \in S} |q_{ii}|$. Then we obtain the following algorithm for computing the sensitivity function:

```
1: $\boldsymbol{f}_2(0) \leftarrow \mathbf{0}$
2: $\boldsymbol{f}_1(0) \leftarrow \boldsymbol{\pi}(0)$
3: for $k = 1 : U$ do
4:     $\boldsymbol{f}_2(k) \leftarrow \boldsymbol{f}_1(k-1)\boldsymbol{A}$
5:     $\boldsymbol{f}_1(k) \leftarrow \boldsymbol{f}_1(k-1)\boldsymbol{P}$
6: end for
7: $s \leftarrow 0$
8: $\boldsymbol{\xi} \leftarrow 0$
9: for $k = 0 : U$ do
10:     $s \leftarrow s + \mathrm{Poi}(k; qt)\boldsymbol{f}_2(k)\mathbf{r}$
11:     $\boldsymbol{\xi} \leftarrow \boldsymbol{\xi} + \mathrm{Poi}(k; qt)\boldsymbol{f}_1(k)$
12: end for
```

The outputs s and $\boldsymbol{\xi}$ of the above algorithm are the sensitivity function and the probability vector $\boldsymbol{\pi}(t)$, respectively. The time complexity of the algorithm is also given by $O(qtn^2)$. In the above algorithm, since the probability vector $\boldsymbol{\pi}(t)$ can also be computed, the algorithm is appropriate to get the temporal behavior of the sensitivity function $s_\theta(t)$.

APPENDIX B
SENSITIVITY ESTIMATION METHOD

Suppose that the system consists of K components. Let I_S and I_k be the performance indices of system and component k, respectively. This thesis assumes that the performance indices can be computed by an MRM with instantaneous rewards. Also we define the reward vectors corresponding to I_S and I_k as $\boldsymbol{r}_S$ and $\boldsymbol{r}_k$, respectively. Without loss of generality, we assume

$$I_S = \boldsymbol{\pi}\boldsymbol{r}_S, \tag{B.1}$$
$$I_k = \boldsymbol{\pi}\boldsymbol{r}_k, \quad \text{for } k = 1, \ldots, K, \tag{B.2}$$

where $\boldsymbol{\pi}$ is a state probability vector of the underlying CTMC at arbitrary time point. In the case of transient measure, $\boldsymbol{\pi}$ should be $\boldsymbol{\pi}(t)$. On the other hand, $\boldsymbol{\pi}$ is the steady-state probability vector when the performance index is a steady-state measure $\boldsymbol{\pi}_{ss}$.

Then we estimate the sensitivities of system performance index with respect to component performance indices:

$$\frac{\partial I_S}{\partial I_k}, \quad \text{for } k = 1, \ldots, K. \tag{B.3}$$

As mentioned before, except for the case where I_S is explicitly given by a function of $I_1, \ldots, I_k$, the above sensitivities cannot be obtained analytically. Then we consider the estimation from the parametric sensitivities.

Consider the sensitivity of system performance index with respect to model parameter vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)$. According to the chain rule in partial differentiation, we have, for $k = 1, \ldots, K$ and $j = 1, \ldots, m$;

$$\frac{\partial I_S}{\partial \theta_j} = \sum_{k=1}^{K} \frac{\partial I_S}{\partial I_k} \frac{\partial I_k}{\partial \theta_j} + \delta_j. \tag{B.4}$$

In the equation, δ_j means the deviation of I_S with respect to θ_j which are not correlated to the deviations of $I_1, \ldots, I_K$. Equation (B.4) can be rewritten as

$$\boldsymbol{z} = \boldsymbol{J}\boldsymbol{u} + \boldsymbol{\delta}, \tag{B.5}$$

where

$$\boldsymbol{z} = \begin{pmatrix} \frac{\partial I_S}{\partial \theta_1} \\ \vdots \\ \frac{\partial I_S}{\partial \theta_m} \end{pmatrix}, \quad \boldsymbol{u} = \begin{pmatrix} \frac{\partial I_S}{\partial I_1} \\ \vdots \\ \frac{\partial I_S}{\partial I_K} \end{pmatrix}, \quad \boldsymbol{\delta} = \begin{pmatrix} \delta_1 \\ \vdots \\ \delta_m \end{pmatrix}, \tag{B.6}$$

$$\boldsymbol{J} = \begin{pmatrix} \frac{\partial I_1}{\partial \theta_1} & \cdots & \frac{\partial I_K}{\partial \theta_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial I_1}{\partial \theta_m} & \cdots & \frac{\partial I_K}{\partial \theta_m} \end{pmatrix}. \tag{B.7}$$

In the above, the matrix $\boldsymbol{J}$ and the vector $\boldsymbol{z}$ can be computed from the parametric sensitivity approach described in Sec. 2.3.1. The problem is reduced to solving the above linear equation with respect to $\boldsymbol{u}$. However, since the vector $\boldsymbol{\delta}$ is also undetermined, we should consider how to treat the vector $\boldsymbol{\delta}$.

If I_S is a function of only $I_1, \ldots, I_K$, then the vector $\boldsymbol{\delta}$ becomes $\mathbf{0}$ theoretically. On the other hand, even if I_S has non-zero $\boldsymbol{\delta}$, it is better to estimate the deviation of $\boldsymbol{\delta}$ as small as possible. For example, we suppose that the system performance index is given by

$$I_S = I_1 + I_2 - I_1 I_2. \tag{B.8}$$

In this case, it is natural to take the following sensitivities;

$$\frac{\partial I_S}{\partial I_1} = 1 - I_2, \quad \frac{\partial I_S}{\partial I_2} = 1 - I_1. \tag{B.9}$$

However, if $I_1 I_2$ is regarded as one variable, namely, $I_3 = I_1 I_2$, then the sensitivities are

$$\frac{\partial I_S}{\partial I_1} = 1, \quad \frac{\partial I_S}{\partial I_2} = 1, \quad \frac{\partial I_S}{\partial I_3} = -1. \tag{B.10}$$

Although Eqs. (B.9) and (B.10) are quite different, they provides the same result on the parametric sensitivity. That is, based on Eq. (B.9), we have

$$\begin{aligned} \frac{\partial I_S}{\partial \theta} &= \frac{\partial I_S}{\partial I_1}\frac{\partial I_1}{\partial \theta} + \frac{\partial I_S}{\partial I_2}\frac{\partial I_2}{\partial \theta} \\ &= (1 - I_2)\frac{\partial I_1}{\partial \theta} + (1 - I_1)\frac{\partial I_2}{\partial \theta}. \end{aligned} \tag{B.11}$$

When Eq. (B.10) is applied, the parametric sensitivity is expressed by

$$\begin{aligned} \frac{\partial I_S}{\partial \theta} &= \frac{\partial I_S}{\partial I_1}\frac{\partial I_1}{\partial \theta} + \frac{\partial I_S}{\partial I_2}\frac{\partial I_2}{\partial \theta} + \frac{\partial I_S}{\partial I_3}\frac{\partial I_3}{\partial \theta} \\ &= \frac{\partial I_1}{\partial \theta} + \frac{\partial I_2}{\partial \theta} - \frac{\partial I_3}{\partial \theta}. \end{aligned} \tag{B.12}$$

Therefore, from the mathematical point of view, both are correct. However, the farmer is a better representation for the relationship between the system performance index and component indices, since $\partial I_S/\partial I_3$ can be expressed by $\partial I_S/\partial I_1$ and $\partial I_S/\partial I_2$. In other words, it is important to explain the deviation of I_S as much as possible by using only the deviations with I_1 and I_2. In Eq. (B.12), the last term corresponds to δ in Eq. (B.4). Therefore, it is better to take the estimates of $\boldsymbol{u}$ so that $\boldsymbol{\delta}$ becomes small.

Based on the above insight, we formulate the following mathematical programming:

$$\min_{\boldsymbol{u}} \|\boldsymbol{\delta}\|_2^2, \tag{B.13}$$

$$\text{s.t.} \quad \boldsymbol{z} = \boldsymbol{J}\boldsymbol{u} + \boldsymbol{\delta}, \tag{B.14}$$

where $\|\boldsymbol{\delta}\|_2$ is a 2-norm of vector $\boldsymbol{\delta}$. The problem is further reduced to the following least square problem:

$$\min_{\boldsymbol{u}} \|\boldsymbol{z} - \boldsymbol{J}\boldsymbol{u}\|_2^2. \tag{B.15}$$

We can apply the several methods to solve the least square problem. The simplest approach is to solve the normal equation [49]:

$$(\boldsymbol{J}^T\boldsymbol{J})\boldsymbol{u} = \boldsymbol{J}^T\boldsymbol{z}, \tag{B.16}$$

where T is the transpose operator. Then the estimates of sensitivities are given by

$$\boldsymbol{u} = (\boldsymbol{J}^T \boldsymbol{J})^{-1} \boldsymbol{J}^T \boldsymbol{z}. \tag{B.17}$$

REFERENCES

1. H. Pham, “Optimal design of k-out-of-n redundant systems,” *Microelectronics Reliability*, vol. 32, no. 1, pp. 119–126, 1992.
2. A. Carzaniga, A. Gorla, and M. Pezzè, “Handling software faults with redundancy,” *Architecting Dependable Systems VI*, Springer Berlin Heidelberg, pp. 148–171, 2009.
3. B. Furht, “Cloud computing fundamentals,” in *Handbook of Cloud Computing*. by B. Furht and E. Armando, Eds. Springer. pp. 3–19, 2010.
4. Clark, Christopher and Fraser, Keir and Hand, Steven and Hansen, Jacob Gorm and Jul, Eric and Limpach, Christian and Pratt, Ian and Warfield, Andrew, “A new look at the statistical model identification,” in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, USENIX Association, vol. 2, pp. 273–286, 2005.
5. A. Agbaria and R. Friedman, “Overcoming Byzantine failures using checkpointing,” *Tech rep*, no. UILU-ENG-03-2228 (CRHC-03-14), Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, 2003.
6. W. Park and C. Park, “Data firewall: a TPM-based security framework for protecting data in thick client mobile environment,” *Journal of Computing Science and Engineering*, vol. 5, no. 4, pp. 331–337, 2011.
7. V. Stavridou, et al., “Intrusion tolerant software architectures,” in *Proceedings of DARPA information survivability conference and exposition (DISCEX II’01)*, vol. 2, 2001.
8. F. B. Schneider, “Implementing fault-tolerant services using the state machine approach: a tutorial,” *ACM Computing Surveys*, vol. 22, no. 4, pp. 299–319, 1990.
9. T. Brooks, C. Caicedo, and J. Park, “Security vulnerability analysis in virtualized computing environments,” *International Journal of Intelligent Computing Research (IJICR)*, Vol. 3, Nos. 1/2, pp. 277–291, 2012.
10. P. A. Laplante, *Real-time systems design and analysis-an engineer’s handbook*, 2^{nd} Ed., IEEE Press, 1997.
11. R. Fricks, K. S. Trivedi, “Modeling failure dependencies in reliability analysis using stochastic Petri nets,” in *Proceedings of the 11th European Simulation Multi-conference (ESM ’97)*, Istanbul: Turkey ACM Press, 1–4 June 1997.
12. S. Kundu and R. Rangaswami and K. Dutta and M. Zhao. Application performance modeling in a virtualized environment, in *Proceedings of the 16th IEEE International Symposium on High-Performance Computer Architecture*. IEEE Computer Society, pp. 10 pages, 2010.
13. H. Okamura, K. Shigeoka, K. Yamasaki, T. Dohi and H. Kihara. Performance evaluation of cloud computing in PaaS environments, in *Supplemental Proceedings of 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN2012)*, 36:122–127, 2012.

14. B. Cully, G. Lefebvre, D. Meyer, M. Feeley, N. Hutchinson and A. Warfield. Remus: high availability via asynchronous virtual machine replication, in *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, USENIX Association, pp. 161–174, 2008.

15. E. Farr, R. Harper, L. Spainhower and J. Xenidis. A Case for High Availability in a Virtualized Environment (HAVEN), in *Proceedings of the 2008 Third International Conference on Availability, Reliability and Security (ARES'08)*. IEEE Computer Society, pp. 675–682, 2008.

16. M. T. H. Myint and T. Thein. Availability improvement in virtualized multiple servers with software rejuvenation and virtualization, in *Proceedings of 4th International Conference on Secure Software Integration and Reliability Improvement*. IEEE Computer Society, pp. 156–162, 2010.

17. K. V. Vishwanath and N. Nagappan. Characterizing cloud computing hardware reliability, in *Proceedings of the first ACM Symposium on Cloud Computing (SoCC'10)*. ACM, pp. 193–204, 2010.

18. D. S. Kim and F. Machida and K. S. Trivedi, "Availability modeling and analysis of a virtualized system," in *Proceedings of the 15th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC-2009)*, IEEE Computer Society, pp. 365–371, 2009.

19. R. D. S. Matos, P. R. M. Maciel, F. Machida, D. S. Kim and K. S. Trivedi, "Sensitivity analysis of server virtualized system availability," *IEEE Transactions on Reliability*, IEEE, pp. 994–1006, 2012.

20. V. S. Junior, L. C.Lung, M. Correia, JD. S. Fraga, and J. Lau, "Intrusion tolerant services through virtualization: a shared memory approach," *(AINA'10)*, 2010.

21. J. Lau, L. Barreto, and JD. S. Fraga, "An infrastructure based in virtualization for intrusion tolerant services," *2012 IEEE 19th International Conference on Web Services (ICWS)*, pp. 170–177, 2012.

22. R. Fricks, K. S. Trivedi, "Importance analysis with Markov chains," *Reliability and Maintainability Symposium, 2003, Annual*, pp. 89–95, 2003.

23. Z. J. Pan, Y. Nonaka, "Importance analysis for the systems with common cause failures," *Reliability Engineering and System Safety*, vol. 50, pp. 297–300, 1995.

24. C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*. USENIX Association, pp. 273–286, 2005.

25. N. G. Leveson, *Safeware: system safety and computers*, Addison-Wesley, Reading, MA, 1995.

26. K. Lampka, M. Siegle, and M. Walter, "An easy-to-use, efficient tool-chain to analyze the availability of telecommunication equipment," in *Formal Methods: Applications and Technology*. vol. 4346, pp. 35–50, 2007.

27. A. Reibman, K. S. Trivedi, "Transient analysis of cumulative measures of Markov model behavior," *Stochastic Models 5*, pp. 683–710, 1989.

28. D. M. Cohen, D. P. Heyman, A. Rabinovitch, and D. Brown, "Parallel implementation of the GTH algorithm for Markov chains," in *Computations with Markov chains: Proceedings of the 2nd international workshop on the numerical solution of Markov chains (35)*, edited by W. J. Stewartpp, pp. 594–596, Springer Science & Business Media, NY, 2012.

29. A. Reibman, R. Smith, and K. S. Trivedi, "Markov and Markov reward model transient analysis: An overview of numerical approaches," *European J. of Operations Research*, Vol. 40, pp. 257–267, 1989.

30. J. K. Ravalico, H. R. Maier, G. C. Dandy, J. P. Norton and B. F. W. Croke, "A comparison of sensitivity analysis techniques for complex models for environmental management," *International Congress on Modeling and Simulation (MODSIM 2005)*, pp. 2533–2539, 2005.

31. V. Castelli and R. E. Harper and P. Heidelberger and S. W. Hunter and K. S. Trivedi and K. Vaidyanathan and W. P. Zeggert, "Proactive management of software aging," *IBM J. Research & Development*, vol. 45, pp. 311–332, 2001.

32. K. S. Trivedi, "Probability and Statistics with Reliability, Queueing, and Computer Sciences Applications," *John Wiley & Sons*, 2nd, 2001.

33. C. R. Cassady and E. A. Pohl and S. Jin, "Managing availability improvement efforts with importance measures and optimization," *IMA Journal of Management Mathematics*, vol. 15, pp. 161–174, 2004.

34. Z. W. Birnbaum, "On the importance of different components in a multicomponent system," *Academic Press*, pp. 581–592, 1969.

35. M. Lanus and L. Yin and K. S. Trivedi, "Hierarchical composition and aggregation of state-based availability and performanbility models," *IEEE Transactions on Reliability*, vol. 52, pp. 44–52, 2003.

36. G. Strang. Introduction to linear algebra, 4th ed. *Wellesley Cambridge Press*, 2009.

37. B. W. Johnson, "Design and analysis of fault-tolerant digital systems," Reading, MA, USA: Addison-Wesley Publishing Company.

38. K. N. Fleming, "A redundant model for common model failures in redundant safety systems," in *Proceedings of the Sixth Pittsburgh Annual Modeling and Simulation Conference*, pp.579–581, Pittsburgh, PA, USA, 1975.

39. P. A. Jensen, J. F. Bard, "Operations research models and methods," John Wiley & Sons Incorporated, 2003.

40. B. Plateau, W. J. Stewart, "Stochastic automata networks," in *Computational Probability*, edited by W. K. Grassmann, university of Saskatchewan, Boston; London: Kluwer Academic, 2000.

41. Z. W. Birnbaum, *On the importance of different components in a multicomponent system*, In P. R. Krishnaiah, editor, *Multivariate Analysis II*, pp. 581–592, New York, NY, USA, Academic Press, 1969.

42. E. J. Henley and H. Kumamoto, *Reliability engineering and risk assessment*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1981.

43. P. M. Frank, *Introduction to system sensitivity theory*, Academic Press, New York, NY, USA, 1978.

44. P. P. Bocharov, C. D'Apice, and A. V. Pechinkin, "Queueing theory (modern probability and statistics)," Published by De Gruyter, Oct. 31, 2003.

45. P. E. Heegaard and K. S. Trivedi, "Network survivability modeling," *Computer Networks*, vol. 53, no. 8, pp. 1215–1234, 2009.

46. A. V. Ramesh and K. S. Trivedi, "On the sensitivity of transient solutions of Markov models," in *Proceedings of the 1993 ACM SIGMETRICS conference*, pp. 122–134, 1993

47. M. Kijima, *Markov processes for stochastic modeling*, Chapman and Hall, 1997.

48. A. L. Reibman and K. S. Trivedi, *Numerical transient analysis of Markov models*, Computers and Operations Research, vol. 15, pp. 19–36, 1988.

49. H. Takajo, T. Takahashi, "Noniterative method for obtaining the exact solution for the normal equation in least-squares phase estimation from the phase difference," *Journal of the Optical Society of America A*, vol. 5, pp. 1818–1827, 1988.

50. H. M. Islam, M. A. Khan, "On system reliability with single strength and multi-component stress model," *International Journal of Quality and Reliability Management*, vol. 26, issue 3, pp. 302–307, 2009.

PUBLICATION LIST

(1) International Journal

1. J. Zheng, H. Okamura and T. Dohi, “Availability importance measures for virtualized system with live migration,” Applied Mathematics, vol. 6, no. 2, pp. 359–372, 2015.
2. J. Zheng, H. Okamura and T. Dohi, “Survivability analysis of VM-based intrusion tolerant systems,” IEICE Transactions on Information & Systems (D), vol. E98-D, no. 12, pp. 2082–2090, 2015.

(2) International Conference

3. J. Zheng, H. Okamura and T. Dohi, “Component importance analysis of virtualized system,” Proceedings of the 9th International Conference on Autonomic and Trusted Computing (ATC 2012), pp. 462–469, IEEE CPS, 2012. (Fukuoka, Japan, September 4-7, 2012).
4. J. Zheng, H. Okamura and T. Dohi, “Sensitivity analysis of reliability function for virtualized system,” Proceedings of the 8th International Conference on Mathematical Methods in Reliability, –Theory, Methods and Applications– (MMR 2013), pp. 229–232, 2013. (Stellenbosch, South Africa, July 1-4, 2013).
5. J. Zheng, H. Okamura and T. Dohi, “Component importance measures for real-time computing systems in the presence of common-cause failures,” Proceedings of the 21st IEEE Pacific Rim International Symposium on Dependable Computing (PRDC 2015), pp. 301–310, IEEE CS Press, 2015. (Zhangjiajie, China, November 18-20, 2015).