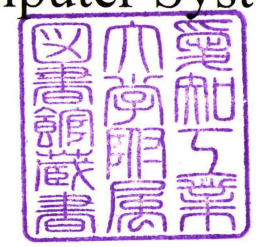
Studies on Optimal Checkpoint Intervals for Computer Systems



Kenichiro Naruse

February 2008

Studies on Optimal Checkpoint Intervals for Computer Systems

by

Kenichiro Naruse

Dissertation submitted in partial fulfillment

for the degree of Doctor of Engineering

Under the supervision of

Professor Toshio Nakagawa

Aichi Institute of Technology

Toyota, Japan

February 2008

Abstract

Computer systems have been required to operate normally and effectively, and also hold high reliability as communication and information systems have been developed and complicated. However, some errors often occur due to noises, human errors, software bugs, hardware faults, computer viruses, and so on, and lastly, they might become faults and incur system failures. To protect such faults, various kinds of fault tolerant techniques such as the redundancy of processors and memories and the configurations of systems have been provided. The high reliability and effective performance of real systems can be achieved by the use of redundant techniques in reliability theory.

Some faults due to errors may be detected after some time has passed. A system consistency that may be lost by some faults should be restored by some recovery techniques. The operation of taking copies of the normal state of the system is called checkpoint. When faults have been occurred, the process goes back to the nearest checkpoint time by rollback operation, and its re-execution is made, using a consistent state stored in the checkpoint time.

An initial chapter gives the introduction which is constructed by redundant techniques for improving reliability and achieving fault tolerance, failure detection and recovery methods and the organization of Thesis. Chapter 2 proposes the checking

model where the backup is carried out until the latest checking time when some failure was detected. The expected cost is obtained by using the inspection policy in reliability theory and optimal policies, which minimize it for two cases of periodic and sequential checking times, are derived.

Chapters 3 to 7 consider several checkpoint models when an original execution time of one process or task is given, and discuss when and how to generate checkpoints to reduce the total overhead of processes: Chapter 3 proposes two-level recovery schemes of soft and hard checkpoints, and derives an optimal interval of soft checkpoint between hard checkpoints. Chapter 4 adopts multiple modular redundant systems as the recovery techniques of error detection and error masking, and derives optimal checkpoint intervals. Chapter 5 considers the modified checkpoint model in Chapter 4 where checkpoints are placed at sequential times and error rates increase with the number of checkpoints and with an original execution time. It is supported in Chapters 6 that tasks with random processing times are executed successively, and two types of checkpoints are placed at the end of tasks. Three schemes are considered and are compared numerically. Chapter 7 proposes the extended checkpoint model where error rates increase with the number of checkpoints as shown in Chapter 5.

The numerical examples are given in each chapter to understand the results easily. The results are summarized in the end of thesis and future studies are described.

Acknowledgment

The author would be like to appreciate Professor Toshio Nakagawa, the supervisor of my study for his constant guidance, encouragement and suggestions through this study.

The author wishes to thank the members of this dissertation reviewing committee: Professor Kazumi Yasui, Professor Naohiro Ishii and Professor Tetsuhisa Oda for their careful reviews of this dissertation.

The author is also grateful thank to Professor Shunji Osaki of Nanzan University for having presented the papers at some national conferences, and wishes to thank Dr Satoshi Mizutani and all members of Nagoya Computer and Reliability Research Group for their useful comment and discussions.

The author wishes to thank Professor Kazuyuki Teramoto for giving me the opportunity to take the degree of Doctor.

Furthermore, the author would like to thank President, all staff of Nagoya Sangyo University and Business College Excellence for continual support for this study.

This dissertation could not have been accomplished without guidance and encouragement of the above members.

Finally, the author wishes to thank my family for their mental and various supports.

Contents

Chapter 1

Introduction

In recent years, computers have been used not only to live our daily life, but also to make and sell good products in industries. Most things have computers within them and are moved by computers. Computers play more important role in a highly civilized society. Especially, computer systems have been required to operate normally and effectively as communication and information systems have been developed rapidly and complicated remarkably. However, some errors due to noises, human errors, hardware faults, computer viruses, and so on, occur certainty in systems. Lastly, those errors might have become faults and incur system failures. Such failures have sometimes caused a heavy damage to a human society and have fallen into general disorder. To prevent such faults, various kinds of fault tolerant techniques such as the redundancy of processors and memories and the configuration of systems have been

provided [12, 23, 42]. The high reliability and effective performance of real systems can be achieved by fault tolerant techniques.

Partial data loss and operational errors in computer systems are generally called *error* and *fault* caused by errors. *Failure* indicates that faults are recognized on the exterior systems. Three different techniques of decreasing the possibility of fault occurrences can be used [1]: *Fault avoidance* is to prevent fault occurrences by improving qualities of structure parts and placing well surroundings. *Fault masking* is to prevent faults by error correction codes and majority voting. *Fault tolerance* is that systems continue to function correctly in the presence of hardware failures and software errors. There techniques above are called simply fault tolerance into one word.

Some faults due to operational errors may be detected after some time has passed and a system consistency may be lost by them. Then, we should restore a consistent state just before fault occurrences by some recovery techniques. The operation that takes copies of the normal state of the system is called *checkpoint*. When faults have been occurred, the process goes back to the nearest checkpoint time by rollback operation [5, 14, 30], and its retry is made, using the copy of a consistent state stored in the checkpoint time.

It is supposed that we have to complete the process of one task with a finite execution time. A module is an element such as a logical circuit or a processor that executes certain lumped parts of the task. Then, we consider the checkpoint models of error detection and masking by redundancy, and propose their modified models. Using

reliability theory, we analyze these models and discuss analytically optimal checkpoint intervals.

1.1 Redundant Techniques

High system reliability can be achieved by redundancy. A classical standard problem is to determine how reliability can be improved by using redundant units. The results of various redundant systems with repair were summarized as *repairman problem*, and optimization problems of redundancy and allocation subject to some constrains were solved and qualitative relationships for multicomponent structures obtained [2]. Further, some useful expressions of reliability measures of many redundant systems were shown [3, 40]. The fundamentals and applications of system reliability and reliability optimization in system design were well described [9]. Various combinatorial reliability optimization problems with multiple constrains for different system structures were considered [38] and their computational techniques were surveyed [13].

Redundancy techniques of a system for improving reliability and achieving fault tolerance are classified commonly into the following forms [1, 12, 23]:

(1) Hardware Redundancy

(a) *Static hardware redundancy* is fault masking technique in which effects of faults are essentially hidden from the system with no specific indication of their occurrence. Existing faults are not removed. A typical example is triple modular redundancy.

(b) *Dynamic hardware redundancy* is fault tolerance technique in which the system continues to function by detection and removing faults, replacing faulty units, and making reconfiguration. Typical examples are standby sparing system and graceful degrading system [6].

(c) *Hybrid hardware redundancy* is a combination of the advantages of static and dynamic hardware redundancies.

(2) Software Redundancy

This technique is to use extra codes, small routines or possibly complete program to check the correctness or consistency of the results produced by software. Typical examples are *N*-version programming and Ad-Hoc technique.

(3) Information Redundancy

This technique adds redundant information to data to allow fault detection, fault masking, and fault tolerance. Examples are error-detecting codes such as parity codes and signatures, and watchdog processor.

(4) Time Redundancy

This technique is the repetition of a given computation at a number of times and the comparison of results. This is used to detect transient of intermittent

faults, to mask faults and to recover the system. Typical examples are retries and checkpoint schemes.

Redundancies (1), (2) and (3) are also called *Space Redundancy* because high reliability is attained by providing multiple resources of hardware and software.

In this thesis, we take up several checkpoint schemes for redundant modular systems as recovery techniques.

1.2 Failure Detection and Recovery Methods

It is important to know useful techniques for failure detection because most systems have become larger and more complex. Actually, several methods to detect failures have been proposed: O'Connor [24] surveyed widely the techniques related with tests for electronic circuits. Lala [11] summarized fault-tolerant design techniques with self-checking of digital circuits.

The methods of self-checking which involves fault-secure and self-testing are required to design high reliable systems. Fault-secure means that a failed system outputs codes except an assumed output code space. Self-testing means that a failed system outputs codes except an assumed code space for at least one input code.

Another method for systems such as digital circuits is the comparison-checking with outputs of double modular systems: Two modules execute the same process and compare two states at checkpoint times. If two states of each module do not match with each other, this means system failure. One extending system is a majority decision system, *i.e.* , an $(n+1)$-out-of-$(2n+1)$ system as an error masking system. If $(n+1)$ or more states of $(2n+1)$ modules match, the system is correct.

On the other hand, when the logical consistency is lost by failures, the following two operations of process recovery are performed: One is forward recovery which keeps running forward in a fault status without backward. Such method is applied to real time systems as weather satellite's picture and to forward the voice of IP telephony system. The other is backward recovery which goes back before failure occurrence. Typical three methods are usually used [1, 10, 12, 23]:

(a) Retry Recovery Method

Retry recovery method is very popular and easily method: If a fault occurs, the procedure of retry is made immediately. But if the fault changes an original data, the retry fails. This method is used for hard disk read, memory read, and so on.

(b) Checkpoint Recovery Method

This is the most general method of backward recovery system. This method records system states which need to run continually the process at suitable intervals. If a fault is detected at some checkpoint, the process goes back to

the latest checkpoint. It is very important to determine the interval times of checkpoints. If we generate checkpoints at short intervals, their overheads are large. So that, the system performance becomes low. But, if some error is detected in the process, the process goes back to near checkpoint and is re-executed. In this case, the re-execution time is small. On the other hand, if we generate checkpoints at long intervals, their overheads are small. So that, the system performance becomes high. But, if some error is detected in the process, the process goes back to far checkpoint and is re-executed. In this case, the re-execution time is large. Therefore, it is one kind of trade-off problem to generate checkpoint intervals. We study about optimal checkpoints intervals in this thesis.

(c) Journal Recovery Method

Journal recovery method is an easy method, but, it needs a longer time than above two methods when a failure occurs. This method records an initial point state and records all of transactions about changing data. If a failure occurs, the process is re-executed from initial data and all transactions. So that, this method needs a longer time than above two methods.

1.3 Outline of Thesis

This section describes the outline of this thesis. This thesis is divided into Introduction, Chapters 2-7, Conclusions and Bibliography.

Chapter 2 considers a modified inspection model in reliability theory: When some failure is detected, the backup operation is carried out until the latest checking time. The expected cost from the failure detection to the latest checking time is obtained. Optimal policies, which minimize the expected cost rates for two cases of periodic and sequential checking times, are analytically discussed. Modified models where the operation time is finite and some fault remains hidden and proposed.

Chapter 3 considers a two-level recovery scheme of the checkpoint model with soft and hard checkpoints: Soft checkpoint is less reliable and less overhead than hard checkpoint, and is set up between them. The total expected overhead during the interval of hard checkpoints is obtained, using Markov renewal processes, and an optimal policy which minimizes it is discussed. A numerical example shows that a two-level scheme can achieve a good performance.

Chapter 4 considers multiple modular redundant systems of error detection and error masking on a finite process execution when checkpoints are placed at periodic times. The mean times to the completion of the process are obtained, using renewal equations, and optimal checkpoint intervals which minimize them are discussed analytically. It is shown numerically what a majority decision system is optimal.

Chapter 5 adopts a modular redundant system on a finite process execution when checkpoints are placed at sequential times. Two checkpoint models where error rates increase with the number of checkpoints and with an original execution time are considered. The mean times to the completion of the process are obtained, and optimal checkpoint intervals which minimize them are computed numerically by solving simultaneous equations. Further, an approximation method is proposed.

Chapter 6 considers a double modular system when tasks with random processing times are executed successively. Two types of checkpoints such as compare-checkpoint and compare-and-store-checkpoint can be placed at the end of tasks. The mean execution times per one task for three schemes are obtained, and optimal policies which minimize them are discussed analytically. Extended models with majority decision modules and a spare module are proposed.

Chapter 7 considers the random checkpoint model with increasing error rates. The mean execution times per one task for three schemes are obtained, and optimal policies which minimize them are derived analytically. It is shown numerically that what a majority system is optimal

Finally, Chapter 8 summarizes the results derived in this thesis and presents some future studies.

Chapter 2

Checking Time of Backup Operation for a Database System

When a failure occurs in the process of a database system, we execute the rollback operation until the latest checking time and make the recovery of database files. This chapter proposes a modified inspection model where the backup is carried out until the latest checking time when some failure was detected. The expected cost until the backup operation is made to the latest checking time is derived, and optimal policies, which minimize it for two cases of periodic and sequential checking times, are analytically discussed. Some further modified models where the operating time is finite and a fault remains hidden are proposed.

2.1 Introduction

Most units in standby [39, 16] and in storage [7, 15] have to be checked at planned times to detect failures. Barlow and Proschan [2] summarized such inspection policies which minimize the total expected cost until a failure detection. All inspection models have assumed that any failure is known only through checking and summarized in [17]. But, when a failure was detected in the recovery technique of a database system, we execute the rollback operation until the latest checkpoint [5, 31] and reconstruct the consistency of a database. It has been assumed in such models that any failures are always detected immediately, however, there is a loss time or cost associated with the lapsed time of rollback operation between a failure detection and the latest checkpoint.

Further, this model would be applied to the backup policy for hard disks [32, 33]: There is a variety of files in the disk, however, they may be sometimes lost due to human errors or disk failures. To prevent such events, backup files are made at suitable times, which are called a backward time. When failures have occurred, we can make the recovery of files at each backward time.

From the practical viewpoints of database recovery and backup files, we propose the following backup operation model which is one of the modified inspection policies: When some failure of a unit was detected, we carry out the backup operation to the latest checking time. In such a model, we do not wish to provide checks much frequently, and on the other hand, we wish to avoid a long elapsed time between a

failure detection and the checking time. It would be an important problem to determine an optimal checking schedule of this model.

By the similar method to that of the usual inspection model [17], we derive the total expected cost until the completion of backup operation after a failure detection, and discuss optimal checking times which minimize it for two cases of periodic and sequential policies. We give numerical examples when failure times of a unit have a Weibull and uniform distributions.

Further, we consider the case where a unit has to be operating for a finite interval. The expected cost is obtained, and an optimal checking time which minimizes it is numerically computed. Finally, the expected cost per unit of time and the availability are also derived. We propose one modified model where a fault occurs and is hidden, and after that, a failure occurs, and obtain the expected cost.

2.2 Expected Costs

Suppose that the failure time of a unit has a general distribution $F(t)$ with a finite mean $\mu \equiv \int_0^\infty \overline{F}(t)dt$, where $\overline{F}(t) \equiv 1-F(t)$. The checking schedule of a unit is made at successive times $T_k (k=1, 2, \cdots)$ where $T_0 \equiv 0$. Let c_1 be the cost required for each check. Further, when a failure was detected between T_k and T_{k+1}, we carry out the backup operation to the latest checking time T_k. This incurs a loss cost c_2 per unit of time (Figure 2.1).

T_k Checking time ✖ Failure

Figure 2.1: Process of sequential checking times T_k.

The total expected cost until a failure is detected and the backup operation is made to the latest checking time is, using the theory of inspection policy [2],

$$C_1(T_1, T_2, \cdots) = \sum_{k=0}^{\infty} \int_{T_k}^{T_{k+1}} [kc_1 + c_2(t - T_k)] \mathrm{d}F(t)$$

$$= \sum_{k=1}^{\infty} [c_1 - c_2(T_k - T_{k-1})] \overline{F}(T_k) + \mu c_2. \tag{2.1}$$

If a unit is checked at periodic times kT(k=1, 2, $\cdots$) then

$$C_1(T) = (c_1 - c_2 T) \sum_{k=1}^{\infty} \overline{F}(kT) + \mu c_2. \tag{2.2}$$

Next, we obtain the expected cost per unit of time for an infinite time span. Since the mean time of backup operation from a failure detection to the latest checking time is

$$\sum_{k=0}^{\infty} \int_{T_k}^{T_{k+1}} (t - T_k) \mathrm{d}F(t),$$

the expected cost rate is given by

$$C_2(T_1,T_2,\cdots)=\frac{C_1(T_1,T_2,\cdots)}{\mu+\sum_{k=0}^{\infty}\int_{T_k}^{T_{k+1}}(t-T_k)\mathrm{d}F(t)}$$

$$=\frac{c_1\sum_{k=1}^{\infty}\overline{F}(T_k)-\mu c_2}{2\mu-\sum_{k=1}^{\infty}(T_k-T_{k-1})\overline{F}(T_k)}+c_2. \tag{2.3}$$

If a unit is checked at periodic times kT (k=1, 2, ···) then

$$C_2(T)=\frac{c_1\sum_{k=1}^{\infty}\overline{F}(kT)-\mu c_2}{2\mu-T\sum_{k=1}^{\infty}\overline{F}(kT)}+c_2. \tag{2.4}$$

2.3 Optimal Policies

We discuss optimal checking times T_k^* which minimize the expected cost $C_1(T_1, T_2, \cdots)$ in (2.1). Let f(t) be a density function of $F(t)$, *i.e.*, $f(t)\equiv F'(t)$. Then, differentiating $C_1(T_1, T_2, \cdots)$ with respect to T_k and setting it equal to zero, we have

$$\frac{F(T_{k+1})-F(T_k)}{f(T_k)}=T_k-T_{k-1}-\frac{c_1}{c_2}\qquad(k=1,2,\cdots). \tag{2.5}$$

Thus, we can determine the optimal checking times T_k^*, using Algorithm 1 of [2].

In the periodic inspection case, from (2.2), we have

$$C_1(0) = \lim_{T\to 0} C_1(T) = \infty,$$
$$C_1(\infty) = \lim_{T\to \infty} C_1(T) = \mu c_2.$$

Hence, we have

$$C_1(\infty) - C_1(T) = (c_2 T - c_1)\sum_{k=1}^{\infty} \overline{F}(kT).$$

Thus, there exists an optimal checking time T_1^* $(c_1/c_2 < T_1^* < \infty)$ which minimizes $C_1(T)$, and

$$C_1(c_1/c_2) = C_1(\infty) = \mu c_2 .$$

Further, differentiating $C_1(T)$ in (2.2) with respect to T and setting it equal to zero imply

$$T - \frac{\sum_{k=1}^{\infty} \overline{F}(kT)}{\sum_{k=1}^{\infty} k f(kT)} = \frac{c_1}{c_2}. \tag{2.6}$$

In the case of $F(t)=1-e^{-\lambda t}$, Equation (2.6) is

$$T - \frac{1-e^{-\lambda T}}{\lambda} = \frac{c_1}{c_2}. \tag{2.7}$$

It can be easily seen that the left-hand side of (2.7) is strictly increasing from 0 to ∞. Thus, there exists a finite and unique T_1^* which satisfies (2.7), and the resulting cost is

$$C_1(T^*) = c_2 T_1^* - c_1. \tag{2.8}$$

Since $e^a \approx 1 + a + a^2/2$ for small $a > 0$, the optimal checking time is approximately given by

$$\tilde{T}_1 = \sqrt{\frac{2c_1}{\lambda c_2}}, \tag{2.9}$$

and $T_1^* > \tilde{T}_1$

We can compute an optimal schedule which minimizes $C_2(T_1, T_2, \cdots)$ in (2.3), using the algorithm 2 of [2]. When $F(t)=1-e^{-\lambda t}$, Equation (2.4) is

$$C_2(T) = \frac{c_1 e^{-\lambda T} - c_2(1-e^{-\lambda T})/\lambda}{2(1-e^{-\lambda T})/\lambda - Te^{-\lambda T}} + c_2, \tag{2.10}$$

and

$$C_2(0) = \infty, \quad C_2(\infty) = \frac{c_2}{2}.$$

Differentiating (2.10) with respect T and setting it equal to zero,

$$\frac{T-(1-e^{-\lambda T})/\lambda}{2-e^{-\lambda T}} = \frac{c_1}{c_2}. \tag{2.11}$$

It can be easily seen that the left-hand side of (2.11) is strictly increasing from 0 to ∞. Thus, there exists a finite and unique T_2^* which satisfies (2.11). By comparing (2.7) and (2.11), it can be shown that $T_1^* < T_2^*$, and it is approximately

$$\tilde{T}_2 = \frac{c_1}{c_2} + \sqrt{\left(\frac{c_1}{c_2}\right)^2 + \frac{2c_1}{\lambda c_2}}. \tag{2.12}$$

2.4 Finite Interval

Suppose that a unit has to be operating for a finite interval $(0, S]$ $(0 < S < \infty)$, and is replaced at time $S=T_N$ [17]. The other assumptions are the same as those of the previous model. Then, the total expected cost before replacement is

$$C_3(N)=\sum_{k=0}^{N-1}\int_{T_k}^{T_{k+1}}[kc_1+c_2(t-T_k)]\mathrm{d}F(t)+c_1(N-1)\overline{F}(T_N)\quad(N=1,2,\cdots). \tag{2.13}$$

Putting that $\partial C_3(N)/\partial T_k=0$, we have (2.5), and the resulting minimum expected cost is

$$\begin{aligned}\widetilde{C}_3(N) &= C_3(N)+c_1-c_2\int_0^S\overline{F}(t)\mathrm{d}t\\ &=\sum_{k=0}^{N-1}[c_1-c_2(T_{k+1}-T_k)]\overline{F}(T_k).\end{aligned} \tag{2.14}$$

For example, when N=3, the checking times T_1 and T_2 are given the solution of simultaneous equations

$$\begin{aligned}\frac{F(S)-F(T_2)}{f(T_2)} &= T_2-T_1-\frac{c_1}{c_2},\\ \frac{F(T_2)-F(T_1)}{f(T_1)} &= T_1-\frac{c_1}{c_2},\end{aligned}$$

and the expected cost is

$$\widetilde{C}_3(3)=c_1-c_2T_1+[c_1-c_2(T_2-T_1)]\overline{F}(T_1)+[c_1-c_2(S-T_2)]\overline{F}(T_2).$$

From the above discussions, we compute T_k(k=1, 2, ···, N−1) which satisfies (2.5), and substituting them into (2.14), we obtain the expected cost $\widetilde{C}_3(N)$. Next, computing $\widetilde{C}_3(N)$ for all $N\geq 1$, we can get the optimal checking number N^* and times T_k^*(k = 1, 2, ···, N^*).

2.5 Numerical Examples

We compute the optimal checking times numerically when the failure time has a Weibull distribution for the periodic checking time, and when the failure time has a Weibull distribution and a uniform distribution for the sequential checking one.

Suppose that $F(t)=1-\exp[-(\lambda t)^{\alpha}]$ ($\alpha > 1$). Then, from (2.6), an optimal time T_1^* is given by a solution of the equation

$$T-\frac{\sum_{k=1}^{\infty} e^{-(\lambda kT)^{\alpha}}}{\sum_{k=1}^{\infty} \lambda\alpha k(\lambda kT)^{\alpha-1} e^{-(\lambda kT)^{\alpha}}}=\frac{c_1}{c_2}.$$

Table 2.1 presents the optimal checking times T_1^* for α=1.0, 1.5, 2.0, 2.5, 3.0 and c_1/c_2=5, 10, 20, 30, 40, 50 when $1/\lambda$=500. Note that when α=1, this corresponds to an exponential case. Approximate times $\tilde{T}_1$ in (2.9) give a good lower bound for T_1^* for α=1. This indicates that the optimal times become longer as c_1/c_2 are large, however, they are changed little as the values of parameter α increase for $\alpha > 1$.

Table 2.1: Optimal checking times T_1^* to minimize $C_1(T)$ when $F(t) = 1 - \exp[-(t/500)^\alpha]$.

α	c_1/c_2					
	5	10	20	30	40	50
1.0	72.42	103.45	148.42	183.81	214.27	241.59
1.5	67.40	95.50	135.53	166.49	192.78	216.11
2.0	66.57	94.14	133.13	163.06	188.28	210.50
2.5	66.59	94.16	133.08	162.89	187.96	210.09
3.0	66.82	94.48	133.57	163.51	188.71	210.87
$\tilde{T}_1$	70.71	100.00	141.42	173.21	200.00	223.61

Table 2.2 gives the optimal checking schedule $\{T_k^*\}$ which satisfies (2.5), and $\delta_k \equiv T_{k+1}^* - T_k^*$ when $F(t)=1-\exp[-(\lambda t)^2]$ for $c_1/c_2=10$, 20, 30. It is roughly seen that the periodic intervals are almost the same value as δ_1, *i.e.*, the interval between the first and second checking times. Each checking time becomes longer as c_1/c_2 are large.

Next, suppose that the failure time is uniformly distributed during $[0, S]$ $(0 < S < \infty)$, *i.e.*,

$$f(t) = \begin{cases} \dfrac{t}{S} & \text{for } 0 \le t \le S, \\ 0 & \text{for } t > S. \end{cases}$$

Then, Equation (2.5) is rewritten as

$$T_{k+1} - T_k + \frac{c_1}{c_2} = T_k - T_{k-1},$$

which is equal to that of [2]. For example, when S=1000 and c_1/c_2=20, we have that N=10 since $N(N-1) < 2c_2S / c_1 = 100$ [2], and $T_k^* = \{100, 360, 510, 640, 750, 840, 910, 960, 990, 1000\}$. These values are a little larger than those in Table 2.3 for k=2, 3, ···, 9 when N=10. It would be trivial in the case of a uniform distribution that the optimal schedule is equal to that of the standard inspection model [17].

Table 2.3 gives the checking times T_k (k=1, 2, ···, N) and the expected cost $\tilde{C}_3(N)$ for S=1000 and c_1/c_2=20 when $F(t)$=1−exp[−(t/500)^2]. Comparing $\tilde{C}_3(N)$ for N=1, 2, ···, 10, the expected cost is minimum at N=9. That is, the optimal checking number is N^*=9 and optimal checking times are 183.56, 319.94, 436.78, 542.25, 640.21, 732.98, 822.41, 910.45 and 1000. These optimal times are almost the same ones in Table 2.2 for c_1/c_2=20.

Table 2.4 gives T_k and the expected cost $\tilde{C}_3(N)$ for S=500 and the same parameters in Table 2.3. In this case, the optimal checking number is N^*=4 and optimal checking times are 174.82, 302.60, 408.84, 500. All checking times in Table 2.4 are smaller than those in Table 2.3 for the same number N.

Table 2.2: Optimal checking times T_k^* to minimize $C_1(T_1, T_2, \cdots)$ and $\delta_k = T_{k+1}^* - T_k^*$ when $F(t)=1-\exp[-(t/500)^2]$.

k	$c_1/c_2=10$		$c_1/c_2=20$		$c_1/c_2=30$	
	T_k^*	δ_k	T_k^*	δ_k	T_k^*	δ_k
1	145.04	107.24	183.23	136.06	211.09	157.44
2	252.28	91.62	319.29	116.43	368.53	135.06
3	343.90	82.54	435.72	104.90	503.59	121.86
4	426.44	76.49	540.92	97.09	625.45	112.86
5	502.93	72.20	637.71	91.40	738.29	106.13
6	575.13	69.12	729.11	87.15	844.42	100.88
7	644.25	67.03	816.26	84.06	945.30	96.61
8	711.28	65.90	900.32	82.17	1041.91	93.04
9	777.18	65.89	982.49	81.92	1134.95	89.99
10	843.07		1064.41		1224.94	

Table 2.3: Checking times T_k and the expected cost $\widetilde{C}_3(N)$ for N=1, 2, ···, 10 when S=1000, c_1/c_2=20 and $F(t)$=1−exp[−$(t/500)^2$].

k	N				
	1	2	3	4	5
1	1000	358.24	270.74	231.15	209.68
2		1000	518.75	420.95	373.79
3			1000	623.89	529.76
4				1000	703.78
5					1000
6					
7					
8					
9					
10					
$\widetilde{C}_3(N)/c_2$	−17.95	−213.82	−273.18	−299.18	−312.05

k	N				
	6	7	8	9	10
1	197.27	190.00	185.83	183.56	182.43
2	347.79	332.93	324.51	319.94	317.68
3	483.50	458.25	444.27	436.78	433.10
4	618.15	576.18	553.93	542.25	536.58
5	769.00	694.31	658.35	640.21	631.54
6	1000	824.00	761.87	732.98	719.61
7		1000	870.82	822.41	801.36
8			1000	910.45	876.46
9				1000	943.53
10					1000
$\widetilde{C}_3(N)/c_2$	−318.61	−321.85	−323.28	−323.73	−323.67

Table 2.4: Checking times T_k and expected cost $\widetilde{C}_3(N)$ for N=1, 2, ⋯, 7 when S=500, c_1/c_2=20 and $F(t)$=1−exp[−$(t/500)^2$].

k	N						
	1	2	3	4	5	6	7
1	500	263.71	201.84	174.82	162.01	156.55	155.33
2		500	357.26	302.60	277.67	267.21	264.87
3			500	408.84	369.89	353.91	350.37
4				500	443.77	421.41	416.49
5					500	470.32	463.88
6						500	491.91
7							500
$\widetilde{C}_3(N)/c_2$	-176.58	-264.10	-280.93	-282.10	-277.75	-271.19	-263.89

2.6 Concluding Remarks

We have considered the recovery model where we carry out the backup operation to the latest checking time when a failure was detected, and have discussed the optimal policies which minimize the total expected cost or the expected cost rate, using the

techniques of inspection policies. Further, we have computed numerically the optimal checking time when a unit should be operating for a finite interval when the failure time has a Weibull distribution. These techniques would be applied to other backup models of a database system [25] and the reliability model with backward time [21].

It is more useful in some systems to adopt the availability than the expected cost as an appropriate objective function. If β_1 is the time for one check then the availability is

$$A(T_1,T_2,\cdots) = \frac{\text{Mean operating time}}{\text{Mean time until the complection of backup operation}}$$

$$= \frac{\mu}{2\mu - \sum_{k=1}^{\infty}(\beta_1 + T_k - T_{k-1})\overline{F}(T_k)}. \tag{2.15}$$

Thus, the optimal policy which maximizes the availability corresponds to the policy which minimizes the expected cost $C(T_1,T_2,\cdots)$ in (2.1) by replacing $c_1 = \beta_1$ and $c_2 = 1$.

Finally, we consider the following model as one of modified ones: A fault occurs between the j-th and the $(j+1)$-th checking times according to a general distribution $F(t)$ with a finite mean μ, and is hidden. After that, a failure occurs between the k-th and the $(k+1)$-th checking times and is detected immediately, and the time from the occurrence of a fault to the failure detection obeys a general distribution $G(x)$ with mean θ. This is called a fault latency [34]. When a failure was detected, we carry out the backup operation to the j-th checking time. Then, the expected cost until a failure is detected and the backup operation is made to the j-th checking time is

$$C_4(T_1,T_2,\cdots)=\sum_{k=1}^{\infty}\sum_{j=0}^{k-1}\int_{T_k}^{T_{k+1}}\left\{\int_{T_j}^{T_{j+1}}\left[kc_1+c_2(x-T_j)\right]\mathrm{d}F(t)\right\}\mathrm{d}G(x-t)$$

$$+\sum_{j=0}^{\infty}\int_{T_j}^{T_{j+1}}\left\{\int_{T_j}^{x}\left[jc_1+c_2\left(x-T_j\right)\right]\mathrm{d}F(t)\right\}\mathrm{d}G(x-t)$$

$$=\sum_{j=0}^{\infty}\int_{T_j}^{T_{j+1}}\left\{\sum_{k=j+1}^{\infty}\int_{T_k}^{T_{k+1}}\left[kc_1+c_2\left(x-T_j\right)\right]\mathrm{d}G(x-t)\right\}\mathrm{d}F(t)$$

$$+\sum_{j=0}^{\infty}\int_{T_j}^{T_{j+1}}\left\{\int_{x}^{T_{j+1}}\left[jc_1+c_2\left(x-T_j\right)\right]\mathrm{d}G(x-t)\right\}\mathrm{d}F(t)$$

$$=c_1\sum_{j=0}^{\infty}\int_{T_j}^{T_{j+1}}\left[j+\sum_{k=j+1}^{\infty}\overline{G}(T_k-t)\right]\mathrm{d}F(t)+c_2\sum_{j=0}^{\infty}\int_{T_j}^{T_{j+1}}\left[\theta+t-T_j\right]\mathrm{d}F(t)$$

$$=c_1\sum_{k=1}^{\infty}\left[\overline{F}(T_k)+\int_0^{T_k}\overline{G}(T_k-t)\mathrm{d}F(t)\right]+c_2\left\{\mu+\theta-\sum_{k=1}^{\infty}T_k\left[\overline{F}(T_k)-\overline{F}(T_{k+1})\right]\right\}. \tag{2.16}$$

If a faults is not hidden and is detected immediately, *i.e.*, $G(x)\equiv 1$ for $x\geq 0$ and $\theta=0$, this corresponds to the previous model and Equation (2.16) agrees with (2.1).

Further, suppose that $f(t)$ and $g(x)$ are density functions of $F(t)$ and $G(x)$, respectively. Then, differentiating $C_4(T_1, T_2, \cdots)$ with respect to T_k and setting it equal to zero,

$$\frac{(T_k-T_{k-1})f(T_k)-\left[F(T_{k+1})-F(T_k)\right]}{\int_0^{T_k}g(T_k-t)\mathrm{d}F(t)}=\frac{c_1}{c_2}. \tag{2.17}$$

Thus, using Algorithm 1 of [2], we can compute the optimal checking time numerically for specified distributions $F(t)$ and $G(x)$.

Chapter 3

Checkpoint Interval for Two-Level Recovery Schemes

It is important to design computer systems to tolerate some failures. This chapter proposes two-level recovery schemes; soft checkpoint (SC) and hard checkpoint (HC) which are useful to recover from failures. Soft checkpoint is less reliable and less overhead than those of HC, and is set up between HCs to reduce the overhead of the process. The total expected overhead of one cycle from HC to HC is obtained, using Markov renewal processes, and an optimal interval which minimizes it is computed. It is shown in a numerical example that a two-level recovery scheme can achieve a good performance.

3.1 Introduction

In computer and database information systems, some errors often occur due to noises, human errors, software bugs and hardware faults, and make these systems inherently unreliable. In such cases, it is important to restore a consistent state by rollback *recovery techniques. Checkpoint is the most effective recovery mechanism which* stores a consistent state in the secondary storage at suitable times. Even if failures occur, the process goes back to the latest checkpoint and can resume its normal operation [5, 12, 30]. Ling *et al.* [14] made a good survey of such checkpoint problems.

Vaidya [41, 42] considered two-level recovery schemes in which *N-checkpoint* can recover from several number of failures, and *1-checkpoint* is taken between *N*-checkpoint and can recover from only a single failure. He presented an analytical approach for evaluating the performance of two-level schemes, using a Markov chain. Further, Ssu *et al.* [36] described an adaptive protocol that manages the storage for base stations in mobile environments, where *soft checkpoint* is saved in a mobile host, *e.g.*, in a local disk or flash memory, and *hard checkpoint* is saved in a base station. Soft checkpoints will be lost if a mobile host fails, however, hard checkpoints can survive but have higher overheads since they must be transmitted through the wireless channels.

This chapter considers two-level recovery schemes based on the proposed scheme [41]: Soft checkpoint (SC) and hard checkpoint (HC) which are useful to recover from only one failure and several failures, respectively. SC are set up at periodic intervals between HC, and are less reliable and less overhead than those of HC. We discuss an optimal checkpoint interval of SC when HC are placed on the beginning and at the end of the process. The total expected overhead of one cycle from HC to HC is obtained, using Markov renewal processes [27], and an optimal interval which minimizes it numerically computed. It is shown in a numerical example that two-level schemes reduce the total overhead of the process.

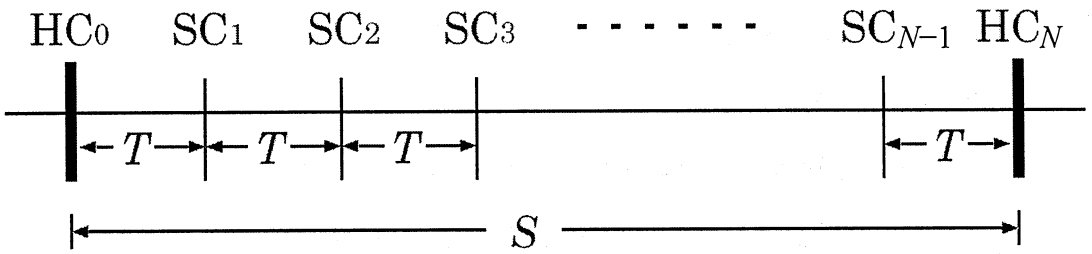


Figure 3.1: Soft checkpoints between hard checkpoints.

3.2 Two-Level Recovery Schemes

Suppose that S is an original execution time of one process or task which does not include the overheads of retries and checkpoint generations. Then, to tolerate some

failures, we consider two different types of checkpoints:

- *Soft checkpoint* (SC) can recover from some kinds of failures and its overhead is small.
- *Hard checkpoint* (HC) can recover from any kinds of failures and its overhead is large.

We propose the two-level recovery scheme with the following assumptions:

1. The original execution time of one process is S ($0 < S < \infty$). We divide S equally into N time intervals where $T \equiv S/N$, and take $(N-1)$SC every at times kT ($k = 1, 2, \cdots, N-1$), and two HC at time 0 and time NT, *i.e.*, SC_1, SC_2, $\cdots$, SC_{N-1} are set up between HC_0 and HC_N (Figure 3.1).
2. Failures of the process occur at constant rate λ ($\lambda > 0$), *i.e.*, the process has a failure distribution $F(t) = 1 - e^{-\lambda t}$ and $\bar{F}(t) \equiv 1 - F(t) = e^{-\lambda t}$.
3. If failures occur between HC_0 and SC_1, then the process is rolled back to HC_0 and begins its re-execution. If failures occur between SC_j and SC_{j+1} ($j = 1, 2, \cdots, N-1$), then the process is rolled back to SC_j where $SC_N \equiv HC_N$:
   (a) The process can recover from failures with probability q ($0 \leq q \leq 1$) and begins its re-execution from SC_j.
   (b) The process cannot recover with probability $1 - q$, and further, is rolled back to HC_0 and begins its re-execution.
4. If there is no failure between SC_j and SC_{j+1} ($j = 0, 1, \cdots, N-1$) where $SC_0 \equiv HC_0$, the process goes forward and begins its execution from SC_{j+1}.
5. The process ends when it attains to HC_N.

3.3 Performance Analysis

We define the following states of the process:

State 0: The process begins to execute its processing from HC_0.

State j: The process begins to execute its processing from SC_j ($j = 1, 2, \cdots, N-1$).

State N: The process attains to HC_N and ends.

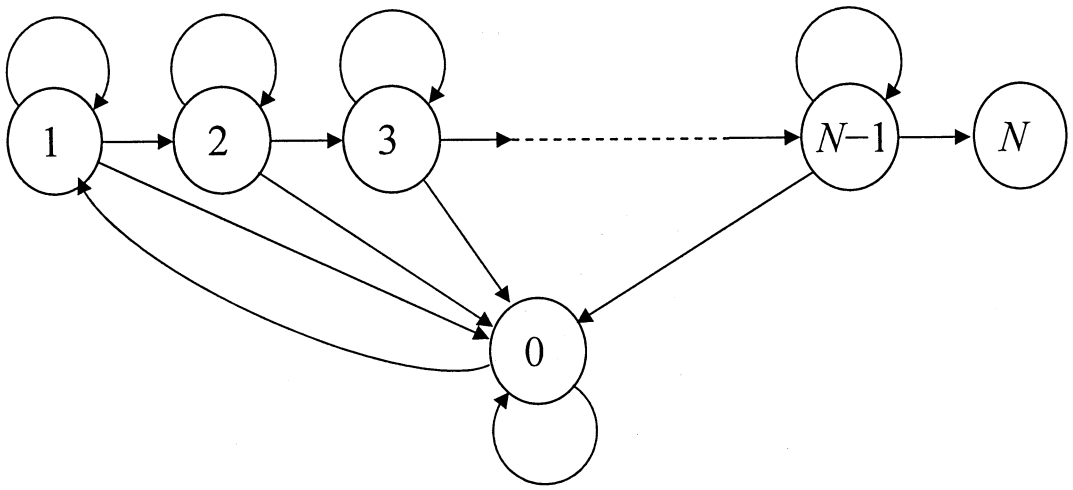


Figure 3.2: Transition diagram between states.

The process states defined above form a Markov renewal process [27] in which

State N is an absorbing state. All states are regeneration points and the transition diagram between states is shown in Figure 3.2. Let $Q_{ij}(t)$ ($i, j = 0, 1, 2, \cdots, N$) be one-step transition probabilities of a Markov renewal process. Then, by the similar method of Yasui *et al.* [43], mass functions $Q_{ij}(t)$ from State i at time 0 to State j at time t are

$$Q_{00}(t) = \int_0^t F(u)\mathrm{d}D(u), \tag{3.1}$$

$$Q_{jj+1}(t) = \int_0^t \overline{F}(u)\mathrm{d}D(u) \qquad (j = 0,1,\cdots N-1), \tag{3.2}$$

$$Q_{jj}(t) = q\int_0^t F(u)\mathrm{d}D(u) \qquad (j = 0,1,\cdots N-1), \tag{3.3}$$

$$Q_{j0}(t) = (1-q)\int_0^t F(u)\mathrm{d}D(u) \qquad (j = 0,1,\cdots N-1), \tag{3.4}$$

where $D(t)$ is a degenerate distribution placing unit mass at T, *i.e.*, $D(t) \equiv 1$ for $t \geq T$, and 0 for $t < T$.

Further, let $\varphi(s)$ be the Laplace-Stieltjes transform of any function $\Phi(t)$, *i.e.*,

$$\varphi(s) \equiv \int_0^\infty \mathrm{e}^{-st}\mathrm{d}\Phi(t) \quad \text{for } s \geq 0.$$

Then, the LS transforms of $Q_{ij}(t)$ are, from (3.1) − (3.4),

$$q_{00}(s) = \mathrm{e}^{-sT}F(T), \tag{3.5}$$

$$q_{jj+1}(s) = \mathrm{e}^{-sT}\overline{F}(T) \qquad (j = 0,1,\cdots N-1), \tag{3.6}$$

$$q_{jj}(s) = \mathrm{e}^{-sT}qF(T) \qquad (j = 0,1,\cdots N-1), \tag{3.7}$$

$$q_{j0}(s) = \mathrm{e}^{-sT}(1-q)F(T) \qquad (j = 0,1,\cdots N-1). \tag{3.8}$$

Denoting $H_{0N}(t)$ by the first-passage time distribution from State 0 to State N, its

LS transform is given by

$$h_{0N}(s) = q_{01}(s)\frac{q_{12}(s)}{1-q_{11}(s)} \times \cdots \times \frac{q_{N-1N}(s)}{1-q_{N-1N-1}(s)}$$

$$+ \left\{ q_{00}(s) + \sum_{j=1}^{N-1} \left[q_{01}(s)\frac{q_{12}(s)}{1-q_{11}(s)} \times \cdots \times \frac{q_{j0}}{1-q_{jj}(s)} \right] \right\} h_{0N}(s). \qquad (3.9)$$

To simplify equations, we put that $q_{j0} \equiv a_0(s)$, $q_{jj}(s) \equiv a_1(s)$ and $q_{jj+1}(s) \equiv a_2(s)$. Then, we easily have that $q_{00}(s) = a_0(s) + a_1(s)$ and $a_0(0) + a_1(0) + a_2(0) = 1$. Using these notations and solving (3.9) for $h_{0N}(s)$,

$$h_{0N}(s) = \frac{a_2(s)\left[\frac{a_2(s)}{1-a_1(s)}\right]^{N-1}}{1-a_0(s)-a_1(s)-\left[\frac{a_0(s)a_2(s)}{1-a_1(s)-a_2(s)}\right]\left\{1-\left[\frac{a_2(s)}{1-a_1(s)}\right]^{N-1}\right\}}. \qquad (3.10)$$

It is evident that $h_{0N}(0) = 1$. Thus, the mean first-passage time from State 0 to State N is

$$l_{0N} \equiv \lim_{s\to 0}\frac{1-h_{0N}(s)}{s}$$

$$= \frac{T}{\overline{F}(T)}\sum_{j=0}^{N-1}\left[\frac{\overline{F}(T)}{1-qF(T)}\right]^{j-(N-1)} \qquad (N = 1,2,\cdots). \qquad (3.11)$$

Moreover, the LS transform of the expected number of returning to State 0 is given by a renewal equation

$$m_H(s) = \left[q_{00}(s) + \sum_{j=1}^{N-1} q_{01}(s)\frac{q_{12}(s)}{1-q_{11}(s)} \times \cdots \times \frac{q_{j0}(s)}{1-q_{jj}(s)} \right][1 + m_H(s)]. \qquad (3.12)$$

Solving this equation and arranging it,

$$m_H(s)=\frac{a_0(s)+a_1(s)+a_0(s)\sum_{j=1}^{N-1}\left[\frac{a_2(s)}{1-a_1(s)}\right]^j}{1-a_0(s)-a_1(s)-a_0(s)\sum_{j=1}^{N-1}\left[\frac{a_2(s)}{1-a_1(s)}\right]^j}. \tag{3.13}$$

Thus, the expected number of returning to State 0 is

$$\begin{aligned} M_H &\equiv \lim_{s\to 0} m_H(s) \\ &= \frac{1-\overline{F}(T)\left[\frac{\overline{F}(T)}{1-qF(T)}\right]^{N-1}}{\overline{F}(T)\left[\frac{\overline{F}(T)}{1-qF(T)}\right]^{N-1}} \quad (N=1,2,\cdots). \end{aligned} \tag{3.14}$$

Note that M_H represents the expected number of rollbacks to HC until the process ends.

Next, we compute the expected number of rollbacks to SC. The expected numbers of returning to State j when the process transits from State j to State $j+1$ and State 0 are, respectively,

$$\sum_{i=1}^{\infty} i\left[q_{jj}(s)\right]^i q_{jj+1}(s)=\frac{q_{jj}(s)q_{jj+1}(s)}{\left[1-q_{jj}(s)\right]^2},$$

$$\sum_{i=1}^{\infty} i\left[q_{jj}(s)\right]^i q_{j0}(s)=\frac{q_{jj}(s)q_{j0}(s)}{\left[1-q_{jj}(s)\right]^2}.$$

Thus, the LS transform of the expected number of returning to State j (j = 1, 2, ···,$N-1$) is

$$m_S(s)=\sum_{j=1}^{N-1} q_{01}(s)\frac{q_{12}(s)}{1-q_{11}(s)}\times\cdots\times\frac{q_{j-1j}(s)}{1-q_{j-1j-1}(s)}\frac{q_{jj}(s)\left[q_{jj+1}(s)+q_{j0}(s)\right]}{\left[1-q_{jj}(s)\right]^2}$$

$$+\left[q_{00}(s)+\sum_{j=1}^{N-1}q_{01}(s)\frac{q_{12}(s)}{1-q_{11}(s)}\times\cdots\times\frac{q_{j-1j}(s)}{1-q_{j-1j-1}(s)}\frac{q_{j0}(s)}{1-q_{jj}(s)}\right]m_S(s).$$

(3.15)

Solving this equation,

$$m_S(s)=\frac{\dfrac{a_1(s)a_2(s)[a_0(s)+a_2(s)]}{1-a_1(s)}\dfrac{1-\left[\dfrac{a_2(s)}{1-a_1(s)}\right]^{N-1}}{1-a_1(s)-a_2(s)}}{1-a_0(s)-a_1(s)-a_0(s)a_2(s)\dfrac{1-\left[\dfrac{a_2(s)}{1-a_1(s)}\right]^{N-1}}{1-a_1(s)-a_2(s)}}. \tag{3.16}$$

Therefore, the expected number of returning to State j (j = 1, 2, ···, $N-1$) is, for $0 \le q <$ 1,

$$M_S \equiv \lim_{s\to 0} m_S(s)=\frac{q}{1-q}\frac{1-\left[\dfrac{\overline{F}(T)}{1-qF(T)}\right]^{N-1}}{\left[\dfrac{\overline{F}(T)}{1-qF(T)}\right]^{N-1}} \quad (N=1, 2, \cdots), \tag{3.17}$$

and for q = 1,

$$M_S=\frac{(N-1)F(T)}{\overline{F}(T)} \qquad (N=1, 2, \cdots). \tag{3.18}$$

Note also that M_S represents the expected number of rollbacks to SC until the process ends.

3.4 Expected Overhead

Assume that the overheads for rollbacks to HC and SC are C_H and C_S ($C_S < C_H$), respectively, and C_T for setting up one SC. The other overheads except C_H, C_S and C_T would be neglected because they are small. Then, the total expected overhead is, from (3.11), (3.14), and (3.17),

$$C_1(N) \equiv l_{0N} + C_H M_H + C_S M_S + (N-1)C_T - S$$
$$= \frac{T + C_H + [T + qF(T)C_S]\sum_{j=1}^{N-1}\left(\frac{\overline{F}(T)}{1-qF(T)}\right)^j}{\overline{F}(T)\left(\frac{\overline{F}(T)}{1-qF(T)}\right)^{N-1}} - C_H + (N-1)C_T - S$$

(N=1, 2, ···), (3.19)

where note that $\Sigma_{j=1}^{0} \equiv 0$. In particular, when $N = 1$, *i.e.*, SC is not set up,

$$C_1(1) = \frac{S + C_H F(S)}{\overline{F}(S)} - S. \tag{3.20}$$

When $q = 1$, *i.e.*, SC can recover from all failures,

$$C_1(N) = \frac{S + F(T)[C_H + (N-1)C_S]}{\overline{F}(T)} + (N-1)C_T - S, \tag{3.21}$$

and when $q = 0$, *i.e.*, SC cannot recover from any failures,

$$C_1(N)=\frac{1-[\overline{F}(T)]^N}{[\overline{F}(T)]^N}\left(\frac{T}{F(T)}+C_H\right)+(N-1)C_T-S. \tag{3.22}$$

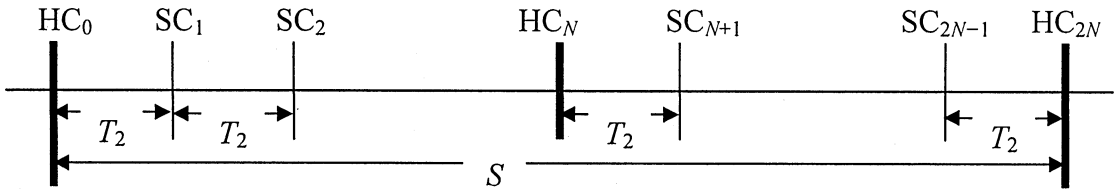


Figure 3.2: Soft checkpoints between three hard checkpoints.

Next, we divide S equally into $2N$ time intervals where $T_2 \equiv S/(2N)$ (N=1, 2, ···), and set up three HCs at time 0, NT_2, $2NT_2$, and SCs every at time kT_2 except for k=0, N, $2N$ (Figure 3.2). Then, by the similar method of obtaining $C_1(N)$ in (3.19), the total expected overhead is

$$C_2(N)=2\left\{\frac{T_2+C_H+[T_2+qF(T_2)]C_S\sum_{j=1}^{N-1}\left[\frac{\overline{F}(T_2)}{1-qF(T_2)}\right]^j}{\overline{F}(T_2)\left[\frac{\overline{F}(T_2)}{1-qF(T_2)}\right]^{N-1}}+NC_T\right\}$$

(N=1, 2, ···). (3.23)

Generally, when we divide S equally into kN time intervals where $T_k \equiv S/(kN)$, and set up HCs at times kN (k=0, 1, 2, ···) and SCs between HCs, the total expected

overhead is

$$C_k(N) = k\left\{\frac{T_k + C_H + \left[T_k + qF(T_k)\right]C_S\left[\sum_{j=1}^{N-1}\left[\frac{\overline{F}(T_k)}{1-qF(T_k)}\right]^j\right.}{\overline{F}(T_k)\left[\frac{\overline{F}(T_k)}{1-qF(T_k)}\right]^{N-1}} + NC_T\right\}$$

(N=1, 2, ···, k=1, 2, ···). (3.24)

3.5 Numerical Examples

We compute an optimal number N^* of SC which minimizes the total expected overhead $C_1(N)$. Since $F(t) = 1-e^{-\lambda t}$ and $T=S/N$, Equation (3.19) becomes

$$\lambda C_1(N) = \frac{\frac{\lambda S}{N} + \lambda C_H + \left[\frac{\lambda S}{N} + q\left(1-e^{-\lambda S/N}\right)\right]\lambda C_S\left[\sum_{j=1}^{N-1}\left[\frac{e^{-\lambda S/N}}{1-q\left(1-e^{-\lambda S/N}\right)}\right]^j\right.}{e^{-\lambda S/N}\left[\frac{e^{-\lambda S/N}}{1-q\left(1-e^{-\lambda S/N}\right)}\right]^{N-1}}$$

$$-\lambda C_H + (N-1)\lambda C_T - \lambda S \qquad (N=1, 2, \cdots). \qquad (3.25)$$

Table 3.1 gives the optimal number N^* for q = 0.0, 0.2, 0.4, 0.6, 0.8, 1.0 and λC_T =0.0001, 0.0005, 0.001, 0.005 when λS = 0.1, λC_H = 0.001 and λC_S = 0.0002. For example, when q = 0.8 and $\lambda C_T = 5 \times 10^{-4}$, the optimal number is N^* = 4 and the resulting overhead is $\lambda C_1(4) = 4.866 \times 10^{-3}$, *i.e.*, we should take 3SCs between HCs. It is evident that optimal N^* decrease and their overheads $C_1(N^*)$ increase as the overhead

Table 3.1: Optimal number N^* and total expected overhead $\lambda C_1(N^*)$ when $\lambda S = 0.1$, $\lambda C_H = 0.001$, $\lambda C_S = 0.0002$.

	$\lambda C_T = 1 \times 10^{-4}$		$\lambda C_T = 5 \times 10^{-4}$		$\lambda C_T = 1 \times 10^{-3}$		$\lambda C_T = 5 \times 10^{-3}$	
q	N^*	$\lambda C_1(N^*)\times 10^3$	N^*	$\lambda C_1(N^*)\times 10^3$	N^*	$\lambda C_1(N^*)\times 10^3$	N^*	$\lambda C_1(N^*)\times 10^3$
0.0	7	6.629	3	8.039	2	8.927	1	10.622
0.2	8	5.687	4	7.280	3	8.312	1	10.622
0.4	9	4.746	4	6.470	3	7.588	1	10.622
0.6	9	3.803	4	5.665	3	6.868	1	10.622
0.8	10	2.867	4	4.866	3	6.151	1	10.622
1.0	10	1.933	5	4.056	3	5.437	2	10.189

C_T increases. This also indicates that N^* increase as q increase, because SC becomes useful to recover from failures. Further, the overhead of two-level schemes is smaller than that of one-level scheme in the case of $N = 1$. From this example, two-level recovery schemes would achieve better performances as compared to one-level scheme.

We give the optimal number N^* which minimizes $C_k(N)$ in (3.24) when λS=0.1, λC_H=0.0003, λC_S=0.0002, λC_T=0.0001 and q=0.8 in Table 3.2. We find the optimal number k=2 and N^*=5.

Table 3.2: Optimal number N^* and total expected overhead $\lambda C_k(N^*)$ when λS=0.1, λC_H=0.0003 , λC_S=0.0002, λC_T=0.0001, q=0.8.

k	N^*	$\lambda C_k(N^*)$
1	9	0.1032
2	5	0.1030
3	3	0.1031
4	2	0.1034
5	2	0.1036
6	2	0.1039
7	1	0.1042

3.6 Conclusions

We have taken two types of checkpoints as the fault tolerance technique of recovery mechanism and obtained the total expected overhead of one cycle from HC to HC, using Markov renewal processes. Further, we have computed numerically the optimal interval of SC between HC which minimizes the total overhead. It has been shown in a numerical example that two-level recovery schemes would be more useful to recover from failures. Moreover, by making suitable modification and further extension, this model would be applied to storage management in mobile environments [36], and other computer and database information systems.

Chapter 4

Checkpoint Intervals for Error Detection by Multiple Modular Redundancies

This chapter considers multiple modular redundant systems as the recovery techniques of error detection and error masking on the finite process execution, and discusses analytically optimal checkpoint intervals. Introducing the overheads of comparison and decision by majority, an error occurrence rate and a native execution time of the process, we obtain the mean times to the completion of the processes for multiple modular systems, using renewal equations, and derive analytically optimal checkpoint intervals which minimize them. Further, it is shown numerically that what a majority decision system is optimal.

4.1 Introduction

In computer systems, some errors often occur due to noises, human errors, hardware faults, and so on. To attain the accuracy of the computing, it is important to detect and /or mask such errors by fault tolerant computing techniques [4, 12].

This chapter considers the redundant techniques of error detection and error masking on a finite process execution. Firstly, an error detection of the process can be made by two independent modules where they compare two results at suitable checkpoint times. If their results do not match with each other, we go back to the newest checkpoint and make a retrial of the processes. Secondly, a majority decision system with multiple modules is adopted as the technique of an error masking and the result is decided by its majority of modules. In this case, we determine numerically what a majority system is optimal.

In such situations, if we compare results frequently, then the time required for rollback could decrease, however, the total overhead for comparisons at checkpoints would increase. Thus, this is one kind of trade-off problems how to decide an optimal checkpoint interval.

Several studies of deciding a checkpoint frequency have been discussed for the hardware redundancy above. Pradhan and Vaidya [29] evaluated the performance and reliability of a duplex system with a spare processor. Ziv and Bruck [44, 45] considered the checkpoint schemes with task duplication and evaluated the

performance of schemes. Kim and Shin [8] derived the optimal instruction-retry period which minimizes the probability of the dynamic failure on the triple modular redundant controller.

This chapter considers a double modular redundancy as redundant techniques of error detection and summarizes the results [19, 20]. Next, we consider a redundant system of a majority decision with $(2n+1)$ modules as an error masking system, and compute the mean time to completion of the process and decide numerically what a majority system is optimal.

4.2 Multiple Modular System

Suppose that S is a native execution time of the process which does not include the overheads of retries and checkpoint generations. Then, we divide S equally into N parts and create a checkpoint at planned times $kT(k=1, 2, \cdots, N-1)$ where $S=NT$ (Figure 4.1).

To detect errors, we firstly provide two independent modules where they compare two results at periodic checkpoint times. If two results agree with each other, two processes are correct and go forward. However, if two results do not agree, it is judged that some errors have occurred. Then, we make a rollback operation to the newest checkpoint and a retry of the processes. The process completes when two processes are succeeded in all intervals above.

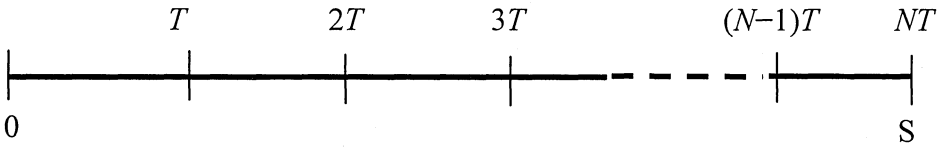


Figure 4.1: Checkpoint intervals.

Let us introduce a constant overhead C_1 for the comparison of two results. We neglect any failures of the system caused by common mode faults to make clear an error detection of the processes. Further, it is assumed that some errors of one process occur at constant rate λ, *i.e.*, the probability that any errors do not occur during $(0,t]$ is given by $e^{-\lambda t}$. Thus, the probability that two processes have no error during $(0,t]$ is $\bar{F}_1(T)=e^{-2\lambda T}$ [27].

The mean time $L_1(N)$ to completion of the process is the summation of the processing times and the overhead C_1 of comparison of two processes. From the assumption that two processes are rolled back to the previous checkpoint when an error has been detected at a checkpoint, the mean execution time of the process for one checkpoint interval $(0,T]$ is given by a renewal equation:

$$L_1(1) = (T + C_1)e^{-2\lambda T} + [T + C_1 + L_1(1)](1 - e^{-2\lambda T}), \tag{4.1}$$

and solving it,

$$L_1(1) = (T + C_1)e^{2\lambda T}. \tag{4.2}$$

Thus, the mean time to completion of the process is

$$L_1(N) \equiv NL_1(1) = N(T + C_1)e^{2\lambda T} = (S + NC_1)e^{2\lambda S/N}. \tag{4.3}$$

We seek an optimal number N_1^* which minimizes $L_1(N)$ for a specified S. Evidently, $L_1(\infty)=\infty$ and

$$L_1(1) = (S + C_1)e^{2\lambda S}. \tag{4.4}$$

Thus, there exists a finite number $N_1^*(1 \le N_1^* < \infty)$. However, it would be difficult to find analytically N_1^* which minimizes $L_1(N)$ in (4.3). Putting $T=S/N$ in (4.3) and rewriting it by the function T,

$$L_1(T) = S\left(1 + \frac{C_1}{T}\right)e^{2\lambda T} \qquad (0 < T \le S). \tag{4.5}$$

It is evident that $L_1(0)=\lim_{T\to 0}L_1(T)=\infty$ and $L_1(S)$ is given by (4.4). Thus, there exists an optimal $\tilde{T}_1$ $(0 < \tilde{T}_1 \le S)$ which minimizes $L_1(T)$ in (4.5). Differentiating $L_1(T)$ with respect to T and setting it equal to zero,

$$T^2 + C_1 T - \frac{C_1}{2\lambda} = 0. \tag{4.6}$$

Solving it with T,

$$\tilde{T}_1 = \frac{C_1}{2}\left[\sqrt{1 + \frac{2}{\lambda C_1}} - 1\right]. \tag{4.7}$$

Therefore, we have the following optimal interval number N_1^* [18]:

(i) If $\tilde{T}_1 < S$, we put $[S / \tilde{T}_1]= N$, where $[x]$ denotes the greatest integer contained in x, and calculate $L_1(N)$ and $L_1(N+1)$ from (4.3). If $L_1(N)\le L_1(N+1)$ then $N_1^*=N$ and $T_1^* = S / N_1^*$, and conversely, if $L_1(N+1)< L_1(N)$ then $N_1^*=N+1$.

(ii) If $\tilde{T}_1 \geq S$, *i.e.*, we should make no checkpoint until time S then $N_1^* = 1$, and the mean time is given in (4.4).

Note that $\tilde{T}_1$ in (4.7) does not depend on S. Thus, if S is very large, is changed greatly or is unclear, then we may adopt $\tilde{T}_1$ as an approximate checkpoint time.

Further, the mean time for one checkpoint interval per this interval is

$$\tilde{L}_1(T) \equiv \frac{L_1(1)}{T} = \left(1 + \frac{C_1}{T}\right) e^{2\lambda T}. \tag{4.8}$$

Thus, the optimal time which minimizes $\tilde{L}_1(T)$ also agrees with $\tilde{T}_1$ in (4.7).

Next, consider a redundant system of a majority decision with $(2n+1)$ modules as an error masking system, *i.e.*, $(n+1)$-out-of-$(2n+1)$ system $(n=1,2,\cdots)$. If more than $(n+1)$ results of $(2n+1)$ modules agree, the system is correct. Then, the probability that the system is correct during $(0, T]$ is

$$\overline{F}_{n+1}(T) = \sum_{k=n+1}^{2n+1} \binom{2n+1}{k} \left(e^{-\lambda T}\right)^k \left(1 - e^{-\lambda T}\right)^{2n+1-k}. \tag{4.9}$$

Thus, the mean time to completion of the process is

$$L_{n+1}(N) = \frac{N(T + C_{n+1})}{\overline{F}_{n+1}(T)} \qquad (n = 1,2,\cdots), \tag{4.10}$$

where C_{n+1} is the overhead of a majority decision of $(2n+1)$ modules.

Table 4.1: Optimal checkpoint number N_1^*, interval λT_1^* and mean time $\lambda L_1(N_1^*)$ for a double modular system when $\lambda S=10^{-1}$.

$\lambda C_1\times10^3$	$\lambda\tilde{T}_1\times10^2$	N_1^*	$\lambda L_1(N_1^*)\times10^2$	$\lambda T_1^*\times10^2$
0.5	1.556	6	10.650	1.67
1.0	2.187	5	10.929	2.00
1.5	2.665	4	11.143	2.50
2.0	3.064	3	11.331	3.33
3.0	3.726	3	11.715	3.33
4.0	4.277	2	11.936	5.00
5.0	4.756	2	12.157	5.00
10.0	6.589	2	13.435	5.00
20.0	9.050	1	14.657	10.00
30.0	10.839	1	15.878	10.00

4.3 Numerical Examples

We show numerical examples of optimal checkpoint intervals for a double modular system when $\lambda S=10^{-1}$. Table 4.1 presents $\lambda\tilde{T}_1$ in (4.7), optimal number N_1^*, λT_1^* and $\lambda L_1 T(N_1^*)$ for λC_1=0.5, 1.5, 2, 3, 4, 5, 10, 20, 30($\times10^{-3}$). For example, when $\lambda=10^{-2}$ (1/sec), $C_1=10^{-1}$(sec) and S=10.0(sec), the optimal number is N_1^*=5, the optimal interval

is $T_1^* = S / N_1^* = 2.0$ (sec), and the resulting mean time is $L_1(5)$=10.929 (sec), which is longer about 9.3 percent than S.

Next, we consider the problem what a majority system is optimal. When the overhead of comparison of two processes is C_1, it is assumed that the overhead C_{n+1} of an $(n+1)$-out-of-$(2n+1)$ system is given by $C_{n+1} \equiv \binom{2n+1}{2} C_1$ (n=1, 2, ···). This is to select and compare 2 from each of $(2n+1)$ processes. Table 4.2 presents optimal number N^*_{n+1} and the resulting mean time $\lambda L_{n+1}(N^*_{n+1})\times 10^2$ for n=1, 2, 3, 4 when $\lambda C_1 = 0.1\times 10^{-3}$, 0.5×10^{-3}. When $\lambda C_1 = 0.5\times 10^{-3}$, the optimal checkpoint number is N_3^*=2 and $\lambda L_3(2) = 10.37\times 10^{-2}$ which is the smallest among these systems, that is, a 2-out-of-3 system is optimal. The mean times for n=1, 2 are smaller than 10.65×10^{-2} for a double modular system.

Table 4.2: Optimal checkpoint number N^*_{n+1} and mean time $\lambda L_{n+1}(N^*_{n+1})$ for $(n+1)$−out−of−$(2n+1)$ system when $\lambda C_1 = 0.5\times 10^{-3}$, $\lambda C_1 = 0.1\times 10^{-3}$ and $\lambda S = 10^{-1}$.

	$\lambda C_1 = 0.1\times 10^{-3}$		$\lambda C_1 = 0.5\times 10^{-3}$	
n	N^*_{n+1}	$\lambda L_{n+1}(N^*_{n+1})\times 10^2$	N^*_{n+1}	$\lambda L_{n+1}(N^*_{n+1})\times 10^2$
1	3	10.12 *	2	10.37 *
2	1	10.18	1	10.58
3	1	10.23	1	11.08
4	1	10.36	1	11.81

4.4 Conclusions

In this chapter, the simple checkpoint models are formulated for error detection and error masking by redundancy on the finite process execution. We have obtained the mean times to completion of the process for a double modular and a majority decision systems. The optimal checkpoint intervals which minimize them are derived analytically. In general, the overhead C_1 and the native execution time S would be estimated for real systems. Therefore, the establishment of checkpoint schemes would be depend on whether we can accurately estimate error rates or not.

Chapter 5

Sequential Checkpoint Intervals for Error Detection

This chapter adopts a modular redundant system as the recovery techniques of error detection and error masking on the finite process execution: Checkpoints are placed at sequential times $T_k(k=1, 2, \cdots, N)$. We consider two checkpoint models where error rates during the interval (T_{k-1}, T_k) $(k=1, 2, \cdots, N)$ increase with the number of checkpoints and with the original execution time. The mean times to the completion of the process are obtained analytically, and optimal checkpoint intervals which minimize them are computed numerically by solving simultaneous equations. Further, approximate checkpoint intervals are derived by denoting that the probability of the occurrence of errors during $(T_{k-1}, T_k]$ is constant. It is shown that the approximate method is simple and these intervals give good approximations to optimal ones.

5.1 Introduction

This chapter considers a general modular system of error detection and error masking on a finite process execution: Suppose that checkpoints are placed at sequential times $T_k(k=1, 2, \cdots, N)$, where $T_N \equiv S$ (Figure 5.1). First, it is assumed that error rates during the interval $(T_{k-1}, T_k]$ $(k=1, 2, \cdots, N)$ increase with the number k of checkpoints. The mean times to completion of the process are obtained, and optimal checkpoint intervals which minimize them are derived by solving simultaneous equations.

Further, approximate checkpoint intervals are given by denoting that the probability of the occurrence of errors during $(T_{k-1}, T_k]$ is constant. Secondly, it is assumed that error rates during $(T_{k-1}, T_k]$ increase with the original execution time, irrespective of the number of recoveries. Optimal checkpoint intervals which minimize the mean time to completion of the process are discussed, and their approximate times are shown. Numerical examples of optimal checkpoint times for a double modular system are presented. It is shown numerically that the approximate method is simple and these intervals give good approximations to optimal ones.

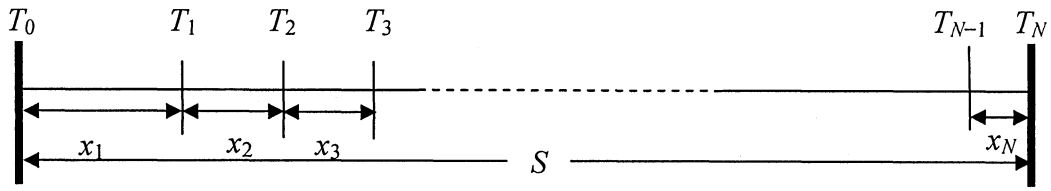


Figure 5.1: Sequential checkpoint interval.

5.2 Sequential Checkpoint Interval

The error rate λ is constant and S is divided into an equal part. In general, error rates would be increasing with time, and so that, their intervals should be decreasing with their number. We assume for the simplicity of the model that error rates are increasing with the number of checkpoints.

Suppose that S is a native execution time of the process which does not include the overheads of retries and checkpoint generations. Then, we divide S into N parts and create a checkpoint at sequential times $T_k(k = 1, 2, \cdots, N-1)$, where $T_0 \equiv 0$ and $T_N \equiv S$ (Figure 5.1)[17]. Let us introduce a constant overhead C for the comparison of a modular system. Further, the probability that the system has no error during the interval $(T_{k-1}, T_k]$ is $F_k(T_k - T_{k-1})$, irrespective of other intervals and rollback operation. Then, the mean time $L_1(N)$ to completion of the process is the summation of the

processing times and the overhead C for the comparison of a modular system.

From the assumption that the system is rolled back to the previous checkpoint when some error has been detected at a checkpoint, the mean execution time of the process for the interval $(T_{k-1}, T_k]$ is

$$L_1(k) = (T_k - T_{k-1} + C)\bar{F}_k(T_k - T_{k-1}) + [T_k - T_{k-1} + C + L_1(k)]F_k(T_k - T_{k-1}), \tag{5.1}$$

and solving it,

$$L_1(k) = \frac{T_k - T_{k-1} + C}{\bar{F}_k(T_k - T_{k-1})} \qquad (k=1, 2, \cdots, N), \tag{5.2}$$

where $F_k(t) \equiv 1 - \bar{F}_k(t)$. Thus, the mean time to completion of the process is

$$L_1(N) \equiv \sum_{k=1}^{N} L_1(k) = \sum_{k=1}^{N} \frac{T_k - T_{k-1} + C}{\bar{F}_k(T_k - T_{k-1})} \qquad (N=1, 2, \cdots). \tag{5.3}$$

We find optimal times T_k which minimize $L_1(N)$ for a specified N. Let $f_k(t)$ be a density function of $F_k(t)$ and $r_k(t) \equiv f_k(t)/\bar{F}_k(t)$ that is the failure rate of $F_k(t)$. Then, differentiating $L_1(N)$ with respect to T_k and setting it equal to zero,

$$\frac{1}{\bar{F}_k(T_k - T_{k-1})}[1 + (T_k - T_{k-1} + C)r_k(T_k - T_{k-1})]$$

$$= \frac{1}{\bar{F}_{k+1}(T_{k+1} - T_k)}[1 + (T_{k+1} - T_k + C)r_{k+1}(T_{k+1} - T_k)]. \tag{5.4}$$

Setting that $x_k \equiv T_k - T_{k-1}$ and rewriting (5.4) as a function of x_k,

$$\frac{1}{\bar{F}_k(x_k)}[1 + (x_k + C)r_k(x_k)] = \frac{1}{\bar{F}_{k+1}(x_{k+1})}[1 + (x_{k+1} + C)r_{k+1}(x_{k+1})] \quad (k=1,2, \cdots, N-1).$$

(5.5)

Next, suppose that $\bar{F}_k(t)=e^{-\lambda_k t}$, *i.e.*, an error rate during $(T_{k-1},T_k]$ is constant λ_k which increases with k. Then, Equation (5.5) is rewritten as

$$\frac{1+\lambda_{k+1}(x_{k+1}+C)}{1+\lambda_k(x_k+C)}-e^{(\lambda_k x_k-\lambda_{k+1}x_{k+1})}=0. \tag{5.6}$$

It is easily noted that $\lambda_{k+1}x_{k+1}\le\lambda_k x_k$, and hence, $x_{k+1}\le x_k$ since $\lambda_{k+1}\le\lambda_k$.

In particular, when $\lambda_k\equiv\lambda$ for $k=1, 2, \cdots, N$, Equation (5.6) becomes

$$\frac{1+\lambda(x_{k+1}+C)}{1+\lambda(x_k+C)}-e^{\lambda(x_k-x_{k+1})}=0. \tag{5.7}$$

Since $x_{k+1}\le x_k$, we have that $x_{k+1}\ge x_k$ from (5.7), *i.e.*, it is easily proved that a solution to satisfy (5.7) is restricted only to $x_{k+1}=x_k\equiv T$, irrespective of the interval number k. Then, the mean time to completion of the process is

$$L_1(N)=(S+NC)e^{\lambda S/N}. \tag{5.8}$$

If $\lambda_{k+1}>\lambda_k$, then $x_{k+1}<x_k$ from (5.6). Let $Q(x_{k+1})$ be the left-hand side of (5.7) for a fixed x_k. Then, $Q(x_{k+1})$ is strictly increasing from

$$Q(0)=\frac{1+\lambda_{k+1}C}{1+\lambda_k(x_k+C)}-e^{\lambda_k x_k}$$

to $Q(x_k)>0$. Thus, if $Q(0)<0$, then an optimal x_{k+1}^* $(0<x_{k+1}^*<x_k)$ to satisfy (5.6) exists uniquely, and if $Q(0)\ge 0$, then $x_{k+1}^*=S-T_k$.

Therefore, noting that $T_0=0$ and $T_N=S$, we have the following result:

(i) When $N=1$ and $T_1=S$, the mean time is

$$L_1(1)=(S+C)e^{\lambda_1 S}. \tag{5.9}$$

(ii) When $N=2$, from (5.5),

$$[1+\lambda_1(x_1+C)]e^{\lambda_1 x_1}-[1+\lambda_2(S-x_1+C)]e^{\lambda_2(S-x_1)}=0. \tag{5.10}$$

Letting $Q_1(x_1)$ be the left-hand side of (5.10), it is strictly increasing from $Q_1(0)<0$ to

$$Q_1(S)=[1+\lambda_1(S+C)]e^{\lambda_1 S}-(1+\lambda_2 C).$$

Hence, if $Q_1(S)>0$, then $x_1^*=T_1^*$ $(0<T_1^*<S)$ to satisfy (10) exists uniquely, and conversely, if $Q_1(S)\le 0$ then $x_1^*=T_1^*=S$.

(iii) When N=3, we compute $x_k(k=1,2)$ which satisfy the simultaneous equations:

$$[1+\lambda_1(x_1+C)]e^{\lambda_1 x_1}=[1+\lambda_2(x_2+C)]e^{\lambda_2 x_2}, \tag{5.11}$$

$$[1+\lambda_2(x_2+C)]e^{\lambda_2 x_2}=[1+\lambda_3(S-x_1-x_2)]e^{\lambda_3(S-x_1-x_2)}. \tag{5.12}$$

(iv) When N=4, 5, ⋯, we compute x_k^* and $T_k=\sum_{j=1}^{k}x_j^*$ similarly.

We compute sequential checkpoint intervals $T_k(k=1, 2, \cdots, N)$ for a double modular system. It is assumed that $\lambda_k=2[1+0.1(k-1)]\lambda$ $(k=1, 2, \cdots)$, *i.e.*, an error rate increases by 10% of an original rate λ. Table 5.1 presents optimal sequential intervals T_k and the resulting mean times $L_1(N)$ for N=1, 2, ⋯, 9 when $\lambda S=10^{-1}$ and $\lambda C=10^{-3}$. In this case, the mean time is the smallest when N=5, *i.e.*, the optimal checkpoint number is $N^*=5$ and the checkpoint times T_k^* $(k=1, 2, 3, 4, 5)$ should be placed at 2.38, 4.53, 6.50, 8.32, 10.00(sec) for $\lambda=10^{-2}$(1/sec), and the mean time 11.009 is about 10% longer than an original execution time S=10. Further, all values of $x_k=T_k-T_{k-1}$ decrease with k because error rates increase with the number of checkpoints.

Table 5.1: Checkpoint intervals λT_k and mean time $\lambda L_1(N)$ when $\lambda_k=2[1+0.1(k-1)]\lambda$, $\lambda S=10^{-1}$ and $\lambda C=10^{-3}$.

N	1	2	3	4	5	6	7	8	9
$\lambda T_1\times10^2$	10.00	5.24	3.65	2.85	2.38	2.05	1.83	1.65	1.52
$\lambda T_2\times10^2$		10.00	6.97	5.44	4.53	3.91	3.48	3.15	2.89
$\lambda T_3\times10^2$			10.00	7.81	6.50	5.62	4.99	4.52	4.15
$\lambda T_4\times10^2$				10.00	8.32	7.19	6.39	5.78	5.31
$\lambda T_5\times10^2$					10.00	8.65	7.68	6.95	6.38
$\lambda T_6\times10^2$						10.00	8.88	8.03	7.37
$\lambda T_7\times10^2$							10.00	9.05	8.31
$\lambda T_8\times10^2$								10.00	9.18
$\lambda T_9\times10^2$									10.00
$\lambda L_1(N)$ $\times10^2$	12.3362	11.3266	11.0792	11.0095	11.0089	11.0423	11.0950	11.1596	11.2322

5.3 Approximation Method

It is very troublesome to solve simultaneous equations. We consider the following approximate checkpoint times: It is assumed that the probability that a modular system has no error during $(T_{k-1},T_k]$ is constant, *i.e.*, $\bar{F}_k(T_k-T_{k-1})\equiv q$ $(k=1, 2, \cdots, N)$. From this

assumption, we derive $T_k - T_{k-1} \equiv \overline{F}_k^{-1}(q)$ as a function of q. Substituting this $T_k - T_{k-1}$ into (5.3), the mean time to completion of the process is

$$L_1(N) = \sum_{k=1}^{N} \frac{\overline{F}_k^{-1}(q) + C}{q}. \tag{5.13}$$

We discuss an optimal q which minimizes $L_1(N)$.

For example, when $\overline{F}_k(t) = e^{-\lambda_k t}$,

$$e^{-\lambda_k (T_k - T_{k-1})} = q \equiv e^{-\tilde{q}},$$

and hence,

$$T_k - T_{k-1} = \frac{\tilde{q}}{\lambda_k}.$$

Since

$$\sum_{k=1}^{N} (T_k - T_{k-1}) = T_N = S = \tilde{q} \sum_{k=1}^{N} \frac{1}{\lambda_k},$$

we have

$$L_1(N) = e^{\tilde{q}} \left[\tilde{q} \sum_{k=1}^{N} \frac{1}{\lambda_k} + NC \right] = e^{\tilde{q}} (S + NC). \tag{5.14}$$

Therefore, we compute $\tilde{q}$ and $L_1(N)$ for a specified N. Comparing $L_1(N)$ for N=1, 2, ···, we obtain an optimal $\tilde{N}$ which minimizes $L_1(N)$ and $\tilde{q} = S / \sum_{k=1}^{N} (1/\lambda_k)$. Lastly, We may compute $\tilde{T}_k^* = \tilde{q} \sum_{j=1}^{k} (1/\lambda_j)$ (k=1, 2, ···, $\tilde{N}$−1) for an approximate optimal $\tilde{N}$ which minimizes $L_1(N)$.

Table 5.2: Mean time $\lambda L_1(N)$ for $\tilde{q}$ when $\lambda S=10^{-1}$ and $\lambda C=10^{-3}$.

N	$\tilde{q}$	$\lambda L_1(N)\times 10^2$
1	0.2000000	12.33617
2	0.1047619	11.32655
3	0.0729282	11.07923
4	0.0569532	11.00951
5	0.0473267	11.00888
6	0.0408780	11.04229
7	0.0362476	11.09496
8	0.0327555	11.15962
9	0.0300237	11.23222

Table 5.2 presents $\tilde{q} = S/\sum_{k=1}^{N}(1/\lambda_k)$ and $\lambda L_1(N)$ in (5.14) for N=1,2, ⋯, 9 under the same assumptions as those in Table 5.1. In this case, $\tilde{N}$=5=N^* and the mean time $L_1(5)$ is a little longer than that in Table 5.1. When $\tilde{N}$ =5, approximate optimal checkpoint times are $\lambda\tilde{T}_k^* \times 10^2$=2.37, 4.52, 6.49, 8.31, 10.00 that are a little shorter than those in Table 5.1. Such computations are much easier than to solve simultaneous equations.

It would be sufficient to adopt approximate checkpoint intervals as optimal ones in actual fields. Figure 5.2 draws the mean times $\lambda L_1(N)$ for $1 \le N \le 20$.

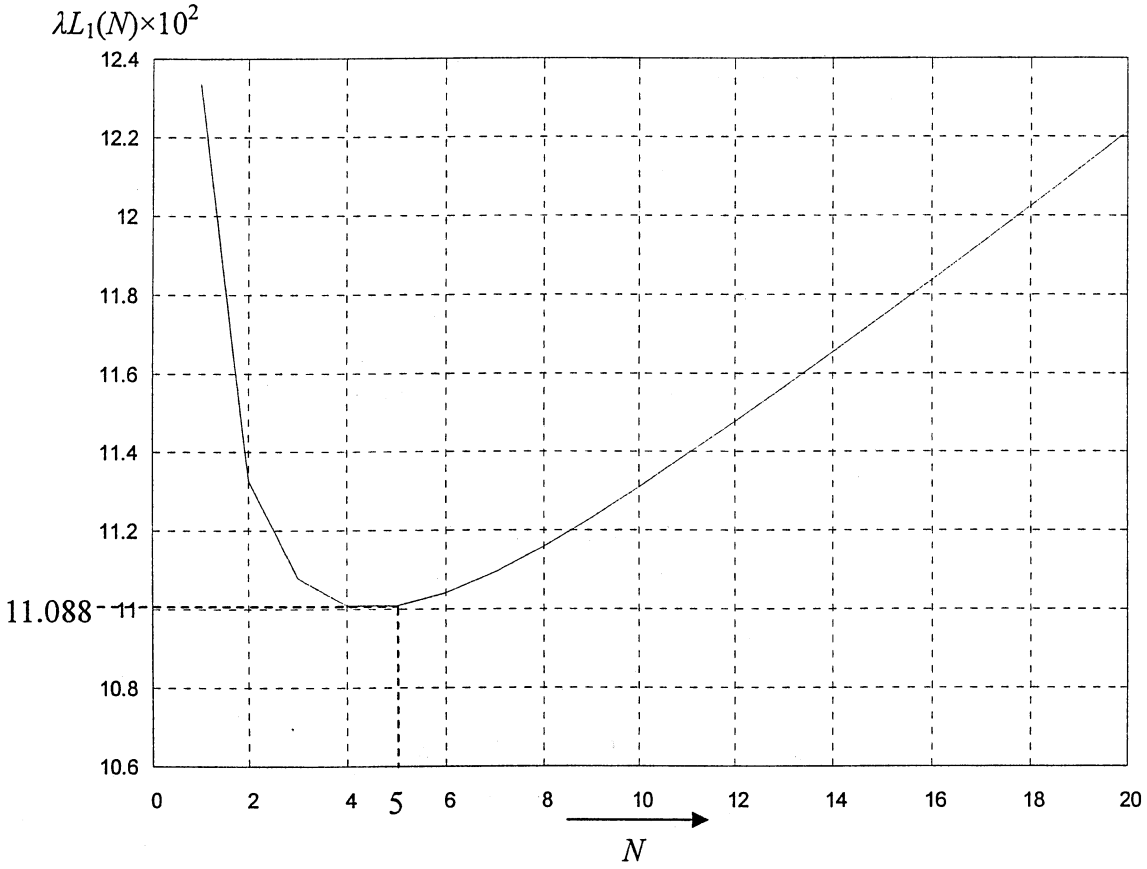


Figure 5.2: Mean time $\lambda L_1(N)$ when $1 \leq N \leq 20$.

5.4 Modified Model

It has been assumed until now that error rates increase with the number of checkpoints. We assume for the simplicity of the model that a modular system has no error during the interval $(T_{k-1}, T_k]$ is $\overline{F}(T_k) \ / \ \overline{F}(T_{k-1})$, irrespective of rollback operation. Then, the mean

execution time of the process for the interval $(T_{k-1},T_k]$ is

$$L_2(k)=(T_k-T_{k-1}+C)\frac{\overline{F}(T_k)}{\overline{F}(T_{k-1})}+[T_k-T_{k-1}+C+L_2(k)]\frac{F(T_k)-F(T_{k-1})}{\overline{F}(T_{k-1})}, \tag{5.15}$$

and solving it,

$$L_2(k)=\frac{(T_k-T_{k-1}+C)\overline{F}(T_{k-1})}{\overline{F}(T_k)} \qquad (k=1,2,\cdots,N). \tag{5.16}$$

Thus, the mean time to completion of the process is

$$L_2(N)=\sum_{k=1}^{N}\frac{(T_k-T_{k-1}+C)\overline{F}(T_{k-1})}{\overline{F}(T_k)} \qquad (N=1, 2, \cdots). \tag{5.17}$$

We find optimal times T_k which minimize $L_2(N)$ for a specified N. Let $f(t)$ be a density function of $F(t)$ and $r(t)\equiv f(t)/\overline{F}(t)$ be the failure rate of $F(t)$. Then, differentiating $L_2(N)$ with respect to T_k and setting it equal to zero,

$$\frac{\overline{F}(T_{k-1})}{\overline{F}(T_k)}[1+r(T_k)(T_k-T_{k-1}+C)]=\frac{\overline{F}(T_k)}{\overline{F}(T_{k+1})}[1+r(T_k)(T_{k+1}-T_k+C)]$$

$$(k=1, 2, \cdots, N-1). \tag{5.18}$$

Therefore, we have the following result:

(i) When $N=1$ and $T_1=S$, the mean time is

$$L_2(1)=\frac{S+C}{\overline{F}(S)}. \tag{5.19}$$

(ii) When $N=2$, from (5.18),

$$\frac{1}{\overline{F}(T_1)}[1+r(T_1)(T_1+C)]-\frac{\overline{F}(T_1)}{\overline{F}(S)}[1+r(T_1)(S-T_1+C)]=0. \tag{5.20}$$

Letting $Q_2(T_1)$ be the left-hand side of (5.20), it is evidently seen that

$$Q_2(0)=1+r(0)C-\frac{1}{\overline{F}(S)}[1+r(0)(S+C)]<0,$$

$$Q_2(S)=\frac{1}{\overline{F}(S)}[1+r(S)(S+C)]-[1+r(S)C]>0.$$

Thus, there exists some T_1^* that satisfies (5.20).

(iii) When N=3, we compute $T_k(k=1,2)$ which satisfies the simultaneous equations:

$$\frac{1}{\overline{F}(T_1)}[1+r(T_1)(T_1+C)]=\frac{\overline{F}(T_1)}{\overline{F}(T_2)}[1+r(T_1)(T_2-T_1+C)], \tag{5.21}$$

$$\frac{\overline{F}(T_1)}{\overline{F}(T_2)}[1+r(T_1)(T_2-T_1+C)]=\frac{\overline{F}(T_2)}{\overline{F}(S)}[1+r(T_2)(S-T_2+C)]. \tag{5.22}$$

(iv) When N=4,5, ···, we compute T_k similarly.

We compute sequential checkpoint intervals $T_k(k=1, 2, \cdots, N)$ for a double modular system when error rates increase with the number of checkpoints. It is assumed that $\overline{F}(t)=e^{-2(\lambda t)^m}$ (m>1), $\lambda C=10^{-3}$ and $\lambda S=10^{-1}$.

Table 5.3: Checkpoint intervals λT_k and mean time $\lambda L_2(N)$ when $\bar{F}(t)=\exp[-2(\lambda t)^{1.1}]$, $\lambda C=10^{-3}$ and $\lambda S=10^{-1}$.

N	1	2	3	4	5	6	7	8	9
$\lambda T_1\times10^2$	10.00	5.17	3.51	2.67	2.16	1.81	1.57	1.38	1.23
$\lambda T_2\times10^2$		10.00	6.80	5.17	4.18	3.51	3.03	2.67	2.39
$\lambda T_3\times10^2$			10.00	7.60	6.15	5.17	4.46	3.93	3.51
$\lambda T_4\times10^2$				10.00	8.09	6.80	5.87	5.17	4.62
$\lambda T_5\times10^2$					10.00	8.41	7.26	6.39	5.71
$\lambda T_6\times10^2$						10.00	8.63	7.60	6.80
$\lambda T_7\times10^2$							10.00	8.81	7.87
$\lambda T_8\times10^2$								10.00	8.94
$\lambda T_9\times10^2$									10.00
$\lambda L_2(N)\times10^2$	11.8390	11.0424	10.8593	10.8207	10.8384	10.8839	10.9452	11.0162	11.0938

Table 5.3 presents sequential intervals λT_k and the resulting mean times $\lambda L_2(N)$ for N=1, 2, ⋯, 9 when $\bar{F}(t)=\exp[-2(\lambda t)^{1.1}]$, $\lambda S=10^{-1}$ and $\lambda C=10^{-3}$. In this case, the mean time is the smallest when N=4, *i.e.*, the optimal checkpoint number is N^*=4 and the checkpoint times T_k^* (k=1, 2, 3, 4) should be placed at 2.67, 5.17, 7.60, 10.00(sec) for $\lambda=10^{-2}$(1/sec), and the mean time 10.8207 is about 8% longer than an original execution time S=10.

Next, we consider the approximate method similar to that of the previous model. It is assumed that the probability that a modular system has no error during $(T_{k-1}, T_k]$ is constant, *i.e.*, $\overline{F}(T_k)/\overline{F}(T_{k-1})=q$ $(k=1,2,\cdots,N)$. When $\overline{F}(t)=\mathrm{e}^{-2(\lambda t)^m}$,

$$\frac{\overline{F}(T_k)}{\overline{F}(T_{k-1})}=\mathrm{e}^{-2\left[(\lambda T_k)^m-(\lambda T_{k-1})^m\right]}=q\equiv\mathrm{e}^{-\tilde{q}},$$

and hence,

$$2(\lambda T_k)^m-2(\lambda T_{k-1})^m=\tilde{q}\quad(k=1,2,\cdots,N).$$

Thus,

$$(\lambda T_k)^m=\frac{k\tilde{q}}{2},$$

i.e.,

$$\lambda T_k=\left(\frac{k\tilde{q}}{2}\right)^{1/m}\quad(k=1,2,\cdots,N-1),$$

and

$$\lambda T_N=\lambda S=\left(\frac{N\tilde{q}}{2}\right)^{1/m}.$$

Therefore,

$$L_2(N)=\mathrm{e}^{\tilde{q}}(S+NC)=\mathrm{e}^{2(\lambda S)^m/N}(S+NC). \tag{5.23}$$

Forming the inequality $L_2(N+1)-L_2(N)\geq 0$,

$$C\geq(S+NC)\left\{\mathrm{e}^{2(\lambda S)^m/[N(N+1)]}-1\right\}. \tag{5.24}$$

It is proved that the right-hand side of (5.24) is strictly decreasing to 0. Thus, an optimal $\tilde{N}$ to minimize $L_2(N)$ in (5.23) is given by a unique minimum which satisfies (5.24).

Table 5.4: Mean time $\lambda L_2(N)$ for $\tilde{q}$ when $\lambda S=10^{-1}$ and $\lambda C=10^{-3}$.

N	$\tilde{q}$	$\lambda L_2(N)\times 10^2$
1	0.1588656	11.83902
2	0.0794328	11.04326
3	0.0529552	10.86014
4	0.0397164	10.82136
5	0.0317731	10.83897
6	0.0264776	10.88441
7	0.0226951	10.94561
8	0.0198582	11.01661
9	0.0176517	11.09411

Table 5.4 presents $\tilde{q}=2(\lambda S)^m/N$ and $\lambda L_2(N)$ in (5.23) for $N=1, 2, \cdots, 9$ under the same assumptions in Table 5.3. In this case, $\tilde{N}=4=N^*$ and approximate optimal checkpoint times are $\lambda \tilde{T}_k\times 10^2$ = 2.84, 5.33, 7.70, 10.00, that are a little longer than those of Table 5.3.

5.5 Conclusions

We have considered two checkpoint models with a finite execution time S where error rates increase with the number of checkpoints and with the original execution time. The mean times to completion of the process for two models have been obtained and the computing procedures for determining optimal checking intervals to minimize them have been shown. When error rates have an exponential and Weibull distributions, sequential checkpoint intervals have been computed numerically by solving simultaneous equations. Furthermore, approximate checkpoint intervals have been derived by assuming that the probability of the occurrence of errors during each checkpoint interval is constant. This is very simple and gives good approximations to optimal intervals. It would be sufficient to use practically approximate checkpoint intervals for actual models.

Chapter 6

Random Checkpoint Models for a Double Modular System

Tasks with random processing times are executed successively. A double modular system of error detection for the processing of each task is adopted. Two types of checkpoints such as compare-checkpoint and compare-and-store checkpoint can be placed at the end of tasks. The problem is that in what places we set suitable checkpoints. The mean execution times per one task for three schemes are obtained, and optimal numbers which minimize them are derived analytically. Extended models with majority decision modules and a spare module are proposed.

6.1 Introduction

Most computer systems in offices and industries execute successively tasks each of which has a random processing time. In such systems, some errors often occur due to noises, human errors and hardware faults. To detect and mask errors, some useful fault tolerant computing techniques have been adopted [12, 35]. The simplest scheme in recovery techniques of error detection is as follows [20]: We execute two independent modules which compare two states at checkpoint times. If two states of each module do not match with each other, we go back to the newest checkpoint and make their retrials.

Several studies of deciding optimal checkpoint frequencies have been made: The performance and reliability of a double modular system with one spare module were evaluated [22, 28]. Furthermore, the performance of checkpoint schemes with task duplication was evaluated [44, 45]. The optimal instruction-retry period that minimizes the probability of the dynamic failure by a triple modular controller was derived [8]. Evaluation models with finite checkpoints and bounded rollback were discussed [26].

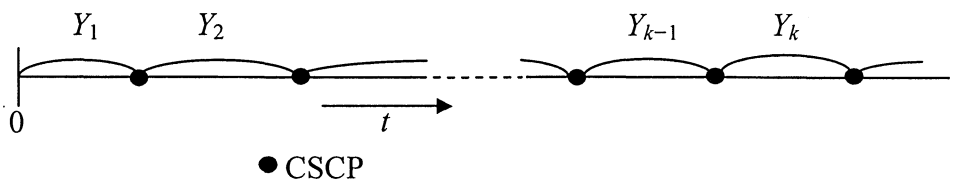


Figure 6.1: Task execution for Scheme 1.

Suppose that we have to execute the successive tasks with a processing time Y_k (k=1, 2, ···) (Figure 6.1). A double modular system of error detection for the processing of each task is adopted. Then, introducing two types of checkpoints; compare-and-store checkpoint (CSCP) and compare- checkpoint (CCP) [20], we consider the following three checkpoint schemes:

(1) CSCP is placed at each end of tasks.

(2) CSCP is placed at the N-th end of tasks.

(3) CCP is placed at each end of tasks and CSCP is placed at the N-th end of tasks.

The mean execution times per one task for each scheme are obtained, and optimal numbers N^* that minimize them for Schemes 2 and 3 are derived analytically and are compared numerically. This is one of applied models with random maintenance times [17, 37] to checkpoint models. Such schemes would be useful when it is better to place checkpoints at the end of tasks than those on one's way. Further, we extend a double modular system to a majority decision system and the system with one spare module [22].

6.2 Double Modular System

Suppose that task k has a processing time Y_k (k=1, 2, ···) with an identical distribution $G(t) \equiv \Pr\{Y_k \leq t\}$ and a finite mean $\mu = \int_0^\infty [1 - G(t)]\,dt < \infty$, and is executed successively. To detect errors, we provide two independent modules where they compare two states at checkpoint times. Further, it is assumed that some errors occur at a constant rate λ ($\lambda > 0$), *i.e.*, the probability that two modules have no error during $(0, t]$ is $e^{-2\lambda t}$.

(1) Scheme 1

CSCP is placed at each end of task k: When two states of modules match with each other at the end of task k, the process of task k is correct and its state is stored (Figure 6.1). In this case, two modules go forward and execute task k+1. However, when two states do not match, it is judged that some errors have occurred. Then, two modules go back and make the retry of task k again.

Let C be the overhead for the comparison of two states and Cs be the overhead for their store. Then, the mean execution time of the process of task k is given by a renewal equation:

$$\tilde{L}_1(1) = \int_0^\infty \left[e^{-2\lambda t}(C + Cs + t) + (1 - e^{-2\lambda t})(C + t + \tilde{L}_1(1))\right] dG(t). \tag{6.1}$$

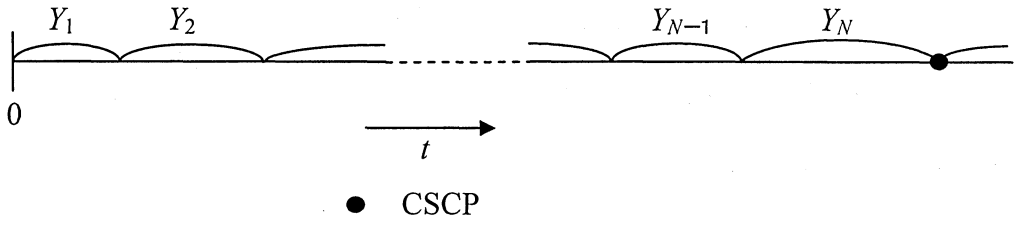


Figure 6.2: Task execution for Scheme 2.

Solving (6.1) for $\tilde{L}_1(1)$,

$$\tilde{L}_1(1) = \frac{C + \mu + CsG^*(2\lambda)}{G^*(2\lambda)},$$

where $G^*(s)$ is the Laplace-Stieltjes (LS) transform of $G(t)$, *i.e.*, $G^*(s) = \int_0^\infty e^{-st} dG(t)$ for $s>0$. Therefore, the mean execution time per one task is

$$L_1(1) \equiv \tilde{L}_1(1) = \frac{C + \mu}{G^*(2\lambda)} + Cs. \tag{6.2}$$

(2) Scheme 2

CSCP is placed only at the end of task N (Figure 6.2): When two states of all task k (k=1, 2, ···, N) match at the end of task N, its state is stored and two modules execute task N+1. When two states do not match, two modules go back in the first task 1 and make their retries.

By the method similar to obtaining (6.1), the mean execution time of the process of all task k (k=1, 2, ···, N) is

$$\tilde{L}_2(N)=\int_0^\infty\left[e^{-2\lambda t}\ (C+Cs+t)+\left(1-e^{-2\lambda t}\right)\left(C+t+\tilde{L}_2(N)\right)\right]dG^{(N)}(t), \tag{6.3}$$

where $G^{(N)}(t)$ is the N-fold Stieltjes convolution of $G(t)$ with itself, *i.e.*, $G^{(N)}(t)\equiv\int_0^\infty G^{(N-1)}(t-u)dG(u)$ (N=1, 2, ···), and $G^{(0)}(t)\equiv 1$ for $t\geq 0$ and $G^{(1)}(t)=G(t)$. Solving (6.3) for $\tilde{L}_2(N)$,

$$\tilde{L}_2(N)=\frac{C+N\mu+Cs\left[G^*(2\lambda)\right]^N}{\left[G^*(2\lambda)\right]^N}.$$

Therefore, the mean execution time per one task is

$$L_2(N)\equiv\frac{\tilde{L}_2(N)}{N}=\frac{C+N\mu}{N\left[G^*(2\lambda)\right]^N}+\frac{Cs}{N}\qquad(N=1, 2, \cdots). \tag{6.4}$$

When N=1, $L_2(1)$ agrees with (6.2).

We find an optimal number N_2^* that minimizes $L_2(N)$. There exists a finite N_2^* ($1\leq N_2^*<\infty$) because $\lim_{N\to\infty}L_2(N)=\infty$. From the inequality $L_2(N+1)-L_2(N)\geq 0$,

$$N\left[(N+1)\mu+C\right]\frac{1-G^*(2\lambda)}{G^*(2\lambda)}-Cs\left[G^*(2\lambda)\right]^N\geq C\qquad(N=1, 2, \cdots). \tag{6.5}$$

The left-hand side of (6.5) is strictly increasing to ∞ in N. Thus, there exists a finite and unique minimum N_2^* ($1\leq N_2^*<\infty$) which satisfies (6.5). If $(2\mu+C)[1-G^*(2\lambda)]\geq[C+Cs\times G^*(2\lambda)]\,G^*(2\lambda)$, then $N_2^*=1$.

When $G(t)=1-e^{-t/\mu}$, Equation (6.5) is rewritten as

$$N\left(N+1+\frac{C}{\mu}\right)2\lambda\mu-\frac{Cs}{\mu}\left(\frac{1}{2\lambda\mu+1}\right)^N\geq\frac{C}{\mu}\qquad(N=1, 2, \cdots). \tag{6.6}$$

Table 6.1 presents the optimal number N_2^* and the resulting execution time $L_2(N_2^*)/\mu$ and $L_2(1)/\mu$ in (6.2) for $\lambda\mu$ and C/μ when Cs/μ=0.1. This indicates that optimal N_2^*

Table 6.1: Optimal number N_2^* and the resulting execution time $L_2(N_2^*)/\mu$ for Scheme 2 when $G(t)=1-e^{-t/\mu}$ and $Cs/\mu=0.1$.

$\lambda\mu$	$C/\mu=0.5$			$C/\mu=0.1$		
	N_2^*	$L_2(N_2^*)/\mu$	$L_2(1)/\mu$	N_2^*	$L_2(N_2^*)/\mu$	$L_2(1)/\mu$
0.1	2	1.850	1.900	1	1.420	1.420
0.05	2	1.563	1.750	1	1.310	1.310
0.01	5	1.234	1.630	3	1.130	1.222
0.005	7	1.163	1.615	4	1.092	1.211
0.001	17	1.071	1.603	10	1.040	1.202
0.0005	24	1.050	1.602	14	1.028	1.201
0.0001	54	1.022	1.600	32	1.013	1.200

decrease with $\lambda\mu$ and increase with C/μ. For example, when $\lambda\mu=0.005$ and $C/\mu=0.1$, $N_2^*=4$ and $L_2(N_2^*)/\mu$ is 1.092 that is about 10% shorter than $L_2(1)/\mu=1.211$ for Scheme 1.

(3) Scheme 3

CSCP is placed at the end of task N and CCP is placed only at the end of task k (k=1, 2, ···, $N-1$) between CSCPs (Figure 6.3): When two states of task k (k=1, 2, ···, $N-1$) match at the end of task k, two modules execute task $k+1$. When two states of task k (k=1, 2, ···, N) do not match, two modules go back in the first task 1. When two states of task N match,

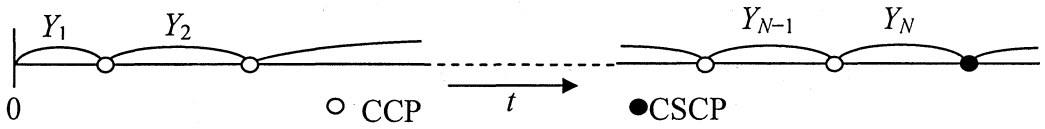


Figure 6.3: Task execution for Scheme 3.

the process of all tasks N is completed, and its state is stored. Two modules execute task N+1.

Let $\tilde{L}_3(k)$ be the mean execution time from task k to the completion of task N. Then, by the method similar to obtaining (6.3),

$$\tilde{L}_3(k) = \int_0^\infty \left\{ e^{-2\lambda t} \left[C + t + \tilde{L}_3(k+1) \right] + \left(1 - e^{-2\lambda t} \right) \left[C + t + \tilde{L}_3(1) \right] \right\} dG(t) \quad (k=1, 2, \cdots, N-1), \tag{6.7}$$

$$\tilde{L}_3(N) = \int_0^\infty \left\{ e^{-2\lambda t} (C + t + Cs) + \left(1 - e^{-2\lambda t} \right) \left[C + t + \tilde{L}_3(1) \right] \right\} dG(t). \tag{6.8}$$

Solving (6.7) and (6.8) for $\tilde{L}_3(1)$,

$$\tilde{L}_3(1) = \frac{(C+\mu)\left\{ 1 - \left[G^*(2\lambda) \right]^N \right\}}{\left[1 - G^*(2\lambda) \right] \left[G^*(2\lambda) \right]^N} + Cs.$$

Therefore, the mean execution time per one task is

$$L_3(N) \equiv \frac{\tilde{L}_3(1)}{N} = \frac{(C+\mu)\left\{ 1 - \left[G^*(2\lambda) \right]^N \right\}}{N \left[1 - G^*(2\lambda) \right] \left[G^*(2\lambda) \right]^N} + \frac{Cs}{N} \quad (N=1, 2, \cdots). \tag{6.9}$$

When N=1, $L_3(1)$ agrees with (6.2). If

$$\frac{\mu}{C+\mu} > \frac{1}{N-1} \sum_{k=1}^{N-1} \left[G^*(2\lambda) \right]^k \quad (N=2, 3, \cdots), \tag{6.10}$$

Table 6.2: Optimal number N_3^* and the resulting execution time $L_3(N_3^*)/\mu$ for Scheme 3 when $G(t)=1-e^{-t/\mu}$ and $Cs/\mu=0.1$.

$\lambda\mu$	$C/\mu=0.5$		$C/\mu=0.1$	
	N_3^*	$L_3(N_3^*)/\mu$	N_3^*	$L_3(N_3^*)/\mu$
0.1	1	1.900	1	1.420
0.05	1	1.750	1	1.310
0.01	3	1.594	3	1.178
0.005	4	1.563	4	1.153
0.001	8	1.526	9	1.122
0.0005	12	1.518	13	1.115
0.0001	26	1.508	30	1.107

then Scheme 3 is better than Scheme 2 for the same number N (N=2, 3,···).

It can be clearly seen that a finite $N_3^*(1 \le N_3^* < \infty)$ that minimizes $L_3(N)$ exists. From the inequality $L_3(N+1)-L_3(N) \ge 0$,

$$\frac{1}{\left[G^*(2\lambda)\right]^{N+1}} \sum_{j=1}^{N} \left\{1-\left[G^*(2\lambda)\right]^j\right\} \ge \frac{Cs}{C+\mu} \quad (N=1, 2, \cdots). \tag{6.11}$$

The left-hand side of (6.11) is strictly increasing to ∞ in N. Thus, there exists a finite and unique minimum $N_3^*(1 \le N_3^* < \infty)$ which satisfies (6.11). If $(C+\mu)[1-G^*(2\lambda)] \ge Cs[G^*(2\lambda)]^2$, then $N_3^*=1$.

When $G(t)=1-e^{-t/\mu}$, Equation (6.11) is rewritten as

$$(2\lambda\mu+1)^{N+1}\sum_{j=1}^{N}\left[1-\left(\frac{1}{2\lambda\mu+1}\right)^{j}\right]\geq\frac{Cs/\mu}{C/\mu+1}\quad(N=1,2,\cdots). \tag{6.12}$$

Table 6.2 presents the optimal number N_3^* and the resulting execution time $L_3(N_3^*)/\mu$ in (6.9) for $\lambda\mu$ and C/μ when Cs/μ=0.1. In this case, Scheme 3 is not better than Scheme 2 when C/μ=0.5. However, if C/μ would be smaller, Scheme 3 would be better than Scheme 2 as shown in (6.10) when $\lambda\mu$=0.01 and 0.05. Further, if $\lambda\mu$ would be larger, we would not need to consider Scheme 3.

6.3 Extended Models

We consider the following two extended checkpoint modules:

(4) Majority System

We take up a majority decision system with $(2n+1)$ modules as an error masking system, *i.e.*, $(n+1)$-out-of-$(2n+1)$ system (n=1, 2, ···). If more than $(n+1)$ states of $(2n+1)$ modules match, the process of task k is correct and its state is stored. In this case, the probability that the process is correct during $(0, t\,]$ is

$$\overline{F}_{n+1}(t)=\sum_{j=n+1}^{2n+1}\binom{2n+1}{j}\left(e^{-\lambda t}\right)^{j}\left(1-e^{-\lambda t}\right)^{2n+1-j}$$

$$=\sum_{j=n+1}^{2n+1}\binom{2n+1}{j}\sum_{i=0}^{2n+1-j}\binom{2n+1-j}{i}(-1)^{i}\left(e^{-\lambda t}\right)^{j+i}\quad(n=1, 2, \cdots). \tag{6.13}$$

Thus, the mean execution time of the process of task k is

$$\tilde{L}_4(1) = \int_0^\infty [\bar{F}_{n+1}(t)(C+Cs+t)]dG(t) + F_{n+1}(t)[C+t+\tilde{L}_4(1)]dG(t), \tag{6.14}$$

where $F_{n+1}(t) \equiv 1-\bar{F}_{n+1}(t)$. Therefore, by the method similar to obtaining (6.2), the mean time execution time per one task is

$$L_4(1) \equiv \tilde{L}_4(1) = \frac{C+\mu}{\sum_{j=n+1}^{2n+1}\binom{2n+1}{j}\sum_{i=0}^{2n+1-j}\binom{2n+1-j}{i}(-1)^i G^*[(j+i)\lambda]} + Cs \tag{6.15}$$

(5) Spare Model

In Scheme 1, when two states of task k do not match, one spare module executes task k, and two modules go forward and execute task k+1 [22]. It is assumed that a spare module has no error. Furthermore, Cp is the total overhead of preparing a spare module and of setting a correct process at checkpoint times.

Let $\tilde{L}_5(1)$ be the mean execution time from task k. Then, by the similar method in **(1)**,

$$\begin{aligned}\tilde{L}_5(1) &= \int_0^\infty [e^{-2\lambda t}(C+t+Cs) + (1-e^{-2\lambda t})(C+t+Cp+t+Cs)]dG(t) \\ &= C+\mu+Cs+(Cp+\mu)[1-G^*(2\lambda)].\end{aligned} \tag{6.16}$$

If $(Cp+\mu)G^*(2\lambda) \le C+\mu$, then it is useful to provide a spare module.

6.4 Conclusions

We have considered two types of checkpoints for a double modular system and adopted three schemes for answering the problems in what places we set suitable checkpoints. It has been shown in the numerical examples that when an error rate λ and a mean processing time μ increase and the overhead C for the comparison decreases, Schemes 2 and 3 are better than Scheme 1. Further, we have proposed two extended models with a majority decision system and the system with a spare module. It would require to discuss further which schemes including extended models are better in practical situations.

Chapter 7

Random Checkpoint Models for Multiple Modular Systems with Increasing Error Rates

Tasks with random processing times are executed successively. A double modular system of error detection for the processing of each task is adopted. Two types of checkpoints can be placed at the end of tasks. The problem is that in what places we set suitable checkpoints. It is assumed that error rates of two models for task k increase with the number of checkpoints. The mean execution times per one task for three schemes are obtained, and optimal numbers which minimize them are derived analytically. A majority decision system is also proposed. What system is optimal is discussed numerically.

7.1 Introduction

Most computer systems in offices and industries execute successively tasks each of which has a random processing time. In such systems, some errors often occur due to noises, human errors and hardware faults. To detect and mask errors, some useful fault tolerant computing techniques have been adopted [12, 35]. The simplest scheme in recovery techniques of error detection is as follows [20]: We execute two independent modules which compare two states at checkpoint times. If two states of each module do not match with each other, we go back to the newest checkpoint and make their retrials.

Several studies of deciding optimal checkpoint frequencies have been made: The performance and reliability of a double modular system with one spare module were evaluated [28, 22]. Further, the performance of checkpoint schemes with task duplication was evaluated [44, 45]. The optimal instruction-retry period that minimizes the probability of the dynamic failure by a triple modular controller was derived [8]. Evaluation models with finite checkpoints and bounded rollback were discussed [26].

Suppose that we have to execute the successive tasks with a processing time Y_k (k=1, 2, ···) (Figure 6.1). A double modular system of error detection for the processing of each task is adopted. Then, introducing two types of checkpoints; compare-and-store checkpoint (CSCP) and compare- checkpoint (CCP) [20], we consider the following three checkpoint schemes:

(1) CSCP is placed at each end of tasks.

(2) CSCP is placed at the N-th end of tasks.

(3) CCP is placed at each end of tasks and CSCP is placed at the N-th end of tasks.

It is assumed that the error rate of every task k for Scheme 1 is the same one, however, error rates of task k (k=1, 2, ···, N) for Scheme (2) and (3) increase with the number k of checkpoints. The mean execution times per one task for each scheme are obtained, and optimal numbers N^* that minimize them for Schemes 2 and 3 are derived analytically and are compared numerically. This is one of applied models with random maintenance times [17, 37] to checkpoint models. Such schemes would be useful when it is better to place checkpoints at the end of tasks than those on one's way. Further, we extend a double modular system to a majority decision system, and obtain numerically what a majority decision system is optimal.

7.2 Double Modular System

Suppose that task k has a processing time Y_k (k=1, 2, ···) with an identical distribution $G(t) \equiv \Pr\{Y_k \leq t\}$ and a finite mean time $\mu = \int_0^\infty [1 - G(t)]\mathrm{d}t < \infty$, and is executed successively. To detect errors, we provide two independent modules where they compare two states at checkpoint times. Furthermore, the probability that a modular system for task k has no error during (0, t] is assumed to be $e^{-\lambda_1 t}$ for Scheme 1 and $e^{-\lambda_k t}$ for Schemes

2 and 3, irrespective of other tasks and rollback operation. Then, we consider the following three schemes:

(1) Scheme 1

CSCP is placed at each end of task k: When two states of modules match with each other at the end of task k, the process of task k is correct and its state is stored. In this case, two modules go forward and execute task k+1. However, when two states do not match, it is judged that some errors have occurred. Then, two modules go back and re-execute task k again.

Let C be the overhead for the comparison of two states and Cs be the overhead for their store. Noting that every task k has the same error rate λ_1, the mean time execution time of the process of every task k for two modules is given by a renewal equation

$$L_1 = \int_0^\infty \left[e^{-2\lambda_1 t} (C + t + Cs) + (1 - e^{-2\lambda_1 t})(C + t + L_1) \right] dG(t). \tag{7.1}$$

Solving (7.1) for L_1,

$$L_1 = \frac{C + \mu}{G^*(2\lambda_1)} + Cs, \tag{7.2}$$

where $G^*(s)$ is the Laplace-Stieltjes (LS) transform of $G(t)$, *i.e.*, $G^*(s) \equiv \int_0^\infty e^{-st} dG(t)$ for $s>0$.

(2) Scheme 2

CSCP is placed only at the end of task N (Figure 6.2): When two states of all task k(k=1,

2, ···, N) match at the end of task N, its state is stored and two modules execute task N+1. When two states do not match, two modules go back in the first task and make their re-executions.

Let $L_1(k)$ be the mean execution time from task k to the completion of task N. Because the probability that no error of two modules for task k occurs is

$$\int_0^\infty e^{-2\lambda_k t} dG(t) = G^*(2\lambda_k) \qquad (k=1,2,\cdots,N). \tag{7.3}$$

Thus, we have a renewal equation

$$\tilde{L}_2(N) = (NC + N\mu + Cs)\prod_{k=1}^{N} G^*(2\lambda_k) + \left[NC + N\mu + \tilde{L}_2(N)\right]\left[1 - \prod_{k=1}^{N} G^*(2\lambda_k)\right]. \tag{7.4}$$

Solving (7.4) for $\tilde{L}_2(N)$,

$$\tilde{L}_2(N) = \frac{NC + N\mu}{\prod_{k=1}^{N} G^*(2\lambda_k)} + Cs. \tag{7.5}$$

Therefore, the mean execution time per one task is

$$L_2(N) \equiv \frac{\tilde{L}_2(N)}{N} = \frac{C + \mu}{\prod_{k=1}^{N} G^*(2\lambda_k)} + \frac{Cs}{N} \qquad (N=1, 2, \cdots). \tag{7.6}$$

When N=1, $L_2(1)$ agrees with (7.2).

We find an optimal number N_2^* that minimizes $L_2(N)$. There exists a finite N_2^* ($1 \le N_2^* < \infty$) because $\lim_{N\to\infty} L_2(N) = \infty$. From the inequality $L_2(N+1) - L_2(N) \ge 0$,

$$\frac{N(N+1)\left[1 - G^*(2\lambda_{N+1})\right]}{\prod_{k=1}^{N+1} G^*(2\lambda_k)} \ge \frac{Cs}{C + \mu}. \tag{7.7}$$

From the assumption that $\lambda_k \le \lambda_{k+1}$, $G^*(2\lambda_{k+1}) \le G^*(2\lambda_k)$, *i.e.*, $1 - G^*(2\lambda_k) \le 1 - G^*(2\lambda_{k+1})$. Thus, it is clearly noted that the left-hand side of (7.7) is strictly increasing to ∞ in N.

Therefore, there exists a finite and unique minimum N_2^* ($1 \leq N_2^* < \infty$) that satisfies (7.7). If

$$\frac{1-G^*(2\lambda_2)}{G^*(2\lambda_1)G^*(2\lambda_2)} \geq \frac{Cs}{2(C+\mu)},$$

then $N^*=1$.

When $G(t)=1-e^{-t/\mu}$. Equation (7.7) is rewritten as

$$\frac{N(N+1)\left(\frac{2\lambda_{N+1}\mu}{2\lambda_{N+1}\mu+1}\right)}{\prod_{k=1}^{N+1}\frac{1}{2\lambda_k\mu+1}} \geq \frac{Cs}{C+\mu}. \tag{7.8}$$

It is assumed that $\lambda_k=[1+\alpha(k-1)]\lambda$, *i.e.*, an error rate increases by 100α% of an original rate λ. Then, we compute an optimal number N_2^* which satisfies (7.8). Table 7.1 presents optimum N_2^* for $\lambda\mu$ and C/μ when α=0.1 and Cs/μ = 0.1.

(3) Scheme 3

CSCP is placed at the end of task N and CCP is placed only at the end of task k (k=1, 2, ···, N−1) between CSCP (Figure 6.3). When two states of task k match at the end of task k, two modules execute task k+1. When two states of task k (k=1, 2, ···,N) do not match, two modules go back in the first task 1. When two states of task N match, the processes of all N tasks are completed, and its state is stored. Two modules execute task N+1.

Let $\tilde{L}_3(k)$ be the mean execution time from task k to the completion of task N. Then, by the method similar to obtaining (7.4),

Table 7.1. Optimal number N_2^* and the resulting execution time $L_2(N_2^*)/\mu$ for Scheme 2 when $G(t)=1-e^{-t/\mu}$ and $Cs/\mu=0.1$

$\lambda\mu$	$C/\mu=0.5$			$C/\mu=0.1$		
	N_2^*	$\frac{L_2(N_2^*)}{\mu}$	$\frac{L_2(1)}{\mu}$	N_2^*	$\frac{L_2(N_2^*)}{\mu}$	$\frac{L_2(1)}{\mu}$
0.1	1	1.900	1.900	1	1.420	1.420
0.05	1	1.750	1.750	1	1.310	1.310
0.01	2	1.614	1.630	3	1.208	1.222
0.005	4	1.595	1.615	6	1.202	1.211
0.001	14	1.578	1.603	17	1.175	1.202
0.0005	21	1.603	1.602	26	1.170	1.201
0.0001	53	1.569	1.600	63	1.160	1.200

$$\tilde{L}_3(k)=\int_0^\infty \left\{e^{-2\lambda_k t}\left[C+t+\tilde{L}_3(k+1)\right]+\left(1-e^{-2\lambda_k t}\right)\left[C+t+\tilde{L}_3(1)\right]\right\}dG(t) \qquad (k=1, 2, \cdots, N-1), \tag{7.9}$$

$$\tilde{L}_3(N)=\int_0^\infty \left\{e^{-2\lambda_N t}(C+t+Cs)+\left(1-e^{-2\lambda_N t}\right)\left[C+t+\tilde{L}_3(1)\right]\right\}dG(t). \tag{7.10}$$

Solving (7.9) and (7.10) for $\tilde{L}_3(1)$,

$$\tilde{L}_3(1)=\frac{(C+\mu)\sum_{j=0}^{N-1}\left\{\prod_{k=1}^{j}\left[G^*(2\lambda_k)\right]\right\}}{\prod_{k=1}^{N}G^*(2\lambda_k)}+Cs. \tag{7.11}$$

Therefore, the mean execution time per one task is

$$L_3(N) \equiv \frac{\tilde{L}_3(1)}{N} = \frac{(C+\mu)\sum_{j=0}^{N-1}\left[\prod_{k=1}^{j}\left[G^*(2\lambda_k)\right]\right]}{N\prod_{k=1}^{N}G^*(2\lambda_k)} + \frac{Cs}{N} \qquad (N=1, 2, \cdots), \tag{7.12}$$

where $\prod_{k=1}^{0} \equiv 1$. When N=1, $L_3(1)$ agrees with (7.2).

We find an optimal N_3^* that minimizes $L_3(N)$. From the inequality $L_3(N+1) - L_3(N) \geq 0$,

$$\frac{N\sum_{j=0}^{N}\left[\prod_{k=1}^{j}G^*(2\lambda_k)\right] - (N+1)G^*(2\lambda_{N+1})\sum_{j=0}^{N-1}\left[\prod_{k=1}^{j}\left[G^*(2\lambda_k)\right]\right]}{\prod_{k=1}^{N+1}G^*(2\lambda_k)} \geq \frac{Cs}{C+\mu},$$

i.e.,

$$N+1+\frac{[N-(N+1)]G^*(2\lambda_{N+1})\sum_{j=0}^{N}\left[\prod_{k=1}^{j}G^*(2\lambda_k)\right]}{\prod_{k=1}^{N+1}G^*(2\lambda_k)} \geq \frac{Cs}{C+\mu} \qquad (N=1,2, \cdots). \tag{7.13}$$

First, note that

$$N-(N+1)G^*(2\lambda_{N+1})$$

is increasing because

$$N-(N+1)G^*(2\lambda_{N+1})-(N-1)+NG^*(2\lambda_N)=1-G^*(2\lambda_{N+1})+N[G^*(2\lambda_N)-G^*(2\lambda_{N+1})]>0.$$

Further, denoting the left-hand side of (7.13) by $Q(N+1)$,

$$
\begin{aligned}
&Q(N+1)-Q(N)\\
&\quad=\frac{1}{\prod_{k=1}^{N+1}G^*(2\lambda_k)}\left\{\prod_{k=1}^{N+1}G^*(2\lambda_k)+\left[N-(N+1)G^*(2\lambda_{N+1})\right]\sum_{j=0}^{N}\left[\prod_{k=1}^{j}G^*(2\lambda_k)\right]\right.\\
&\quad\left.-\left[N-1-NG^*(2\lambda_N)\right]G^*(2\lambda_{N+1})\sum_{j=0}^{N-1}\left[\prod_{k=1}^{j}G^*(2\lambda_k)\right]\right\}\\
&\quad>\frac{1}{\prod_{k=1}^{N+1}G^*(2\lambda_k)}\left\{\prod_{k=1}^{N+1}G^*(2\lambda_k)+\left[N-1-NG^*(2\lambda_N)\right]G^*(2\lambda_{N+1})\prod_{k=1}^{N}G^*(2\lambda_k)\right\}\\
&\quad=N\left[1-G^*(2\lambda_N)\right]>0.
\end{aligned}
$$

Thus, the left-hand side of (7.13) is strictly increasing in N. Therefore, if a finite N^* to satisfy (7.13) exists, it is a finite and unique minimum such that (7.13). If

$$(C+\mu)\left[1+G^*(2\lambda_1)-2G^*(2\lambda_2)\right]\geq CsG^*(2\lambda_1)G^*(2\lambda_2),$$

then $N_3^*=1$.

By comparing (7.6) with (7.12), Scheme 3 is better than Scheme 2 for N such that

$$\sum_{j=0}^{N-1}\left[\prod_{k=1}^{j}G^*(2\lambda_k)\right]<N. \tag{7.14}$$

Because

$$\sum_{j=0}^{N-1}\left[\prod_{k=1}^{j}G^*(2\lambda_k)\right]-N=\sum_{j=0}^{N-1}\left[\prod_{k=1}^{j}G^*(2\lambda_k)-1\right]<0,$$

Scheme 3 is better than Scheme 2 for the same number N (N=2, 3, ···).

Table 7.2 presents the optimal number N_3^* and the resulting execution time $L_3(N_3^*)/\mu$ for $\lambda\mu$ and C/μ when α=0.1 and Cs/μ=0.1. This indicates that optimal N_3^* decrease with $\lambda\mu$

Table 7.2. Optimal number N_3^* and the resulting execution time $L_3(N_3^*)/\mu$ for Scheme 3 when $G(t)=1-e^{-t/\mu}$ and $Cs/\mu=0.1$

$\lambda\mu$	$C/\mu=0.5$		$C/\mu=0.1$	
	N_3^*	$\frac{L_3(N_3^*)}{\mu}$	N_3^*	$\frac{L_3(N_3^*)}{\mu}$
0.1	1	1.900	1	1.420
0.05	1	1.750	1	1.310
0.01	2	1.598	3	1.184
0.005	3	1.568	4	1.158
0.001	6	1.531	7	1.127
0.0005	8	1.522	9	1.120
0.0001	15	1.511	17	1.110

and C/μ. Compared with Table 7.1, N_3^* are equal to or less than N_2^*, however, $L_3(N_3^*)$ are less than $L_2(N_2^*)$ for $N_3^* \geq 2$, *i.e.*, Scheme 3 is better than Scheme 2.

7.3 Majority Decision System

We take up a majority decision system with $(2n+1)$ modules as an error masking system

that is called an (n+1)-out-of-(2n+ 1) system [17]. If more than (n+1) states of (2n+1) modules match, the process of task k is correct. In this case, the probability that the process of task k for Scheme 1 is correct during (0, t] is

$$\overline{F}_{n+1}(t) = \sum_{m=n+1}^{2n+1} \binom{2n+1}{m} \left(e^{-\lambda_1 t}\right)^m \left(1-e^{-\lambda_1 t}\right)^{2n+1-m}$$

$$= \sum_{m=n+1}^{2n+1} \binom{2n+1}{m} \sum_{i=0}^{2n+1-m} \binom{2n+1-m}{i} (-1)^i e^{-(m+1)\lambda_1 t} \quad (n=1, 2, \cdots). \qquad (7.15)$$

Thus, by the method similar to obtaining (7.1), the mean execution time of the process of task k is

$$L_4 = \int_0^\infty \left[\overline{F}_{n+1}(t)(C + C_s + t) + F_{n+1}(t)(C + t + L_4)\right] + dG(t). \qquad (7.16)$$

Solving (7.16) for L_4

$$L_4 = \frac{C+\mu}{\sum_{m=n+1}^{2n+1} \binom{2n+1}{m} \sum_{i=0}^{2n+1-m} \binom{2n+1-m}{i} (-1)^i G^*[(m+i)\lambda_1]} + Cs \quad (n=1, 2, \cdots). \qquad (7.17)$$

For example, when n=1, *i.e.*, the system is composed of a 2-out-of-3 system,

$$L_4 = \frac{C+\mu}{3G^*(2\lambda_1) - 2G^*(3\lambda_1)} + Cs. \qquad (7.18)$$

Similarly, the mean execution time per one task for Scheme 2 is, from (7.6)

$$L_5(N) = \frac{C+\mu}{\prod_{k=1}^{N} \left[\sum_{m=n+1}^{2n+1} \binom{2n+1}{m} \sum_{i=0}^{2n+1-m} \binom{2n+1-m}{i} (-1)^i G^*[(m+i)\lambda_1]\right]} + \frac{Cs}{N} \quad (N=1, 2, \cdots),$$

(7.19)

and the mean time for Scheme 3 is from (7.12),

$$L_6(N)=\frac{(C+\mu)\sum_{j=0}^{N-1}\left\{\prod_{k=1}^{j}\left[\sum_{m=n+1}^{2n+1}\binom{2n+1}{m}\sum_{i=0}^{2n+1-m}\binom{2n+1-m}{i}(-1)^i G^*[(m+i)\lambda_k]\right]\right\}}{N\prod_{k=1}^{N}\left[\sum_{m=n+1}^{2n+1}\binom{2n+1}{m}\sum_{i=0}^{2n+1-m}\binom{2n+1-m}{i}(-1)^i G^*[(m+i)\lambda_k]\right]}+\frac{Cs}{N}$$

(N=1, 2, ···). (7.20)

Table 7.3. Optimal number N_5^* and the resulting execution time $L_5(N_5^*)/\mu$ when $\lambda\mu$ = 0.01 and Cs/μ = 10.

n	C/μ=0.5		C/μ=0.1	
	N_5^*	$\frac{L_5(N_5^*)}{\mu}$	N_5^*	$\frac{L_5(N_5^*)}{\mu}$
1	1	15.000	2	10.203
2	1	22.000	2	13.005
3	1	33.000	1	16.200
4	1	48.000	1	19.200

Suppose that $C \equiv \binom{2n+1}{2} C_1$ in (7.19) because we have to compare two states of $(2n + 1)$ ones for $(2n + 1)$ modules. When C_1 is the overhead for comparing two states, Table 7.3 presents the optimal numbers N_5^* which minimize $L_5(N)$ in (7.19) and its resulting execution times $L_5(N_5^*)/\mu$ when $\lambda\mu$ = 0.01 and Cs/μ =10. This indicates that optimal N_3^* decrease with n and C/μ, and $L_5(N_5^*)$ increase with n and C/μ. Thus, from this table, an optimal decision system is a 2-out-of-3 system.

7.4 Conclusions

We have considered two types of checkpoints for a double modular system with increasing error rates, and adopted three schemes for answering the problem in what places we set suitable checkpoints. It has been shown in the numerical examples that when an original error rate λ and a mean processing time μ increase and the overhead C for the comparison decreases, Schemes 2 and 3 are better than Scheme 1. Furthermore, we have considered a redundant system of a majority decision as an error masking system, and obtained numerically what system is optimal for each scheme.

Chapter 8

Conclusions

This thesis have formulated the stochastic models of a recovery mechanism and analyzed them theoretically, using stochastic processes. We have adopted checkpoint and backup operation as recovery techniques and obtained expected costs, expected overheads and mean times to the completion of processes. Using reliability theory, we have discussed analytically optimal policies which minimize such objective functions. Further, to understand the results easily, we have given numerical examples at the end of each chapter, evaluated numerically some measures and determined the best scheme. If some parameters of each model are estimated from actual models, we could apply such models to practical recovery models by modifying them.

We have used some useful techniques to analyze models: One is how to formulate stochastic models, using the techniques of stochastic processes, and the other is how to derive optimal policies, using reliability theory. Within there techniques, it would be

instructive to derive optimal times for a finite time span and by solving simultaneous equations. These would be very useful to the analysis of other fields in reliability and computer systems.

Some valuable contributions to the study of recovery techniques for computer systems have been made as follows:

In Chapter 2, we have considered the modified inspection model where checking times of an operating unit are placed at sequential times T_k and the backup operation is carried out until the latest checking time when a failure was detected. The expected costs until the backup operation have been obtained, and optimal policies, which minimize them for two case of periodic and sequential times, have been analytically discussed. Further, modified models where the operating time of the unit is finite and a fault remains hidden have been proposed and analyzed.

In Chapter 3, we have considered two-level recovery schemes; soft checkpoint (SC) and hard checkpoint (HC) as recovery techniques. When HCs are placed on the beginning and at the end of the process, and SCs are placed between HCs, the total expected overhead of the process has been obtained, using Markov renewal processes. Optimal intervals of SCs to minimize the expected cost have been derived and computed numerically. It has been shown that a two-level recovery scheme can achieve a good performance.

In Chapter 4, we have considered multiple modular redundant systems as the recovery techniques of error detection and error masking on a finite process execution when checkpoints are placed at periodic times kT. Introduceing the overheads of

comparison and decision by majority, we have obtained the mean times to the completion of the process and derived analytically optimal checkpoint intervals which minimize them. Further, it has been shown numerically that what a majority decision system is optimal.

In Chapter 5, when checkpoints are placed at sequential times T_k, we have two models where error rates increase with the number of checkpoints and with the original execution time. We have obtained the mean times to the completion of the process and computed numerically optimal checkpoint intervals which minimize them by solving simultaneous equations. Further, approximate checkpoint intervals have been derived by denoting that the probability of error occurrence during each checkpoint interval is constant. It has been shown that the approximate intervals give good approximate to optimal ones.

In Chapter 6, we have taken up the random checkpoint model with two types of checkpoints such as compare-checkpoint and compare-and-store checkpoint, where tasks with random processing times are executed successively. When a double modular system of error detection for each task is adopted, we have considered three schemes, obtained the mean execution times per one task for three schemes, and derived optimal policies which minimize them. It has been shown in numerical examples which scheme is better. Further, extended models with majority decision modules and a spare module have been proposed.

In Chapter 7, we have considered the modified checkpoint model of Chapter 6 where error rates increase with the number of checkpoints. We have obtained the mean

execution times per one task for three schemes and derived analytically optimal policies which minimize them. Further, we have extended a double modular system to a majority system and discussed numerically what a majority system is optimal.

It has been assumed in this thesis that the overheads for the generation of checkpoints and the probability of error occurrences are already known. However, it is important in practical applications to identify what a type of distribution fits the collected data and to estimate several kinds of overheads from the observation of actual models. If such distributions and overheads are given, we can determine optimal policies for recovery models and apply to real systems by modifying them.

Recently, most systems consist of distributed systems as computer network technologies have developed rapidly. A general model of distributed systems is a mobile network system [1]. Coordinated and uncoordinated protocols to achieve checkpointing in such distributed processes have been introduced [1]: Uncoordinated protocols allow each process to take its local checkpoint independently and coordinated protocols force each process to coordinate with other processes to take consistent checkpoints. Two protocols have one's own advantages. A typical advantage of coordinated protocols is to avoid the domino effect. From such viewpoints, a number of techniques of checkpoint protocols have been proposed and their performance have been evaluated [1, 25, 26]. However, there are little research papers to study theoretically optimal policies for checkpoint intervals. Using the methods and techniques used in this thesis, we could analyze optimal intervals of checkpoints for distributed systems.

Bibliography

[1] M. Abd-El-Barr (2007) Reliable and Fault-Tolerant. Imperial Colledge Press, London.

[2] R. E. Barlow and F. Proschan (1965) Mathematical Theory of Reliability. John Wiley and Sons, New York.

[3] A. Birolini (1999) Reliability Engineering Theory and Practice. Springer, New York.

[4] K. M. Chandy and C. V. Ramamoorthy (1972) Rollback and recovery strategies for computer programs. IEEE Transactions on Computers, 21, 546-556.

[5] S. Fukumoto, N. Kaio and S. Osaki (1992) A study of checkpoint generations for a database recovery mechanism. Computers & Mathematics with Applications, 24, 63-70.

[6] E. Gelenbe (2000) System Performance Evaluation. CRC, Boca Roton FL.

[7] K. Ito and T. Nakagawa (1992) Optimal inspection policies for a system in storage. Computers Mathematics with Applications, 24, 87-90.

[8] H. Kim and K. G. Shin (1996) Design and analysis of an optimal instruction-retry policy for TMR controller computers. IEEE Transactions on Computers, 45, 1217-1225.

[9] W. Kuo, V. R. Prsad, F. A. Tillman and C. L. Hwang (2001) Optimal Reliability Design. Cambridge University Press, Cambridge.

[10] P. K. Lala (1985) Fault Tolerant and Faults Testable Hardware. Prentice-Hall, London.

[11] P. K. Lala (2001) Self-Checking and Fault Tolerant Digital Design. Morgan Kaufmann Pub., San Francisco.

[12] P. A. Lee and T. Anderson (1990) Fault Tolerance Principles and Practice. Springer, Wien.

[13] G. Levitin (2007) Computational Intelligence in Reliability Engineering. Springer, Berlin.

[14] Y. Ling, J. Mi and X. Lin (2001) A variational calculus approach to optimal checkpoint placement. IEEE Transactions on Computers, 50, 699-707.

[15] E. C. Martinez (1984) Storage reliability with periodic test. Proceedings of Annual Reliability and Maintainbility Symposium, 181-185.

[16] T. Nakagawa (1980) Optimum inspection policies for a standby unit. Journal of Operations Research Society of Japan, 23, 13-16.

[17] T. Nakagawa (2005) Maintenance Theory of Reliability. Springer, London.

[18] T. Nakagawa and K. Yasui and H. Sandoh (2004) Note on optimal partition problems in reliability models. Journal of Quality in Maintenance Engineering, 10, 282-287.

[19] S. Nakagawa, S. Fukumoto and N. Ishii (1998) Optimal checkpoint interval for redundant error detection and masking systems. Proceeding of the First Euro-Japanese Workshop on Stochastic Risk Modeling for Finance, Insurance, Production and Reliability, vol. II.

[20] S. Nakagawa, S. Fukumoto and N. Ishii (2003) Optimal checkpointing intervals of three error detection schemes by a double modular redundancy. Mathematical and Computing Modeling, 38, 1357-1363.

[21] T. Nakagawa, S. Mizutani and T. Sugiura (2005) Note on the backward time of reliability models. Eleventh ISSAT International Conference on Reliability and Quality in Design, 219-222.

[22] S. Nakagawa, Y. Okuda and S. Yamada (2003) Optimal checkpointing interval for task duplication with spare processing. Ninth ISSAT International Conference on Reliability and Quality in Design, Honolulu, Hawaii, 2003, 215-219.

[23] T. Nanya (1991) Fault Tolerant Computer. Ohm Co., Tokyo.

[24] P. O'Connor (2001) Test Engineering. John Wiley and Sons, Chichester, U.K.

[25] M. Ohara, M. Arai, S. Fukumoto and K. Iwasaki (2005) On the optimal checkpoint interval for uncoordinated checkpointing with a limited number of checkpoints and bound rollbacks. Proceedings of International Workshop on Recent Advances Stochastic Operations Research, 187-194.

[26] M. Ohara, R. Suzuki, M. Arai, S. Fukumoto and K. Iwasaki (2006) Analytical model on hybrid state saving with a limited number of checkpoints and bound rollbacks. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, E89-A, 2386-2395.

[27] S. Osaki (1992) Applied Stochastic System Modeling. Springer, Berlin.

[28] D. K. Pradhan and N. H. Vaidya (1992) Rollforward checkpointing scheme: Concurrent retry with non dedicated spares. IEEE Workshop on Fault-Tolerant Parallel and Distributed Systems, 166-174.

[29] D. K. Pradhan and N. H. Vaidya (1994) Roll-forward and rollback recovery: Performance-reliability trade-off. Proceeding of the 24nd International Symposium on Fault-Tolerant Computings, 186-195.

[30] D. K. Pradhan and N. H. Vaidya (1994) Roll-forward checkpointing scheme: A novel fault-tolerant architecture. IEEE Transaction on Computers, 43, 1163-1174.

[31] A. Reuter (1984) Performance analysis of recovery techniques. ACM Transactions on Database Systems , 9, 526-559.

[32] H. Sandoh and H. Kawai (1995) An economical backup strategy for floppy disks. Mathematical and Computer Modeling, 22, 289-294.

[33] H. Sandoh, N. Kaio and H. Kawai (1992) On backup policies for a hard computer disk. Reliability Engineering and System Safety, 37, 29-32.

[34] K. Shin and Y. Lee (1986) Measurement and application of fault latency. IEEE Transactions on Computers, C-35, 370-375.

[35] D. P. SiewiorekandR. S. Swarz(eds) (1982) The Theory and Practice of Reliable System Design. Digital Press, Bedford, Massachusetts.

[36] K. F. Ssu, B. Yao, W. F. Fuchs and N. F. Neves (1999) Adaptive checkpointing with storage management for mobile environments. IEEE Transactions on Reliability, 48, 315-323.

[37] T. Sugiura, S. Mizutani and T. Nakagawa (2004) Optimal random replacement polices. Tenth ISSAT International Conference on Reliability and Quality in Design, 2004, 99-103.

[38] C. S. Sung, Y. K. Cho and S. H. Song (2003) Combinatorial reliability optimization. H.Pham (ed.) Handbook of Reliability Engineering. Springer, London, 91-144.

[39] L. C. Thomas, I. A. Jacobs and D. I. Gaver (1987) Optimal inspection policies for standby systems. Communications in Statistics Stochastic Models, 3, 259-273.

[40] I. A. Ushakov (1994) Handbook of Reliability Engineering. John Wiley and Sons, New York.

[41] N. H. Vaidya (1995) A case for two-level distributed recovery schemes. Proceedings ACM SIGMETRICS Conference Measurement and Modeling of Computer Systems, 64-73.

[42] N. H. Vaidya (1998) A case for two-level recovery schemes. IEEE Transactions on Computers, 47, 656-666.

[43] K. Yasui, T. Nakagawa and H. Sandoh (2002) Reliability models in data communication systems. S.Osaki (ed.) Stochastic Models in Reliability and Maintenance. Springer, Berlin, 282-301.

[44] A. Ziv and J. Bruck (1997) Performance optimization of checkpointing schemes with task duplication. IEEE Transactions on Computers, 46, 1381-1386.

[45] A. Ziv and J. Bruck (1998) Analysis of checkpointing schemes with task duplication. IEEE Transactions on Computers, 47, 222-227.

List of publications

Chapter 2

1. Kenichiro Naruse, Sayori Nakagawa and Yoshihiro Okuda,
   "Optimal Checking Time of Backup Operation for a Database System",
   Proceedings of International Workshop on Recent Advances In Stochastic Operations Research, 2005 RASOR Canmore, August 25-26, 2005, Canmore, Alberta, Canada, pp.179-186.
2. Kenichiro Naruse, Sayori Nakagawa and Yoshihiro Okuda,
   "Optimal Checking Time of Backup Operation for a Database System",
   Recent Advances in Stochastic Operations Research, World Scientific, Singapore, pp.131-144, 2007.

Chapter 3

1. Kenichiro Naruse, Shizuka Umemura and Sayori Nakagawa,
   "Optimal Checkpointing Interval for Two-Level Recovery Schemes",
   The Second Euro-Japanese Workshop on Stochastic Risk Modelling for Finance, Insurance, Production and Reliability, September 18-20,2002, Chamonix, France, pp 369-375.
2. Kenichiro Naruse, Shizuka Umemura and Sayori Nakagawa,
   "Optimal Checkpointing Interval for Two-Level Recovery Schemes",
   Computers and Mathematics with Applications, Vol. 51, No.2, pp. 371-376, 2006.
3. Kenichiro Naruse, Sayori Nakagawa, Shigeru Yamada,
   "Optimal Checkpointing Number for Multiple Recovery Schemes",
   Proceedings of the Tenth ISSAT International Conference on Reliability and Quality in Design, August 5-7, 2004, Las Vegas, Nevada, U.S.A., pp. 129-132.

Chapter 4

1. Kenichiro Naruse, Toshio Nakagawa and Sayori Maeji,
   "Optimal Checkpoint Intervals for Error Detection by Multiple Modular Redundancies",
   Advanced Reliability Modeling II : Reliability Testing and Improvement: Proceedings of the 2nd Asian International Workshop (AIWARM 2006) Busan, August 24-26, 2006, pp. 293-300.

Chapter 5

1. Kenichiro Naruse, Toshio Nakagawa and Sayori Maeji,
   "Optimal Sequential Checkpoint Intervals for Error Detection",
   Proceedings of International Workshop on Recent Advances in Stochastic Operations Research II, 2007 RASOR Nanzan, March 5-6, 2007, Nanzan University, Nagoya, pp. 185-191.
2. Kenichiro Naruse, Toshio Nakagawa and Sayori Maeji,
   "Optimal Sequential Checkpoint Intervals for Error Detection",
   Accepted for publication in the Special Volume of 2007 RASOR Nanzan.

Chapter 6

1. Kenichiro Naruse, Toshio Nakagawa and Sayori Maeji,
   "Random Checkpoint Models for a Double Modular System",
   13th ISSAT International Conference on Reliability and Quality in Design, August 2-4, 2007, Seattle, Washington, U.S.A, pp.231-235.

Chapter 7

1. Toshio Nakagawa, Kenichiro Naruse and Sayori Maeji,
   "Random Checkpoint Models for Multiple Modular Systems with Increasing Error Rates",
   Submit to 5th International Service Availability Symposium, May 19-21, 2008, Tokyo University, Tokyo, Japan.