Analysis of a Class of Recovery Procedures

ISRAEL KOREN, MEMBER, IEEE, ZAHAVA KOREN, AND STEPHEN Y. H. SU, SENIOR MEMBER, IEEE

Abstract—**Recovery procedures involving time redundancy in the form of instruction retries and program rollbacks have proved to be very effective against transient failures in computer systems. A class of such recovery procedures is presented and analyzed here, and the parameters of each procedure are determined so that the system's operation is optimized. These procedures are then compared in order to select the most appropriate one for given system parameters.**

Index Terms—**Checkpoint, error latency, error recovery procedures, instruction retry, intermittent faults, permanent faults, program rollback.**

I. Introduction

RELIABILITY and availability of a computer system may be increased by incorporating redundancy into the system. This may be either hardware redundancy or software redundancy (time redundancy) or both. Time redundancy, i.e., repeating an instruction or a section of the program which has failed, requires relatively few hardware and software resources and has proved to be very effective against intermittent failures which constitute the majority of failures occurring in computer systems. By using time redundancy, the system may recover from the effects of the fault at the cost of the time needed to repeat an instruction, a section of the program or the complete program. Various recovery strategies may be used for this purpose, each consisting of several recovery steps, which are applied successively as long as the system has not fully recovered from the effect of the fault.

Several researchers have analyzed such recovery procedures [1], [3], [8]–[10]. These works differ in some of their assumptions (like the possibility of errors during recovery) and in their objectives. Most of them are concerned with maximizing the system availability, while others have different goals like minimizing the number of delayed transactions [1] or minimizing the mean response time of transactions [3].

The main recovery technique analyzed in these studies is the program rollback. Our purpose here is to analyze somewhat more general recovery procedures which consist of instruction retries in addition to program rollbacks. Our analysis also takes into account the various types of permanent and intermittent faults that may occur. All these types of faults have different effects on the system's operation and therefore, may have different recovery techniques which are effective against them. In addition, we consider the possibility of faults with a latency period, while a common assumption in most previous works was the coincidence of error occurrence and error detection.

The recovery procedures we examine in this paper are based on two recovery techniques, namely, instruction retry and program rollback. Whenever an error is detected (note that a continuous error-checking mechanism is necessary for any recovery procedure), an attempt to identify the failing instruction is made and this instruction is then repeated [2], [5]. The instruction retry may fail because either the intermittent fault persists, or the damage caused by the fault is so severe that instruction retry is ineffective against it, or the faulty instruction has not been correctly identified. Since the cost (in wasted time) of instruction retry is considerably lower than that of other recovery techniques, it might be worthwhile to repeat the failing instruction more than once up to a predetermined number of times (denoted by K_T) to take care of persistent faults.

If all K_T instruction retries are unsuccessful, the second recovery step, which is program rollback, is initiated. To avoid having to repeat the program from its beginning, it is common practice to periodically save sufficient information to enable the system to restart the program at the last point at which information was saved. These points are called *checkpoints* and the *intercheckpoint interval* is one of the most important parameters of the rollback technique [1], [3], [8], [10].

If the rollback operation fails, we may declare that the program has failed and resort to more severe methods of recovery like reloading the entire program into memory and restarting it. However, less costly steps like repeating the rollback to the last checkpoint or even rolling the program back to an earlier checkpoint may prove to be effective, especially when intermittent faults with long latency periods can be expected [6]. If all these recovery steps fail, one has to debug the system in order to locate permanent faults.

In the next section, some definitions and notations are introduced. In Sections III, IV and V, three different recovery procedures are analyzed. These procedures are then compared in Section VI for several values of system parameters.

II. Preliminaries

The probability of a fault occurrence during the execution of a machine instruction depends upon the functional units used when executing the instruction and the execution time of the instruction. A simple instruction using only a few system units

Manuscript received January 2, 1985; revised August 8, 1985. This work was supported in part by the National Science Foundation under Grant MCS78-24323.

I. Koren is with the Department of Electrical Engineering, Technion—Israel Institute of Technology, Haifa 32000, Israel.

Z. Koren is with the Department of Computer Science, Technion—Israel Institute of Technology, Haifa 32000, Israel.

S. Y. H. Su is with the Department of Computer Science, State University of New York, Binghamton, NY 13901.

IEEE Log Number 8609401.

is more likely to be executed successfully than an instruction employing a large number of system units, since faults (and especially intermittent ones) occurring in unused units are not expected to introduce errors in the instruction's results. Consequently, we partition the instruction set of the computer system into N subsets. Two machine instructions are in the same subset if their execution time is the same and the same system units are used while these instructions are executed (e.g., the instructions ADD, SUB, and COMPARE in fixed-point arithmetic are usually in the same subset). We denote by T_i the execution time of an instruction of type i, and by f_i the frequency at which such an instruction is being executed. Clearly, $\Sigma_{i=l}^{N} f_i = 1$.

Let λ_i denote the rate at which faults occur while executing an instruction of type i. We adopt here the viewpoint that fault occurrences obey a Poisson process [1], [4], [7]. However, we distinguish between permanent faults which cause a system failure, and intermittent faults, which in most cases can be recovered from. Let s denote the fraction of faults that are permanent; $s\lambda_i$ is therefore the rate of permanent fault occurrences while an instruction of type i is being executed. For the intermittent faults we adopt the continuous parameter Markov model [1], [4], [7], namely, at the rate $(1 - s)\lambda_i$ the fault becomes active causing the system to malfunction, and when active it becomes inactive at a rate μ_i, allowing the system to operate correctly. When the intermittent fault becomes inactive, we may try to recover from its effect by repeating the instruction. However, instruction retry is not always effective against intermittent faults even if the failing instruction has been correctly identified; for example, if some of the data needed for repeating the instruction is not available any longer [5].

Clearly, an instruction involving a large number of data movements is more likely to be not recoverable from the effects of an intermittent fault by instruction retry than an instruction with a few data movements. We denote by r_i the percentage of intermittent faults not recoverable by instruction retry when executing an instruction of type i. Consequently, the failure rate λ_i is divided into the following.

1) The rate of intermittent faults recoverable by instruction retry $\lambda_i^{(1)} = \lambda_i(1 - s)(1 - r_i)$.

2) The rate of intermittent faults not recoverable by instruction retry $\lambda_i^{(2)} = \lambda_i(1 - s)r_i$.

3) The rate of permanent faults $\lambda_i^{(3)} = \lambda_i s$.

The correct identification of the failing instruction is a necessary condition for the success of the instruction retry step. A major reason for incorrect identification of the failing instruction is a latency period between the occurrence of the fault and the manifestation of the consequent error [6]. An additional latency period may appear between the occurrence of the error and its detection. However, we may for our purposes lump these two latency periods into one.

We denote by α_i the ratio of faults occurring in instruction i whose short latency period allows correct identification of the failing instruction. Hence, the ratio of the faults whose long latency period results in an incorrect identification of the failing instruction is $1 - \alpha_i$. For mathematical tractability we assume that the ratio α_i is the same for all three types of faults 1), 2), and 3). Consequently, each of the three failure rates $\lambda_i^{(j)}$ is further divided into two rates of $\lambda_i^{(j)}\alpha_i$ and $\lambda_i^{(j)}(1 - \alpha_i)$; $j = 1, 2, 3$; $i = 1, \cdots, N$.

To analyze a recovery procedure, we may view the process of executing instructions as a renewal process. The time periods necessary to *successfully* complete the execution of consecutive instructions form a series of independent, identically distributed random variables, with the starting instants of the instructions being the renewal points. Randomness is introduced into the process by two factors: the type of the next instruction is random, having the probability distribution $(f_1, f_2, \cdots, f_N)$, and the instruction may be correctly executed, or fail and necessitate retries, a program rollback or even reloading and restarting the program. Our analysis consists of investigating the stochastic behavior of these renewal periods and minimizing their average length.

Clearly, the various courses that an instruction execution may take depend upon the recovery procedure employed. All the recovery procedures analyzed in this paper are special cases of a general recovery procedure which can be represented by a vector (K_T, K_B, K_E) with the following interpretation. Whenever an error is detected, a supposedly "failing" instruction is identified. It is then retried up to K_T times. If all K_T retries fail, up to K_B rollbacks to the last checkpoint are performed. If all K_B rollbacks fail, up to K_E rollbacks to the earlier checkpoint are performed. If all K_E earlier rollbacks fail, we say that we have a *program failure* (caused by hardware and not by software failure). In this case the system is diagnosed and if no permanent failures are detected, the program is reloaded and restarted. If a permanent fault is detected, it has to be repaired before reloading the program.

Each instruction when being executed must therefore result in one of the following mutually exclusive events.

$H^{(c)}$ The instruction is *completed* successfully (without retry) when first executed and there is no undetected fault in the system.

$H^{(j)}$ The instruction fails, is correctly identified and the jth *retry* is the first successful one;

$$j = 1, 2, \cdots, K_T.$$

$H^{(RB,j)}$ The instruction fails and is correctly identified; all K_T retries fail but the instruction is completed successfully after the jth program *rollback*;

$$j = 1, 2, \cdots, K_B.$$

$H^{(ERB,j)}$ The instruction fails and is correctly identified; all K_T retries and all K_B rollbacks fail, but the instruction is completed successfully after the jth program rollback to an *earlier* checkpoint;

$$j = 1, 2, \cdots, K_E.$$

$H^{(PF)}$ The instruction fails and is correctly identified but the K_T retries and the program rollbacks fail, resulting in a *program failure* after which the program is reloaded and restarted.

$H^{(RBL,j)}$ A fault occurs while executing the instruction and, due to a *latency* period, a wrong instruction is repeated K_T times. The instruction is completed successfully after the jth program *rollback*:

$$j = 1, 2, \cdots, K_B.$$

$H^{(ERBL,j)}$ A fault occurs while executing the instruction and, due to a *latency* period, a wrong instruction is repeated K_T times. K_B rollbacks fail but the instruction is completed successfully after the jth program rollback to an *earlier* checkpoint;

$j=1, 2, \cdots, K_E$.

$H^{(PFL)}$ A fault occurs while executing the instruction and, due to a latency period, a wrong instruction is repeated K_T times. The program rollbacks fail, resulting in a *program failure* after which the program is reloaded and restarted.

Due to the large number of possible events and the complexity of the formulas in the general case, we will consider the following special cases. First the $(k, 1, 0)$ procedure is analyzed in Section III. Next, the procedures $(k, 2, 0)$, i.e., two rollbacks, and $(k, 1, 1)$ i.e., one rollback and one rollback to an earlier checkpoint, are analyzed in Sections IV and V, respectively.

Once these three procedures have been analyzed, they can be compared for various values of system parameters in order to determine the best one for a given computing system. This comparison is presented in Section VI.

III. The $(k, 1, 0)$ Procedure

In the $(k, 1, 0)$ recovery procedure we first repeat the instruction identified as the failing one up to k times, and if all k retries fail we rollback the program only once to the last checkpoint. Our objective in analyzing the procedure is to determine the optimal values of k and of the number of instructions executed between two successive checkpoints (denoted by M) so as to minimize the average time spent per instruction. We derive next the probabilities of the H events for the $(k, 1, 0)$ procedure. Denote by

$$P_0(\lambda, t)=e^{-\lambda t} \tag{3.1}$$

the probability of no fault occurrences during the time interval $(0, t)$, for a failure rate λ.

In addition,

$$P_{00}(\lambda, \mu, t)=\frac{\mu}{\lambda+\mu}+\frac{\lambda}{\lambda+\mu}e^{-(\lambda+\mu)t} \tag{3.2}$$

is the probability of a transition from the fault-free state of the system at time 0 to the fault-free state at time t, for a model with failure rate λ and "repair" rate μ.

Finally,

$$\bar{P}_{00}(\lambda, \mu, t_1, t)=P_{00}(\lambda, \mu, t)-e^{-\lambda t_1}P_{00}(\lambda, \mu, t-t_1) \tag{3.3}$$

is defined similarly to P_{00} except that at least one fault occurs in $(0, t_1)$.

Using these definitions we can derive the probabilities of the events $H^{(c)}, H^{(j)}(j = 1, 2, \cdots, k), H^{(RB)}, H^{(PF)}, H^{(RBL)}$, and $H^{(PFL)}$, denoted by $P^{(c)}, P^{(j)}(j = 1, 2, \cdots, k), P^{(RB)}, P^{(PF)}, P^{(RBL)}$ and $P^{(PFL)}$, respectively. These probabilities clearly satisfy

$$P^{(c)}+\sum_{j=1}^{k}P^{(j)}+P^{(RB)}+P^{(PF)}+P^{(RBL)}+P^{(PFL)}=1. \tag{3.4}$$

Let $P_i^{(c)}, P_i^{(j)}(j = 1, 2, \cdots, k), P_i^{(RB)}, P_i^{(PF)}, P_i^{(RBL)}$, and $P_i^{(PFL)}$ for $i = 1, 2, \cdots, N$ denote the conditional probabilities of the above events, given that the instruction is of type i. These probabilities satisfy

$$P_i^{(c)}+\sum_{j=1}^{k}P_i^{(j)}+P_i^{(RB)}+P_i^{(PF)}+P_i^{(RBL)}+P_i^{(PFL)}=1; \quad i=1, 2, \cdots, N.$$

Once these conditional probabilities are calculated, the unconditional probabilities can be found by averaging over i with respect to $f_1, \cdots, f_N$, e.g.,

$$P^{(c)}=\sum_{i=1}^{N}f_iP_i^{(c)} \tag{3.5}$$

$P^{(j)}(j = 1, 2, \cdots, k), P^{(RB)}, P^{(PF)}, P^{(RBL)}$, and $P^{(PFL)}$ are obtained similarly.

We proceed now to calculate these conditional probabilities. Since a Poisson process is assumed for the faults, we have

$$P_i^{(c)}=e^{-\lambda_iT_i}. \tag{3.6}$$

$P_i^{(j)}$ is the probability of a successful jth retry given a type i instruction and is calculated as follows:

$$\begin{aligned}P_i^{(j)}&=P(H^{(j)}/\text{type } i \text{ instruction})\\&=P_0(\lambda_i-\lambda_i^{(1)}\alpha_i, T_i+\delta_1) * P_0(\lambda_i^{(1)}\alpha_i, T_i)\\&\quad * \Big[\bar{P}_{00}(\lambda_i^{(1)}\alpha_i, \mu_i, T_i, j(T_i+\delta_1))\\&\quad * P_0(\lambda_i-\lambda_i^{(1)}\alpha_i, jT_i+(j-1)\delta_1)\\&\quad -\sum_{l=1}^{j-1}P_i^{(l)} * P_{00}(\lambda_i^{(1)}\alpha_i, \mu_i, (j-l)\delta_1+(j-l-1)T_i)\\&\quad * P_0(\lambda_i-\lambda_i^{(1)}\alpha_i, (j-l-1)(T_i+\delta_1))\Big]; \; j=1, \cdots, k\end{aligned} \tag{3.7}$$

where δ_1 is the set up time needed to initiate an instruction retry.

The first two terms within the brackets constitute the probability of the system operating correctly at the end of the jth retry, from which we subtract the probability of any successful retry prior to the jth one. The first two terms multiplying the brackets are the probability that no other than a nonlatent fault of type 1 occur during the recovery period, and the probability that no faults occur during the last retry, respectively.

To obtain $P_i^{(RB)}$, the probability of a successful rollback, we denote by m the number of program instructions between the last checkpoint and the failing instruction. The variable m is random, assuming the values $1, 2, \cdots, M$ with probability $1/M$ each (recall that M is the number of instructions between two consecutive checkpoints).

We denote by $P_{i,m}^{(RB)}$ the conditional probability of a successful rollback, given the values of i and m, i.e.,

$$P_{i,m}^{(RB)}=P(H^{(RB)}/\text{type } i \text{ instruction} \cap m \text{ instructions since last checkpoint}).$$

$P_{i,m}^{(RB)}$ is the probability of the system recovering after the rollback but not earlier than that. Hence, in order to calculate it, one must subtract the probability of any successful instruction retry from the probability that the system operates correctly at the end of the rollback. The expression derived for $P_{i,m}^{(RB)}$ is therefore,

$$\begin{aligned} P_{i,m}^{(RB)} = \Bigg[& \alpha_i \bar{P}_{00}(\lambda_i^{(1)} + \lambda_i^{(2)},\ \mu_i,\ T_i,\ k\delta_1 + (k+1)T_i) \\ & * P_0(\lambda_i^{(3)},\ k\delta_1 + (k+1)T_i) \\ & - \sum_{l=1}^{k} P_i^{(l)} * P_{00}(\lambda_i^{(1)} + \lambda_i^{(2)},\ \mu_i,\ (k-l)(\delta_1 + T_i)) \\ & * P_0(\lambda_i^{(3)},\ (k-l)(\delta_1 + T_i)) \Bigg] \\ & * P_0(\lambda_i,\ \delta_2)[P^{(c)}]^{m-1} P_i^{(c)} \end{aligned} \tag{3.8}$$

where δ_2 is the setup time needed to initiate a program rollback including the time needed to load the information saved in the last checkpoint. The last two terms in (3.8) constitute the probability of no faults during the rollback period including the setup time.

Clearly,

$$P_i^{(RB)} = \frac{1}{M} \sum_{m=1}^{M} P_{i,m}^{(RB)}. \tag{3.9}$$

Substituting (3.8) into (3.9) yields

$$\begin{aligned} P_i^{(RB)} = \Bigg[& \alpha_i \bar{P}_{00}(\lambda_i^{(1)} + \lambda_i^{(2)},\ \mu_i,\ T_i,\ k\delta_1 + (k+1)T_i) \\ & * P_0(\lambda_i^{(3)},\ k\delta_1 + (k+1)T_i) \\ & - \sum_{l=1}^{k} P_i^{(l)} * P_{00}(\lambda_i^{(1)} + \lambda_i^{(2)},\ \mu_i,\ (k-l)(\delta_1 + T_i)) \\ & * P_0(\lambda_i^{(3)},\ (k-l)(\delta_1 + T_i)) \Bigg] \\ & * P_0(\lambda_i,\ \delta_2) * \frac{P_i^{(c)}}{M} \frac{1 - [P^{(c)}]^M}{1 - P^{(c)}}. \end{aligned} \tag{3.10}$$

We next obtain an expression for $P_i^{(RBL)}$, the probability of a successful rollback for a latent fault. Denote by l the length of the latency period of the fault, i.e., the number of instructions between the occurrence of the fault and its manifestation. Note that although l can assume any integer value, the rollback will not succeed for l greater than $M - m$ since, in that case, the information saved at the last checkpoint is erroneous. Consequently, for calculating the probability of a successful rollback, we consider only the values $l = 1, 2, \cdots, M - m$ with probabilities

$$(1 - \bar{\alpha})^{l-1} \bar{\alpha}; \quad (l = 1,\ 2,\ \cdots,\ M - m)$$

where $\bar{\alpha}$ is the average probability that a latent fault is detected, i.e., $\bar{\alpha} = \Sigma_{i=1}^{N} f_i \alpha_i$. For given values of i, m, and l denote

$$\begin{aligned} P_{i,m,l}^{(RBL)} = P(H^{(RBL)}/ & \text{type } i \text{ instruction} \\ & \cap\ m \text{ instructions between the last checkpoint and the failing instruction} \\ & \cap\ \text{latency period of } l). \end{aligned}$$

Note that for a latent fault, instruction retry is never successful since a wrong instruction is repeated. Thus, for a rollback to succeed the following events must take place. First, all intermittent faults have to be inactive at the beginning of the rollback, no permanent faults should occur, and, finally, no faults of any type should occur during the rollback itself.

The resulting expression is

$$\begin{aligned} P_{i,m,l}^{(RBL)} = (1 - \alpha_i) & \bar{P}_{00}(\lambda_i^{(1)} + \lambda_i^{(2)}, \\ & \mu_i,\ T_i,\ T_i + l\bar{T} + k(\bar{T} + \delta_1)) \\ & * P_0(\lambda_i^{(3)},\ T_i + l\bar{T} + k(\bar{T} + \delta_1)) * P_0(\lambda_i,\ \delta_2) * (P^{(c)})^m. \end{aligned} \tag{3.11}$$

Averaging $P_{i,m,l}^{(RBL)}$ using the probabilities of m and l, we obtain

$$P_i^{(RBL)} = \sum_{m=1}^{M} \sum_{l=1}^{M-m} \frac{1}{M} (1 - \bar{\alpha})^{l-1} \bar{\alpha} P_{i,m,l}^{(RBL)}. \tag{3.12}$$

To derive an expression for the remaining two probabilities note that $\alpha_i(1 - P_i^{(c)}) = \alpha_i(1 - e^{-\lambda_i T_i})$ is the probability of a nonlatent fault occurring in a type i instruction and is therefore equal to

$$\sum_{j=1}^{k} P_i^{(j)} + P_i^{(RB)} + P_i^{(PF)}.$$

Consequently

$$P_i^{(PF)} = \alpha_i(1 - e^{-\lambda_i T_i}) - \sum_{j=1}^{k} P_i^{(j)} - P_i^{(RB)}. \tag{3.13}$$

$P_i^{(PFL)}$ is calculated in a similar way, using the fact that the probability of a latent fault occurring in a type i instruction is $(1 - \alpha_i)(1 - e^{-\lambda_i T_i})$, it follows that

$$P_i^{(PFL)} = (1 - \alpha_i)(1 - e^{-\lambda_i T_i}) - P_i^{(RBL)} \tag{3.14}$$

The calculation of $P^{(c)}$ and the rest of the unconditional probabilities is now straightforward using (3.5).

Our objective is to determine k and M so as to maximize the number of instructions executed per unit of time. This is equivalent to minimizing the average time spent per instruction, denoted by $\bar{w}$. It consists of the average time required to successfully execute an instruction (denoted by $\bar{\tau}$) and the time lost per instruction due to checkpointing, i.e.,

$$\bar{w} = \bar{\tau} + \frac{T_s}{M} \tag{3.15}$$

where T_s denotes the time needed to store the information at each checkpoint and is called *checkpointing time.*

To derive an expression for $\bar{\tau}$ we introduce the following notations. Let $\bar{T}$ be the average execution time of an instruction, i.e.,

$$\bar{T}=\sum_{i=1}^{N} f_i T_i. \tag{3.16}$$

Let $\bar{T}^{(j)}(j = 1, \cdots, k)$, $\bar{T}^{(RB)}$, and $\bar{T}^{(PF)}$ be the conditional expectations of an instruction execution time given the events $H^{(j)}$, $H^{(RB)}$, and $H^{(PF)}$, respectively. For example,

$$\bar{T}^{(j)}=\frac{\sum_{i=1}^{N} f_i P_i^{(j)} T_i}{P^{(j)}} \quad (j=1, \cdots, k). \tag{3.17}$$

The formulas for $\bar{T}^{(RB)}$ and $\bar{T}^{(PF)}$ are similar to (3.17).

Denote by δ_3 the average time required to diagnose and repair the system and by L the average number of instructions per program; $\bar{\tau}$ can be expressed as

$$\begin{aligned}\bar{\tau}=\bar{T}+\sum_{j=1}^{k} P^{(j)}[j(\bar{T}^{(j)}+\delta_1)]\\ +P^{(RB)}\left[k(\bar{T}^{(RB)}+\delta_1)+\delta_2+\frac{M+1}{2}\bar{T}\right]\\ +P^{(PF)}\left[k(\bar{T}^{(PF)}+\delta_1)+\delta_2+\frac{M+1}{2}\bar{T}+\delta_3+\frac{L+1}{2}\bar{w}\right]\\ +P^{(RBL)}\left[k(\bar{T}+\delta_1)+\frac{\bar{T}}{\bar{\alpha}}+\delta_2+\frac{M+1}{2}\bar{T}\right]\\ +P^{(PFL)}\left[k(\bar{T}+\delta_1)+\frac{\bar{T}}{\bar{\alpha}}\right.\\ \left.+\delta_2+\frac{M+1}{2}\bar{T}+\delta_3+\frac{L+1}{2}\bar{w}\right].\end{aligned} \tag{3.18}$$

Each of the terms in the square brackets multiplying the different probabilities is the time, in addition to $\bar{T}$, that is needed to complete the execution of the instruction for the corresponding event, e.g., $j(\bar{T}^{(j)} + \delta_1)$ is the additional time needed when the jth instruction retry succeeds. The term $(M+1)/2$ is the average number of instructions reexecuted whenever the program is rolled back; hence, $(M+1)/2\cdot\bar{T}$ is the average rollback time. The term $\bar{T}/\bar{\alpha}$ is the average latency period since $1/\bar{\alpha} = \sum_{l=1}^{\infty} l(1 - \bar{\alpha})^{l-1}\bar{\alpha}$ is the average number of instructions executed until the detection of a latent fault. Finally, the term $(L + 1)/2\cdot\bar{w}$ is the average time required for reexecuting the program whenever a program failure occurs.

Substituting (3.18) in (3.15) and solving for $\bar{w}$ yields

$$\begin{aligned}\bar{w}=\frac{1}{1-\frac{L+1}{2}(P^{(PF)}+P^{(PFL)})}\\ \cdot\left\{\bar{T}+\sum_{j=1}^{k} P^{(j)}[j(\bar{T}^{(j)}+\delta_1)]\right.\\ +P^{(RB)}\left[k(\bar{T}^{(RB)}+\delta_1)+\delta_2+\frac{M+1}{2}\bar{T}\right]\\ +P^{(PF)}\left[k(\bar{T}^{(PF)}+\delta_1)+\delta_2+\frac{M+1}{2}\bar{T}+\delta_3\right]\\ +P^{(RBL)}\left[k(\bar{T}+\delta_1)+\frac{\bar{T}}{\bar{\alpha}}+\delta_2+\frac{M+1}{2}\bar{T}\right]\\ \left.+P^{(PFL)}\left[k(\bar{T}+\delta_1)+\frac{\bar{T}}{\bar{\alpha}}+\delta_2+\frac{M+1}{2}\bar{T}+\delta_3\right]+\frac{T_s}{M}\right\}.\end{aligned} \tag{3.19}$$

Expression (3.19) can be minimized with respect to k and M to find the optimal recovery procedure parameters k_{opt} and M_{opt}. The numerical search for k_{opt} proved to be simple and k_{opt} was found to be less than 20 for all cases examined. To find an initial value for the numerical search for M_{opt} we derive an approximate expression for $\bar{w}$ by replacing the denominator in (3.19) (which is nearly 1) by 1. By taking the derivative of the remaining expression with respect to M and letting it equal 0, we obtain

$$M_{\text{opt}}\simeq\sqrt{\frac{2T_s}{\bar{T}(P^{(RB)}+P^{(PF)}+P^{(RBL)}+P^{(PFL)})}} \tag{3.20}$$

which resembles the expressions derived in [1] and [10].

IV. The $(k, 2, 0)$ Procedure

In this recovery procedure, we first retry the failing instruction (or the one that we believe is the failing one due to latency of the fault) up to k times. We then roll the program back to the last checkpoint up to two times and only if all these attempts of recovery fail, we declare a program failure.

It is clear that by adding a second rollback, the probability of a system failure decreases compared to the $(k, 1, 0)$ procedure. However, the $(k, 2, 0)$ procedure is not always superior to the previous one. Although it may take longer to reach a program failure, part of this time may be wasted on a useless additional rollback. In Section VI we present numerical examples in which this procedure is preferred over the single rollback procedure.

To analyze this procedure and derive the optimal values of k and M (using the same cost function as in Section III) we have first to calculate the probabilities of the following events.

$$H^{(c)};\ H^{(j)}[j=1, \cdots, k];\ H^{(RB,1)};$$
$$H^{(RB,2)};\ H^{(RBL,1)};\ H^{(RBL,2)};\ H^{(PF)};\ H^{(PFL)}.$$

Since the success of any instruction retry or of the first rollback are independent of the following steps of the recovery procedure, the probabilities $P_i^{(c)}$, $P_i^{(j)} (j = 1, \cdots, k)$ are the same for the $(k, 2, 0)$ procedure as for the $(k, 1, 0)$ one. Similarly, $P_i^{(RB,1)} = P_i^{(RB)}$; $P_i^{(RBL,1)} = P_i^{(RBL)}$.

As in Section III,

$$P_i^{(PF)} = \alpha_i(1-e^{-\lambda_i T_i}) - \sum_{j=1}^{k} P_i^{(j)} - P_i^{(RB,1)} - P_i^{(RB,2)} \quad (4.1)$$

$$P_i^{(PFL)} = (1-\alpha_i)(1-e^{-\lambda_i T_i}) - P_i^{(RBL,1)} - P_i^{(RBL,2)}. \quad (4.2)$$

In order to calculate $P_i^{(RB,2)}$ and $P_i^{(RBL,2)}$, the probabilities of success in the second rollback, for a nonlatent and latent fault, respectively, denote

$$P_{i,m}^{(RB,2)} = P(H^{(RB,2)}/\text{type } i \text{ instruction} \cap m \text{ instructions since last checkpoint})$$

and

$$P_{i,m,l}^{(RBL,2)} = P(H^{(RBL,2)}/\text{type } i \text{ instruction} \cap m \text{ instructions between the last checkpoint and the failing instruction} \cap \text{latency period of } l).$$

To derive an expression for $P_{i,m}^{(RB,2)}$ note that for the event $H^{(RB,2)}$ to occur, all k retries and the first rollback must fail. We therefore subtract the probabilities of a successful retry or a successful first rollback, from the probability that the system is operating correctly at the end of the second rollback. This yields

$$\begin{aligned} P_{i,m}^{(RB,2)} = \Big\{ & \alpha_i \bar{P}_{00}(\lambda_i^{(1)} + \lambda_i^{(2)}, \mu_i, T_i, k\delta_1 \\ & + (k+1)T_i + \delta_2 + m\bar{T}) \\ & * P_0(\lambda_i^{(3)}, k(T_i+\delta_1) + T_i + \delta_2 + m\bar{T}) - \sum_{l=1}^{k} P_i^{(l)} \\ & * P_{00}(\lambda_i^{(1)} + \lambda_i^{(2)}, \mu_i, (k-l)(T_i+\delta_1) + \delta_2 + m\bar{T}) \\ & * P_0(\lambda_i^{(3)}, (k-l)(T_i+\delta_1) + \delta_2 + m\bar{T}) \\ & - P_{i,m}^{(RB,1)} \Big\} P_0(\lambda_i, \delta_2) * (P^{(c)})^m \end{aligned} \quad (4.3)$$

and clearly

$$P_i^{(RB,2)} = \sum_{m=1}^{M} \frac{1}{M} P_{i,m}^{(RB,2)}. \quad (4.4)$$

To calculate $P_{i,m,l}^{(RBL,2)}$, we subtract from the probability that a latent fault occurred but there were no faults present at the end of the second rollback, the probability of a successful first rollback. Since for a latent fault no instruction retry will succeed, these probabilities need not be subtracted.

Hence,

$$\begin{aligned} P_{i,m,l}^{(RBL,2)} = & (1-\alpha_i) P_0(\lambda_i, \delta_2)(P^{(c)})^m \\ & * P_0(\lambda_i^{(3)}, T_i + l\bar{T} + k(\bar{T}+\delta_1)) \\ & * \{\bar{P}_{00}(\lambda_i^{(1)} + \lambda_i^{(2)}, \mu_i, T_i, T_i \\ & + l\bar{T} + k(\bar{T}+\delta_1) + \delta_2 + m\bar{T}) \\ & * P_0(\lambda_i^{(3)}, \delta_2 + m\bar{T}) - P_{00}(\lambda_i^{(1)} + \lambda_i^{(2)}, \mu_i, T_i, T_i \\ & + l T + k(T+\delta_1)) * P_0(\lambda_i, \delta_2)(P^{(c)})^m\}. \end{aligned} \quad (4.5)$$

Averaging (4.5) over m and l yields

$$P_i^{(RBL,2)} = \sum_{m=1}^{M} \frac{1}{M} \sum_{l=1}^{M-m} \bar{\alpha}(1-\bar{\alpha})^{l-1} P_{i,m,l}^{(RBL,2)}. \quad (4.6)$$

As in Section III, the cost function to be minimized with respect to k and M was chosen to be $\bar{w}$, the average time spent per instruction. The expression for $\bar{w}$ is,

$$\bar{w} = \bar{\tau} + \frac{T_s}{M}. \quad (4.7)$$

To obtain the formula for $\bar{\tau}$, the average time required to successfully execute an instruction, note that some of the recovery periods remain the same as for the $(k, 1, 0)$ model, namely, the periods associated with a successful retry or a successful first rollback. There is an additional time of $\delta_2 + (M+1)/2 \cdot \bar{T}$ (which is the time needed for the second rollback) whenever one of the events $H^{(RB,2)}$, $H^{(RBL,2)}$, $H^{(PF)}$ and $H^{(PFL)}$ occurs. Substituting the resulting expression for $\bar{\tau}$ in (4.7) and solving for $\bar{w}$ yields

$$\begin{aligned} \bar{w} = & \frac{1}{1 - \frac{L+1}{2}(P^{(PF)} + P^{(PFL)})} \\ & \cdot \Big\{ \bar{T} + \sum_{j=1}^{k} P^{(j)}[j(\bar{T}^{(j)} + \delta_1)] \\ & + P^{(RB,1)} \left[k(\bar{T}^{(RB,1)} + \delta_1) + \delta_2 + \frac{M+1}{2}\bar{T} \right] \\ & + P^{(RB,2)} \left[k(\bar{T}^{(RB,2)} + \delta_1) + 2\left(\delta_2 + \frac{M+1}{2}\bar{T}\right) \right] \\ & + P^{(PF)} \left[k(\bar{T}^{(PF)} + \delta_1) + 2\left(\delta_2 + \frac{M+1}{2}\bar{T}\right) + \delta_3 \right] \\ & + P^{(RBL,1)} \left[k(\bar{T} + \delta_1) + \frac{\bar{T}}{\bar{\alpha}} + \delta_2 + \frac{M+1}{2}\bar{T} \right] \\ & + P^{(RBL,2)} \left[k(\bar{T} + \delta_1) + \frac{\bar{T}}{\bar{\alpha}} + 2\left(\delta_2 + \frac{M+1}{2}\bar{T}\right) \right] \\ & + P^{(PFL)} \Big[k(\bar{T} + \delta_1) + \frac{\bar{T}}{\bar{\alpha}} \\ & + 2\left(\delta_2 + \frac{M+1}{2}\bar{T}\right) + \delta_3 \Big] + \frac{T_s}{M} \Big\}. \end{aligned} \quad (4.8)$$

A numerical search should now be performed to yield the optimal values of k and M that minimize $\bar{w}$.

V. The $(k, 1, 1)$ Procedure

In this recovery procedure, after all k instruction retries and the program rollback to the most recent checkpoint have failed, the program is rolled back to an earlier checkpoint. This policy seems hardly worth considering, since the last checkpoint is assumed to contain all the relevant information. Yet it may prove useful to roll the program back to an earlier checkpoint rather than to the most recent one whenever there is a chance of latent faults with long latency periods. Consider, for example, a fault occurring shortly before the last checkpoint and whose resultant error is detected only after it. The information saved at the checkpoint in such a case might be erroneous and a simple rollback may not succeed, while a rollback to an earlier checkpoint may succeed and avoid the need to reload and restart the program. In Section VI we will present some numerical values of system parameters for which the $(k, 1, 1)$ procedure is preferable to the two procedures discussed before, although such cases are very rare.

The analysis of the $(k, 1, 1)$ procedure is similar to that of the previous two procedures. We enumerate the possible outcomes of an instruction execution, calculate the probabilities of these outcomes, find the formula for the average time spent per instruction and minimize it with respect to k and M.

The possible H events for the $(k, 1, 1)$ procedure are

$$H^{(c)},\ H^{(j)}(j=1,\cdots,k),\ H^{(RB)},\ H^{(RBL)},\ H^{(ERB)},\ H^{(ERBL)},\ H^{(PF)},\ \text{and}\ H^{(PFL)}.$$

Since this procedure and the $(k, 2, 0)$ procedure start with the same recovery steps (i.e., k instruction retries and a rollback), both have the same probabilities for the events $H^{(c)}$, $H^{(j)}(j = 1, \cdots, k)$, $H^{(RB)}$, and $H^{(RBL)}$.

Similar to (3.13) and (3.14), we obtain

$$P_i^{(PF)}=\alpha_i(1-e^{-\lambda_i T_i})-\sum_{j=1}^{k} P_i^{(j)}-P_i^{(RB)}-P_i^{(ERB)} \tag{5.1}$$

$$P_i^{(PFL)}=(1-\alpha_i)(1-e^{-\lambda_i T_i})-P_i^{(RBL)}-P_i^{(ERBL)}. \tag{5.2}$$

To calculate $P_i^{(ERB)}$ note that in the case of a nonlatent fault, a rollback to an earlier checkpoint succeeds if and only if a rollback to the last checkpoint would have succeeded, and in addition all M instructions between these two rollbacks are executed correctly. Thus,

$$P_i^{(ERB)}=P_i^{(RB,2)} * (P^{(c)})^M \tag{5.3}$$

where $P_i^{(RB,2)}$ is defined in (4.4).

It is clear from (5.3) that for a nonlatent fault, returning to an earlier checkpoint decreases the probability of recovery and is only a waste of time. However, it increases the chances of recovery for a latent fault that occurred shortly before the last checkpoint. To calculate $P_i^{(ERBL)}$ we define for given values of m and l the conditional probability

$$\begin{aligned}P_{i,m,l}^{(ERBL)}=P(H^{(ERBL)}/&\text{type } i \text{ instruction}\\ &\cap\ m \text{ instructions between the last checkpoint and the failing instruction}\\ &\cap\ \text{latency period of } l).\end{aligned}$$

We have to distinguish between faults occurring before the last checkpoint (for which $M - m < l \le 2M - m$) and those occurring after the last checkpoint (for which $1 \le l \le M - m$). For the first case, we have

$$P_{i,m,l}^{(ERBL)}=P_{i,m,l}^{(RBL,2)} * (P^{(c)})^M \tag{5.4}$$

where $P_{i,m,l}^{(RBL,2)}$ is defined in (4.5).

For the second case, we obtain

$$\begin{aligned}P_{i,m,l}^{(ERBL)}=&(1-\alpha_i)\bar{P}_{00}(\lambda_i^{(1)}+\lambda_i^{(2)},\mu_i,\ T_i,\ T_i\\ &+l\bar{T}+k(\bar{T}+\delta_1)+\delta_2+(l+m-M)\bar{T})\\ &* P_0(\lambda_i^{(3)},\ T_i+l\bar{T}+k(\bar{T}+\delta_1)\\ &+\delta_2+(l+m-M)\bar{T}) * P_0(\lambda_i,\ \delta_2)(P^{(c)})^m.\end{aligned} \tag{5.5}$$

Note that values of l greater than $2M - m$ need not be considered since in this case even a rollback to the earlier checkpoint will not succeed.

Averaging $P_{i,m,l}^{(ERBL)}$ over m and l results in

$$\begin{aligned}P_i^{(ERBL)}=&P_i^{(RBL,2)} * (P^{(c)})^M\\ &+\sum_{m=1}^{M}\sum_{l=M-m+1}^{2M-m}\frac{1}{M}\bar{\alpha}(1-\bar{\alpha})^{l-1} * (1-\alpha_i)\\ &* \bar{P}_{00}(\lambda_i^{(1)}+\lambda_i^{(2)},\ \mu_i,\ T_i, T_i+l\bar{T}+k(\bar{T}+\delta_1)\\ &+\delta_2+(l+m-M)\bar{T}) * P_0(\lambda_i^{(3)},\ T_i+l\bar{T}\\ &+k(\bar{T}+\delta_1)+\delta_2+(l+m-M)\bar{T})\\ &*P_0(\lambda_i,\ \delta_2)(P^{(c)})^m\end{aligned} \tag{5.6}$$

where $P_i^{(RBL,2)}$ is defined in (4.6).

The only difference in the cost function between this procedure and the previous one is that the recovery periods associated with the events $H^{(ERB)}$, $H^{(ERBL)}$, $H^{(PF)}$ and $H^{(PFL)}$ are increased by $M\bar{T}$ as compared to the respective quantities in Section IV.

Applying the same method as in the previous two sections, we derive the following expression for $\bar{w}$:

$$\begin{aligned}\bar{w}=&\frac{1}{1-\frac{L+1}{2}(P^{(PF)}+P^{(PFL)})}\\ &\cdot\left\{\bar{T}+\sum_{j=1}^{k} P^{(j)}[j(\bar{T}^{(j)}+\delta_1)]\right.\\ &+P^{(RB)}\cdot\left[k(\bar{T}^{(RB)}+\delta_1)+\delta_2+\frac{M+1}{2}\bar{T}\right]\end{aligned}$$

$$+P^{(ERB)}\left[k(\bar{T}^{(ERB)}+\delta_1)+2\left(\delta_2+\frac{M+1}{2}\bar{T}\right)+M\bar{T}\right]$$
$$+P^{(PF)}\left[k(\bar{T}^{(PF)}+\delta_1)+2\left(\delta_2+\frac{M+1}{2}\bar{T}\right)+M\bar{T}+\delta_3\right]$$
$$+P^{(RBL)}\left[k(\bar{T}+\delta_1)+\frac{\bar{T}}{\bar{\alpha}}+\delta_2+\frac{M+1}{2}\bar{T}\right]$$
$$+P^{(ERBL)}\left[k(\bar{T}+\delta_1)+\frac{\bar{T}}{\bar{\alpha}}+2\left(\delta_2+\frac{M+1}{2}\bar{T}\right)+M\bar{T}\right]$$
$$+P^{(PFL)}\left[k(\bar{T}+\delta_1)+\frac{\bar{T}}{\bar{\alpha}}\right.$$
$$\left.+2\left(\delta_2+\frac{M+1}{2}\bar{T}\right)+M\bar{T}+\delta_3\right]+\left.\frac{T_s}{M}\right\}. \qquad (5.7)$$

A numerical search can now be performed to yield the optimal values of k and M that minimize $\bar{w}$.

VI. Numerical Results

For the numerical comparison of the three recovery procedures, we consider the following computer system. Let the instruction set of the computer be partitioned into three subsets. The first subset consists of relatively simple instructions with execution time T_1 and failure rate λ_1. The instructions in the second subset have an execution time of $2T_1$ and a failure rate of $2\lambda_1$. The third subset consists of complex instructions with execution time $4T_1$ and failure rate $3\lambda_1$. The percentages of intermittent faults not recoverable by instruction retry for the three instruction subsets are 10, 30, and 20 percent, respectively. For convenience, we choose the execution time T_1 as our basic time unit, thus all time periods are measured with respect to this time unit. If for a given instruction mix, the relative frequencies are $f_1 = 0.5$, $f_2 = 0.3$ and $f_3 = 0.2$, then the average net execution time of an instruction is $\bar{T} = \Sigma_{i=1}^{3} f_i T_i = 1.9\ T_1$.

The other parameters, namely, λ_1, s, α_i, δ_1, δ_2, δ_3, T_s have been changed over a large number of different values. For each set of parameter values we have compared the optimal values of $\bar{w}$ corresponding to the three procedures, thus selecting the best procedure to be employed in the given case. Figs. 1(a), (b), and (c) depict three special cases for which the procedures $(k, 1, 0)$, $(k, 2, 0)$, and $(k, 1, 1)$, are optimal, respectively.

Case a [Fig. 1(a)] is characterized by a high probability of permanent faults and a low percentage of latent faults. If most faults are not latent, a rollback to an earlier checkpoint is unlikely to result in a system recovery. Similarly, if most faults are likely to be permanent, a second rollback to the last checkpoint might be just a waste of time. Consequently, a single rollback to the last checkpoint is sufficient and the $(k, 1, 0)$ procedure is the optimal one.

In case b [Fig. 1(b)] the values of s and α_i are lower than the corresponding ones in case a. The value of δ_2 is lower and the value of δ_3 is higher. These changes make the second rollback worthwhile and the $(k, 2, 0)$ procedure is optimal.

Case c [Fig. 1(c)] is characterized by an extremely high percentage of latent faults resulting in long latency periods. In addition, it is assumed that the number of permanent faults is small. In this case we can profit from a second rollback to an earlier checkpoint, hence the $(k, 1, 1)$ procedure should be preferred over the other two.

The best recovery procedure for a system is determined by the entire set of system parameters, yet some of these parameters have a greater impact than the others. Numerical calculations have shown that the selection of the optimal procedure is not very sensitive to changes in λ_i, μ_i, τ_i, or δ_1. It is more sensitive to changes in δ_2, δ_3, and T_s; but the parameters most affecting the choice of a recovery procedure are s, the percentage of permanent faults, and α_i, the percentage of nonlatent faults.

The dependence of the optimal procedure on α_i for a given set of system parameters, is depicted in Fig. 2. For simplicity of presentation we assume that all α_i's are equal. In the graph, the interval $0 \le \alpha_i \le 1$ is divided into three optimality regions. Small values of α_i, indicating a high percentage of latent faults result in the optimality of the $(k, 1, 1)$ procedure. As α_i increases, the $(k, 2, 0)$ procedure becomes optimal since a rollback to an earlier checkpoint is no longer useful. For very large values of α_i, the $(k, 1, 0)$ procedure becomes optimal.

Similar dependence on α_i is obtained for other sets of system parameters, although in some cases any one of the last two regions (corresponding to the $(k, 2, 0)$ and the $(k, 1, 0)$ procedures) may be missing. The first region will always exist since for values of α_i near 0, the $(k, 1, 1)$ procedure is always the best, regardless of the other parameters' values.

The dependence of the optimal procedure on s can be illustrated in the same way. Similarly to Fig. 2, the interval $0 \le s \le 1$ is divided into three regions with the $(k, 1, 1)$ procedure optimal in the first region, the $(k, 2, 0)$ in the second, and the $(k, 1, 0)$ in the third. Any of the first two regions may be missing but the third one always exists.

VII. Conclusions

A general recovery procedure consisting of instruction retries and program rollbacks to the last or the earlier checkpoint, has been presented in this paper. Due to the complexity of the general procedure only three special cases of it have been completely analyzed, namely, the $(k, 1, 0)$, the $(k, 2, 0)$, and the $(k, 1, 1)$ procedures. Exact formulas for the probabilities of all possible events have been derived enabling the calculation of the average time spent per instruction. The latter can then be minimized with respect to the recovery procedure parameters M (the number of instructions between two consecutive checkpoints) and k (the number of instruction retries).

Finally, the three procedures have been numerically compared illustrating the existence of regions of system parameters for which each one of these procedures is superior to the other two.

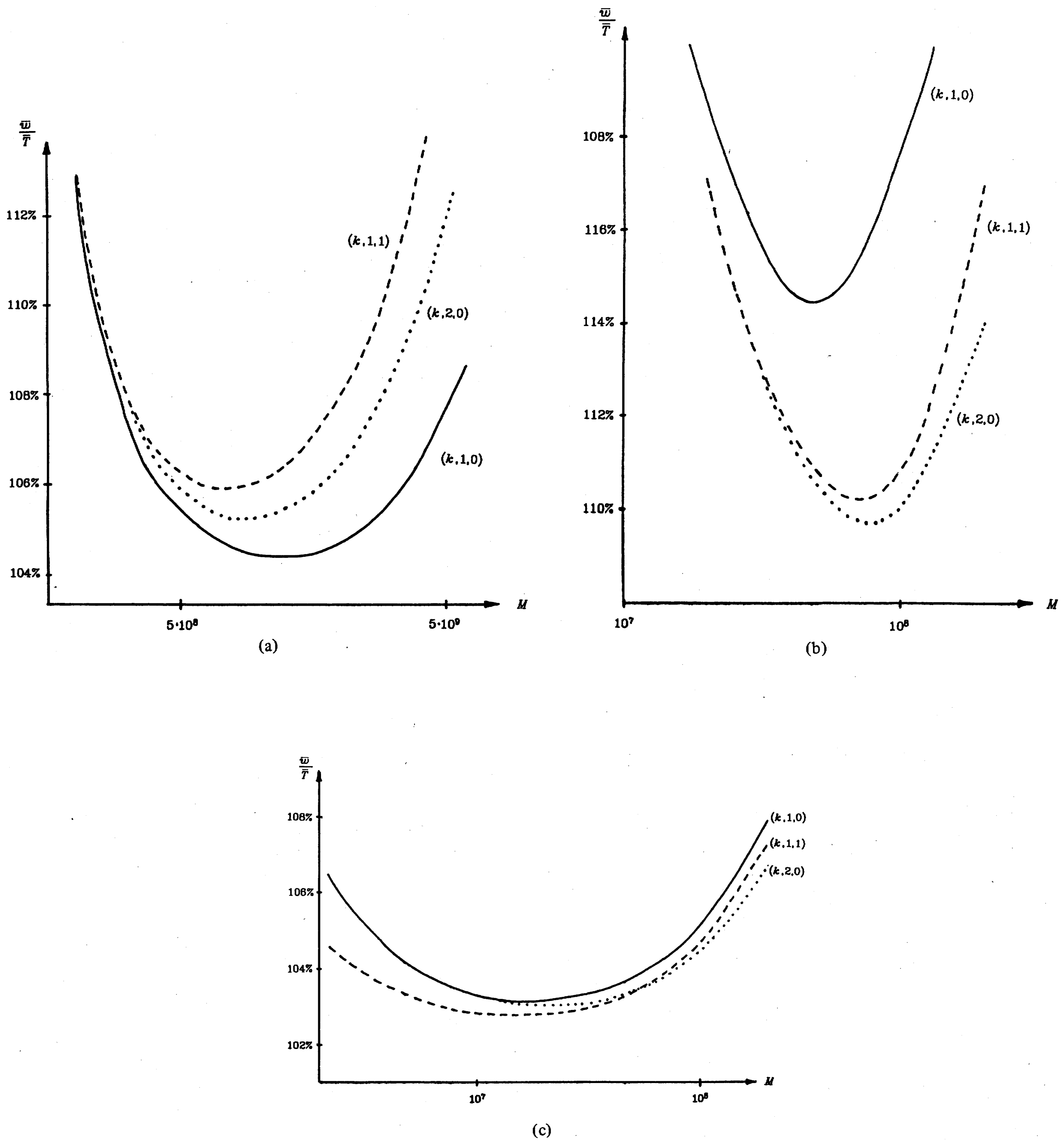


Fig. 1. (a) Percentage of time spent per instruction for the three recovery procedures as a function of the checkpoint interval in case a: $\lambda_i T_1 = 10^{-11}$, $\mu_i T_1 = 0.1$, $\alpha_i = 0.99$, $(i = 1, 2, 3)$, $\delta_1 = 10$, $\delta_2 = 2.5 \cdot 10^7$, $\delta_3 = 10^9$, $T_s = 3 \cdot 10^7$, $s = 0.4$. (b) Percentage of time spent per instruction for the three recovery procedures as a function of the checkpoint interval in case b: $\lambda_i T_1 = 1.8 \cdot 10^{-10}$, $\mu_i T_1 = 0.1$, $\alpha_i = 0.01$, $(i = 1, 2, 3)$, $\delta_1 = 10$, $\delta_2 = 10^5$, $\delta_3 = 5 \cdot 10^9$, $T_s = 5 \cdot 10^6$, $s = 0.005$. (c) Percentage of time spent per instruction for the three recovery procedures as a function of the checkpoint interval in case c: $\lambda_i T_1 = 10^{-10}$, $\mu_i T_1 = 0.1$, $\alpha_i = 10^{-5}$, $(i = 1, 2, 3)$, $\delta_1 = 10$, $\delta_2 = 10^5$, $\delta_3 = 10^9$, $T_s = 10^5$, $s = 0.05$.

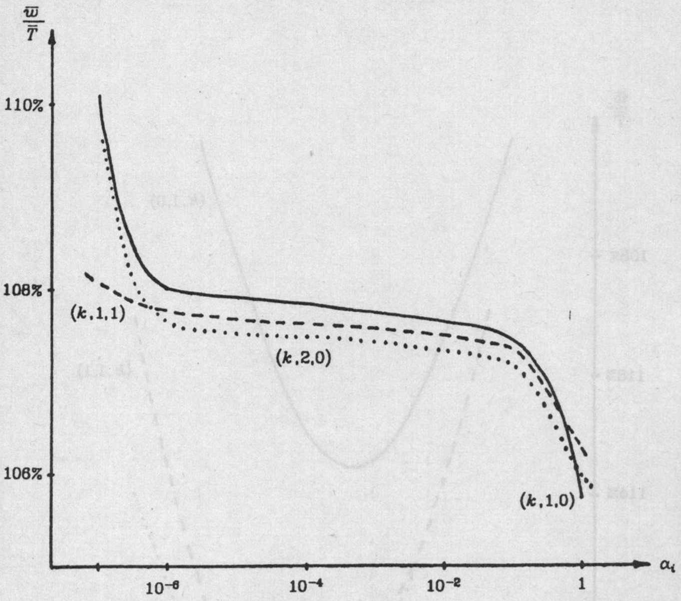


Fig. 2. The dependence of the optimal procedure on α_i for the following case: $\lambda_i T_1 = 5 \cdot 10^{-11}$, $\mu_i T_1 = 0.1$, $(i = 1, 2, 3)$, $\delta_1 = 10$, $\delta_2 = 10^8$, $\delta_3 = 2.5 \cdot 10^9$, $T_s = 5 \cdot 10^8$, $s = 0.1$.

References

[1] K. M. Chandy, J. C. Browne, C. W. Dissly, and W. R. Uhrig, "Analytic models for rollback and recovery strategies in database systems," *IEEE Trans. Software Eng.*, vol. SE-1, pp. 100–110, Mar. 1975.

[2] D. Droulette, "Recovery through programming system/360-system/370," in *Proc. Spring Joint Comput. Conf.*, 1971, pp. 467–476.

[3] E. Gelenbe, "On the optimum checkpoint interval," *J. Ass. Comput. Mach.*, vol. 26, pp. 259–270, Apr. 1979.

[4] I. Koren and S. Y. H. Su, "Reliability analysis of N-modular systems with intermittent and permanent faults," *IEEE Trans. Comput.*, vol. C-28, pp. 514–520, July 1979.

[5] G. H. Maestri, "The retryable processor," in *Proc. FJCC*, 1972, pp. 273–277.

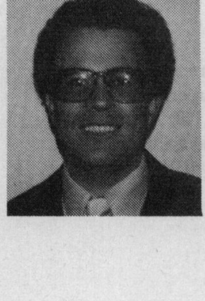
[6] J. J. Shedletsky and E. J. McCluskey, "The error latency of a fault in a combinational digital circuit," in *Dig. 5th Int. Symp. Fault-Tolerant Comput.*, June 1975, pp. 210–214.

[7] S. Y. H. Su, I. Koren, and Y. K. Malaiya, "A continuous parameter markov model and detection procedures for intermittent faults," *IEEE Trans. Comput.*, vol. C-27, pp. 567–570, June 1978.

[8] A. N. Tantawi and M. Ruschitzka, "Performance analysis of checkpointing," *Ass. Comput. Mach., Trans. Comput. Syst.*, vol. 2, pp. 123–144, May 1984.

[9] S. Toueg and O. Babaoglu, "On the optimum checkpoint selection problem," *SIAM J. Comput.*, vol. 13, pp. 630–649, Aug. 1984.

[10] J. W. Young, "A first order approximation to the optimum checkpoint interval," *Commun. Ass. Comput. Mach.*, vol. 17, pp. 530–531, Sept. 1974.



Israel Koren (S'72–M'76) was born on June 23, 1945. He received the B.Sc. (Cum Laude), M.Sc., and D.Sc. degrees from the Technion—Israel Institute of Technology, Haifa, Israel, in 1967, 1970, and 1975, respectively, all in electrical engineering.

From 1968 to 1971 he was with the Computer Center, Israeli Ministry of Defense. In 1972 he joined the Department of Electrical Engineering, Technion—Israel Institute of Technology, Haifa, where he became a Lecturer in 1975. From 1976 to 1978 he was an Assistant Professor in the Department of Electrical Engineering and Computer Science, University of California, Santa Barbara. In 1978 he was an Assistant Professor in the Department of Electrical Engineering - Systems, University of Southern California, Los Angeles. Since 1979 he has been with the Department of Electrical Engineering at the Technion—Israel Institute of Technology. In 1982 he was with the Computer Science Division, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, on leave from Technion—Israel Institute of Technology. Since 1985 he has been the head of the VLSI Systems Research Center at the Technion. His current research interests are fault-tolerant VLSI and WSI architectures, models for yield and performance, reliability evaluation of computer systems, and computer arithmetic.

Zahava Koren was born on September 12, 1946. She received the B.A. and M.A. degrees in mathematics and statistics from the Hebrew University, Jerusalem, Israel, in 1967 and 1969, respectively, and the D.Sc. degree in operations research from the Technion—Israel Institute of Technology, Haifa, Israel in 1976.

From 1967 to 1968 she was a Teaching Assistant in the Department of Statistics, Hebrew University, Jerusalem, Israel. From 1969 to 1974 she was a Teaching Instructor in the Department of Industrial Engineering, the Technion—Israel Institute of Technology, Haifa, and from 1975 to 1976 a Teaching Instructor in the Department of Statistics, University of Haifa, Israel, lecturing on statistical inference, queuing theory, inventory theory, and stochastic processes. From 1972 to 1976 she was a Consulting Statistician in medical and psychological experiments performed at the medical school of Tel-Aviv University, Tel-Aviv, Israel. In 1979 she was an Assistant Professor in the Department of Business and Economics, California State University, Los Angeles, lecturing on statistics and operations research. From 1980 to 1984 she was a Lecturer in the Department of Statistics, University of Haifa, Israel, and a Consultant to SHAHAF, a public opinion research institute. Since 1985 she has been with the Department of Computer Science, the Technion—Israel Institute of Technology, Haifa. Her main interests are optimization in queuing processes, reliability, and communication networks.

Stephen Y. H. Su (S'65–M'67–SM'77) received the BSEE degree from National Taiwan University, Tai Pei, in 1960, and the M.S. and Ph.D. degrees from the University of Wisconsin, Madison, in computer engineering in 1963 and 1967, respectively.

He is currently a Professor in the Computer Science Department and the Head of the Research Group on Design Automation and Fault-Tolerant Computing at the State University of New York, Binghamton. He has done extensive work in the areas of fault-tolerant computer architecture, diagnosis and reliable design of digital systems, logic/system design automation, as well as computer hardware description languages and their applications. He has published over 80 papers in these areas. His interests include fault-tolerant design, computer-aided logic/system design, testing and simulation, computer architecture, and software engineering. From 1974 to 1977, Dr. Su was the Associate Editor of the IEEE TRANSACTIONS ON COMPUTERS in charge of fault-tolerant design, design automation, logic design, and combinatorial theory. He was the Guest Editor for *Computer*'s Special Issue on Hardware Description Language Applications. He was the Chairman of the 1976 International Symposium on Multiple-Valued Logic and the 1975 International Symposium on Computer Hardware Description Languages and Their Applications. He has been an invited speaker at computer seminars at over 80 institutions world-wide.

Dr. Su received the 1981 Engineer of the Year Award. He was included in *Who's Who in Technology, American Men and Women of Science*, in 1982, and *Who's Who in Computer Science Among University Professors* in 1984–1985. He is an Advisor for the IEEE Distinguished Visitor Program. He is a member of Eta Kappa Nu, Tau Beta Pi, and Sigma Xi.