Roll-Forward and Rollback Recovery: Performance-Reliability Trade-Off *

Dhiraj K. Pradhan Nitin H. Vaidya
Department of Computer Science
Texas A&M University
College Station, TX 77843-3112
{pradhan,vaidya}@cs.tamu.edu

Abstract

Performance and reliability achieved by a modular redundant system depend on the recovery scheme used. Typically, gain in performance using comparable resources results in reduced reliability. Several high-performance computers are noted for small mean time to failure. Performance *is measured here in terms of mean and variance of the task completion time,* reliability *being a task-based measure defined as the probability that a task is completed correctly.*

Two roll-forward schemes are compared with two rollback schemes for achieving recovery in duplex systems. The roll-forward schemes discussed here are based on a roll-forward checkpointing concept proposed in [5-8]. Roll-forward recovery schemes achieve significantly better performance than rollback schemes by avoiding rollback in most common fault scenarios. It is shown that the roll-forward schemes improve performance with only a small loss in reliability as compared to rollback schemes.

1. Introduction

Performance and reliability achieved by a modular redundant system depend on the recovery scheme used. Different recovery schemes achieve a different combination of performance and reliability. Given comparable resources, typically gain in performance is achieved with a sacrifice in reliability. In this paper, *performance* is measured in terms of mean and variance of the task completion time and *reliability* is a task-based measure defined as the probability that a task is completed correctly.

This paper compares two roll-forward schemes, based on our earlier work [5-8], with two roll-back schemes for achieving recovery in duplex systems. It is shown that the roll-forward schemes achieve better performance compared to the rollback schemes, with only a marginal loss in reliability.

Figure 1 illustrates an organization that can implement the roll-forward and rollback schemes. Each processing module (PM) is assumed to consist of a processor and a volatile storage (VS). It is assumed that each PM can access a stable storage (SS) which is also readable by the other modules. A reliable *Checkpoint Processor* is assumed available which detects module failures by comparing the state of each pair of processing modules (PMs) that perform the same task. Each pair of processing modules executing an identical task forms a duplex system. Besides the processing modules executing duplicated tasks, it is assumed that a small number of non-dedicated spare modules are available to be utilized for performing diagnosis and recovery when a duplex system experiences a failure.

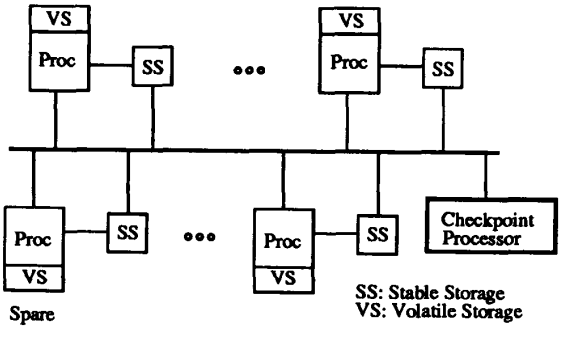


Figure 1: System architecture [6]

Whenever a task checkpoints its state in the stable storage, the state is sent to the Checkpoint Processor. When the Checkpoint Processor receives a state from both of the modules executing a task, it compares the two states. In an implementation, the comparison may be performed using checkpoint signatures. If the two states match, the new checkpoint is considered correct and the old checkpoint discarded. If a mismatch occurs, recovery is initiated.

The initial discussion below makes an implicit assumption that two faulty modules will always produce different checkpoints, which in fact is *not* always valid. However, this assumption primarily affects system reliability and not performance. This issue is therefore considered only when evaluating system reliability.

*Research reported is supported in part by ONR.

0363-8928/94 $3.00 © 1994 IEEE

2. Preliminaries

It is assumed that a processing module may have built-in error detection capability such as parity and other checks. This built-in detection mechanism allows a module to detect its own failure with probability c (coverage). Our earlier work [5-7] as well as related work in [3] and [1] assumed coverage c as 0. It may be noted that most real world systems have $c > 0$.

A spare is ***shared*** by multiple duplex systems to perform "concurrent retry". It is important to note that the spare is not dedicated, therefore, the spare and a duplex system may not be used to form a TMR system. Though the spare is shared by multiple duplex systems, the likelihood is small that the spare will be needed by multiple duplex systems at the same time [6, 8]. Hence, for the discussion here, it is assumed that the spare is available to any duplex system when necessary. The two processing modules in the duplex system under consideration are termed A and B. The spare module is termed S.

The state of the PMs at each checkpoint is saved on stable storage under program control. Checkpointing under program control enables two replicas of a task executed on two PMs to checkpoint at the same points during their execution. The checkpoints are assumed to be equidistant, i.e., the time duration between two consecutive checkpoints is fixed. ([4] presents a technique for inserting approximately equidistant checkpoints.)

The checkpoint intervals are denoted as $I_1, I_2, \cdots, I_j, I_{j+1}, \cdots, I_n$. The checkpoint of processing module Q at the end of interval I_k is termed CP_{kQ}. A module failure is said to be a ***self-detected*** failure if it is detected by the error detection mechanism within the module; the failure is said to be a ***self-undetected*** failure otherwise. A self-undetected failure may be detected through checkpoint comparison at the end of the checkpoint interval. In the diagrams illustrating various fault scenarios, a ***box notation***, shown in Figure 2, is used. The different operations listed in Figure 2 are described later when they are used.

Length of the computation between two consecutive checkpoints is denoted by t_u. The time taken for checkpointing is denoted by t_{ch}. The time required for a rollback (i.e. the time required to make state of the two modules consistent with a previous checkpoint) is t_r. The time required for initiating a restart is t_s. The time required for making the state of the modules in the duplex consistent with the state saved by one of the modules is t_{cp}.

Rollback schemes: When no failure occurs in a checkpoint interval, no rollback is necessary. When a single self-undetected failure occurs, checkpoint comparison only detects the fault. Without the knowledge of the faulty module, only option is to rollback both modules to the previous checkpoint. However, when $c > 0$, some of the failures are self-detected. Figures 3(a) and 3(b) depict a scenario where a self-detected failure occurs in module B during checkpoint interval I_j, while A does not have a self-detected failure. Two actions are possible when such a failure occurs: (i) As shown in Figure 3(c), one may assume that module A is fault-free and make the state of B consistent with the state of module A. In this case, no rollback is required. However, this scheme results in an erroneous outcome if module A had a self-undetected failure in interval I_j. The scheme taking this action is termed ROLLBACK-I scheme. (ii) As shown in Figure 3(b), the system may rollback to the previous checkpoint. The scheme that takes this action is termed ROLLBACK-II scheme.

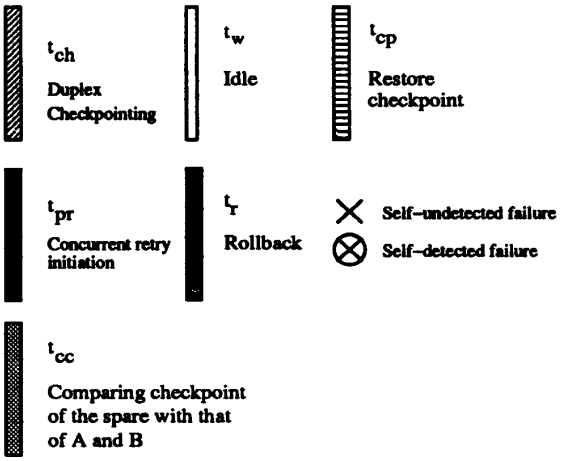


Figure 2: Box notation

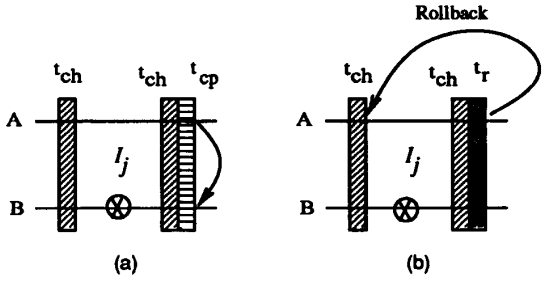


Figure 3: Rollback in duplex systems

3. Roll-Forward Checkpointing Scheme I

This section discusses the first of the two roll-forward checkpointing schemes, named RFCS-I. (The second scheme, RFCS-II, which achieves higher reliability than RFCS-I is discussed in Section 4.) Depending on how the failures occur, there are seven possible types of fault situations in RFCS-I. If failure occurs during the last two checkpoint intervals (I_{n-1} and I_n), ROLLBACK-I is used for recovery rather than

RFCS-I, RFCS-I does not improve performance in these cases.

Let t_0 denote the beginning of a checkpoint interval denoted as I_j. Let the previous interval completed at t_0 be denoted as I_{j-1}. $CP_{(j-1)A}$ and $CP_{(j-1)B}$, checkpoints of A and B at the end of I_{j-1}, are assumed to be identical. It is further assumed that the spare is not permanently faulty. The seven possible fault situations are denoted as (A) through (G). For brevity, only (B) and (D) are discussed in detail, others are briefly summarized. (The reader is referred to [8] for further details.)

The discussion here assumes that the failures may be detected only at the end of each checkpoint interval. The recovery schemes can be easily modified when self-detected failures are detected before the end of a checkpoint interval.

(B) Single self-detected failure: There is no rollback in this situation. This situation occurs when a single module has a self-detected failure in interval I_j. For example, Figure 3(a) illustrates a situation where module B has a self-detected failure in interval I_j. In this case, the state of module B is made consistent with the state of module A, and the two modules then execute interval I_{j+1}. This situation would result in an unreliable outcome if module A had a self-undetected failure in interval I_j. (RFCS-II handles this fault situation differently.)

(D) Concurrent retry without rollback: This situation occurs when a single module has a self-undetected failure in interval I_j. Furthermore, no other module fails in intervals I_j and I_{j+1}. Without loss of generality, assume that processing module B has a self-undetected failure during interval I_j and modules A and S remain fault-free in intervals I_j and I_{j+1}. This case is illustrated in Figure 4.

When a failure occurs in interval I_j, checkpoints CP_{jA} and CP_{jB} of A and B are not identical, and the failure is detected at time t_1 (see Figure 4). When a failure is detected, checkpoint $CP_{(j-1)A}$ and $CP_{(j-1)B}$ are retained in the stable storage. Also, checkpoints CP_{jA} and CP_{jB} are saved for use during concurrent retry. Concurrent retry is performed to determine which processing module, A or B, failed in I_j and to attempt to mask the failure without rollback. The following steps are carried out.

Step 1: To use spare module S for concurrent retry, make state of S consistent with the state CP_{j-1} of modules A and B. Copy the task's executable code to S. The time required for this step is t_{pr}. Now, the spare module S is ready to perform computation in interval I_j. Concurrently, A and B continue execution of the next interval I_{j+1}.

Step 2: When S completes the computation in interval I_j, its state CP_{jS} is compared with both CP_{jA} and CP_{jB}. CP_{jS} will be identical to CP_{jA} and different from CP_{jB}, as A and S were both assumed

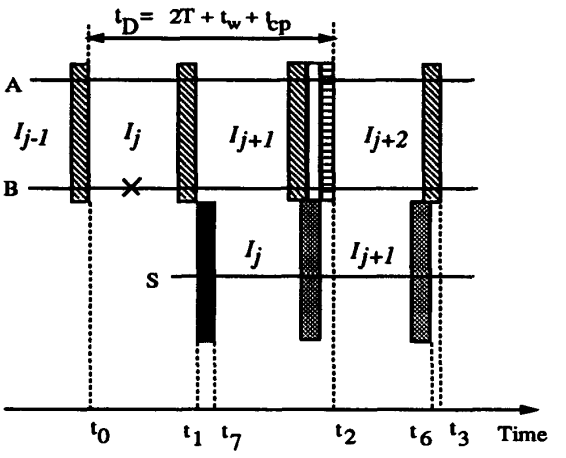


Figure 4: Situation (D) – Concurrent retry without rollback

fault-free in interval I_j. When CP_{jA} and CP_{jS} are found identical, module A is considered fault-free in interval I_j. The time required for this state comparison step is t_{cc}.

While S completes interval I_j, A and B complete interval I_{j+1} and take a checkpoint. Note that A and B were in different states at the beginning of I_{j+1}. A and B wait for state CP_{jS} to be compared with CP_{jA} and CP_{jB}. The length of this wait is termed t_w.[1] Once it is determined that CP_{jA} and CP_{jS} are identical, states of A and B both are made consistent with checkpoint $CP_{(j+1)A}$. The time required for this operation is termed t_{cp}.

Step 3: It is not yet known whether A failed during interval I_{j+1} and whether $CP_{(j+1)A}$ was erroneous or correct. Only CP_{jA} is known to be correct. After completing the state comparison in step 2, spare S executes interval I_{j+1}. Concurrently, modules A and B execute interval I_{j+2}. When S completes I_{j+1}, its state $CP_{(j+1)S}$ is compared with $CP_{(j+1)A}$. As A and S are both assumed fault-free during I_{j+1}, $CP_{(j+1)A}$ and $CP_{(j+1)S}$ will be found identical. $CP_{(j+1)A}$ and $CP_{(j+1)S}$ being identical implies that A was fault-free until the end of interval I_{j+1}. Thus, it is determined that processing modules A and B were in correct state at the start of interval I_{j+2}. With this, the concurrent retry initiated by failure of module B in interval I_j is completed. Any failures in interval I_{j+2} can be treated similar to the failures in interval I_j. Also, the spare is now free to perform any other computation.

As seen above, RFCS-I scheme avoided rollback in spite of a failure of B. The overhead incurred is only $(t_w + t_{cp})$. The traditional rollback scheme requires a much larger overhead, at least $(t_u + t_{ch} + t_r)$.

[1] t_w = maximum$(t_{pr} + t_{cc} - t_{ch}, 0)$.

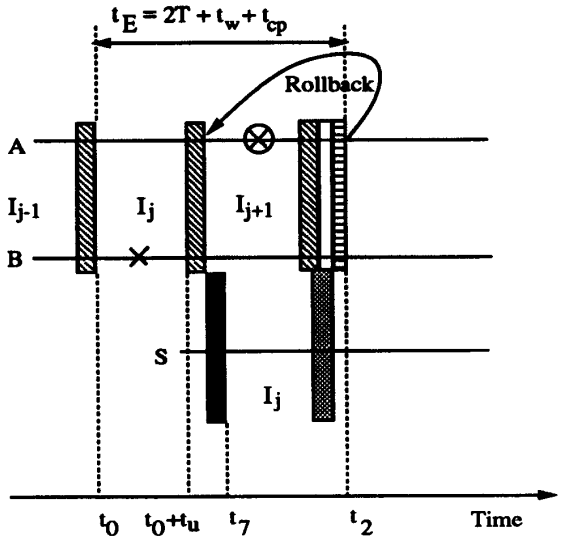


Figure 5: Situation (E)

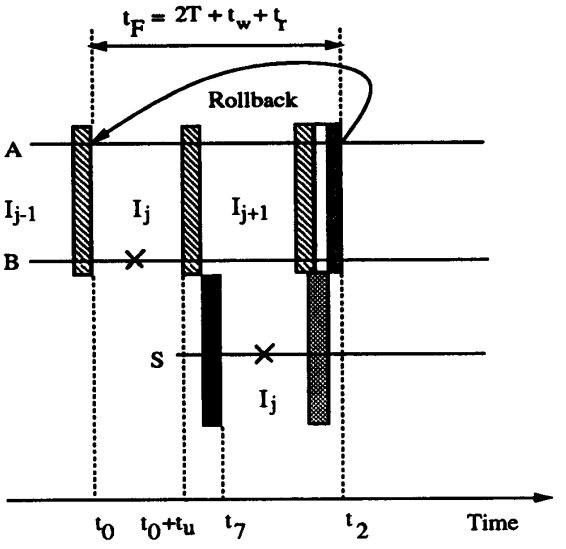


Figure 6: Situation (F)

Other fault situations: In situation (A), neither of the two duplex modules fails in a checkpoint interval. No recovery is, therefore, necessary. In situation (C), both modules have self-detected failures in checkpoint interval I_j; therefore, the modules are rolled back to the previous checkpoint ($CP_{(j-1)A}$).

In situation (E), rollback occurs after the spare has executed one checkpoint interval. This happens when module B(A) has a self-undetected failure during I_j, spare S is fault-free during I_j, and A(B) has a self-detected failure during I_{j+1}. In this case, the system rolls back to CP_{jB} (CP_{jA}), the most recent checkpoint that is identified as correct. Figure 5 illustrates one such scenario where B has a self-undetected failure in I_j and A has a self-detected failure during I_{j+1}.

Situation (F) occurs when either (i) A and B both have self-undetected failures in interval I_j, or (ii) when one of A and B has a self-undetected failure in I_j and the spare also fails in I_j. Figure 6 illustrates one such scenario in which B and S both have self-undetected failures while executing I_j.

In situation (G), module B(A) and spare S are fault-free during I_j and A(B) has a self-undetected failure during I_j. Additionally, either the spare fails in I_{j+1} or B(A) has a self-undetected failure in I_{j+1}. Figure 7 illustrates one such scenario in which B has a self-undetected failure in I_j and the spare fails while executing I_{j+1}.

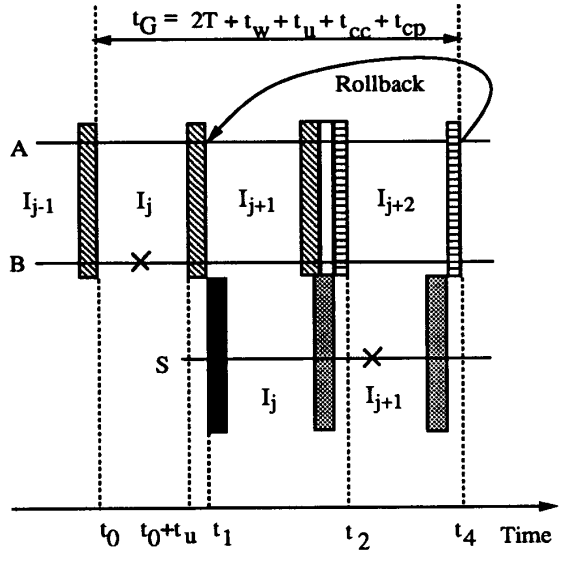


Figure 7: Situation (G)

4. Roll-Forward Checkpointing Scheme II

In RFCS-I scheme presented in Section 3, the task is not completed reliably if in a checkpoint interval both the modules fail and exactly one of them detects its own failure. The cause for this unreliability is Situation (B) of RFCS-I. To overcome this cause of unreliability, another scheme (RFCS-II) is presented below that is a variant of RFCS-I. RFCS-II achieves higher reliability compared to RFCS-I (see Section 6), at the cost of some performance degradation.

The essential difference between RFCS-I and RFCS-II is in the treatment of the fault scenario just described. The treatment of all other fault scenarios in RFCS-II is identical to that in RFCS-I. Specifically, situation (A) and situations (C) through (G) are handled identically in RFCS-I and RFCS-II. Instead of situation (B) in RFCS-I, two situations named (H) and (I) (described below) may occur in RFCS-II. In this scheme, concurrent retry is initiated even when a single self-undetected failure occurs during an interval. If failure occurs during the last checkpoint interval (I_n), then ROLLBACK-II is used for recovery rather than RFCS-II (since, in this case, RFCS-II does not improve performance).

(H) No rollback required: Table below describes the fault scenarios possible in situation (H). Figure 8 illustrates scenario H.1. As illustrated in Figure 8, assume that module B has a self-detected failure in I_j and module A is not faulty in I_j. Also, module S does not fail in I_j during the concurrent retry.

det. ≡ self-detected
undet. ≡ self-undetected
X ≡ don't care

Fault scenarios possible in situation (H)

	Status in interval I_j		
	A	B	S
H.1	fault-free	det. fault	fault-free
H.2	det. fault	fault-free	fault-free

Fault scenarios possible in situation (I)

	Status in interval I_j		
	A	B	S
I.1	fault-free	det. fault	faulty
I.2	det. fault	fault-free	faulty
I.3	undet. fault	det. fault	X
I.4	det. fault	undet. fault	X

After the failure in B is detected, initiate concurrent retry of interval I_j on spare S. Also, make the state of module B consistent with CP_{jA}. While the spare completes interval I_j, module A completes interval I_{j+1}. State CP_{jS} of the spare is then compared with CP_{jA}. As A and S are both fault-free in I_j, the two checkpoints will match. Thus, it is verified that module A did not fail in interval I_j, and the concurrent retry is completed.

(I) Rollback required: The above table describes all the fault scenarios possible in situation (I). The steps described in situation (H) are carried out here also. However, in situation (I), the state comparison results in a mismatch. The system is therefore rolled back to state $CP_{(j-1)A}$ (or $CP_{(j-1)B}$).

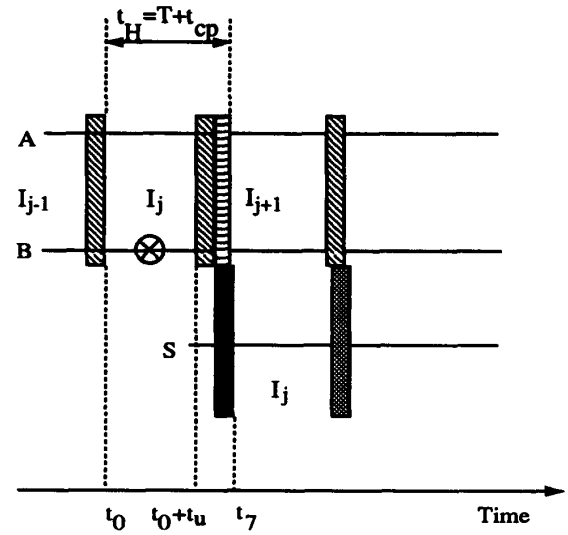


Figure 8: Situation (H)

5. Performance Analysis

The analysis assumes that failures of any two modules are independent. Occurrence of a transient failure of a module is assumed to be a Poisson process with failure rate λ. The notations used here are summarized now. (Some of the notations were introduced in earlier sections.) T_u denotes execution time of a task without failures and checkpointing. n is the number of equidistant checkpoints and t_u denotes T_u/n. $\overline{\tau_n}$ denotes the expected (i.e., mean or average) task completion time. $\overline{\tau_{n|f}}$ denotes the expected completion time of a task given that at least one failure occurred during task execution. v_n is variance of the task completion time. t_{ch} denotes the time required to checkpoint the two modules in a duplex system. t_{ch} includes the time required to compare the two checkpoints. t_r is the time required for a rollback and t_s is the time required for a restart. t_{cp} denotes the time required for rolling back to a previous state of one of the modules. t_{cc} is the time required for comparing state of the spare with checkpoints of the processing modules in a duplex system. t_{pr} is the time required to initiate a concurrent retry. T denotes $t_u + t_{ch}$. t_w denotes idle time equal to maximum($t_{pr}+t_{cc}-t_{ch}$, 0).

For brevity, the details of performance analysis are omitted here [8]. Instead, closed form expressions for RFCS-I scheme are presented. (Similar expressions can be obtained for RFCS-II and rollback schemes, too.) Let p_A through p_I be the likelihood of oc-

currence of situations (A) through (I), respectively, enumerated in Sections 3 and 4. For example, situation (B) occurs when one module has a self-detected failure in an interval and the other module does not have a self-detected failure. The probability of this event is $p_B = 2 * c(1 - e^{-\lambda T}) * (1 - c(1 - e^{-\lambda T}))$. Similarly, the following expressions can be obtained, where $f(t) = e^{-\lambda t}$ and $g(t) = 1 - f(t)$.

$$
\begin{aligned}
p_A &= f(2T) \\
p_B &= 2cg(T)(1 - cg(T)) \\
p_C &= c^2 g^2(T) \\
p_D &= 2(1-c)\ g(T)\ f(T)\ f(T + t_{pr} + 2t_u + 2t_{cc}) \\
p_E &= 2(1-c)\ g(T) f(T)\, cg(T) f(t_{pr} + t_u + t_{cc}) \\
p_F &= (1-c)^2 g^2(T) + \\
&\quad 2(1-c)\ g(T)\ f(T) g(t_{pr} + t_u + t_{cc}) \\
p_G &= 2(1-c)\ g(T)\ f(T)\ f(t_{pr} + t_u + t_{cc}) \\
&\quad \times (g(T + t_u + t_{cc}) - cg(T)) \\
p_H &= 2cg(T) f(T) f(t_{pr} + t_u + t_{cc}) \\
p_I &= 2c(1-c) g^2(T) + 2cg(T) f(T) g(t_{pr} + t_u + t_{cc})
\end{aligned}
$$

As should be expected, $p_H + p_I = p_B$ and $p_A + p_B + p_C + p_D + p_E + p_F + p_G = 1$. Let $p_{roll} = 1 - p_A - p_B - p_C = 2(1-c)\ (1 - e^{-\lambda T})\ e^{-\lambda T} + (1-c)^2(1-e^{-\lambda T})^2$. Let $t_A = T$, $t_B = T + t_{cp}$, $t_C = T + t_r$, $t_D = 2T + t_w + t_{cp}$, $t_E = 2T + t_w + t_{cp}$, $t_F = 2T + t_w + t_r$, $t_G = 2T + t_w + t_u + t_{cc} + t_{cp}$, and $t_{roll} = T + t_r = t_C$.

When $n \leq 2$, the RFCS-I scheme is identical to ROLLBACK-I scheme. For $n \geq 3$, the following expressions for expected completion time $\overline{\tau_n}$ and variance v_n for RFCS-I can be obtained [8].

$$
\overline{\tau_n} = \frac{q_D}{1+q_D} \left(\begin{array}{l} t_m \left((n-2) q_D^{-1} + \frac{1-(-q_D)^{n-2}}{1+q_D} \right) \\ + \overline{\tau_1} \left(q_D^{-1} + (-q_D)^{n-1} \right) \\ + (\overline{\tau_2} - q_{ABEG} \overline{\tau_1}) \left(q_D^{-1} + (-q_D)^{n-2} \right) \end{array} \right)
$$

The expression for v_n is given in Figure 9, where

$$
\begin{aligned}
q_X &= p_X/(1 - p_C - p_F), \text{ for } X = A, \cdots, G, \\
q_{ABEG} &= q_A + q_B + q_E + q_G, \\
t_m &= \textstyle\sum_{X=A,B,C,D,E,F,G}\ q_X t_X, \\
\overline{\tau_1} &= \frac{(p_C + p_{roll}) t_C}{1 - p_C - p_{roll}} + \frac{p_A t_A + p_B t_B}{1 - p_C - p_{roll}}, \\
\overline{\tau_2} &= 2\ \overline{\tau_1}, \\
v_1 &= \frac{(p_C + p_{roll}) t_C^2}{(1 - p_C - p_{roll})^2} + \frac{p_A t_A^2 + p_B t_B^2}{1 - p_C - p_{roll}} - \frac{(p_A t_A + p_B t_B)^2}{(1 - p_C - p_{roll})^2}, \\
v_2 &= 2\ v_1, \\
S_1 &= v_1 + (\overline{\tau_1})^2, \\
S_2 &= v_2 + (\overline{\tau_2})^2, \text{ and} \\
s_m &= \textstyle\sum_{X=A,B,C,D,E,F,G}\ q_X t_X^2.
\end{aligned}
$$

Recall that $\overline{\tau_{n|f}}$ is the average task completion time, given that at least one failure occurred during task execution. Using the expression for $\overline{\tau_n}$, $\overline{\tau_{n|f}}$ can be obtained.

$$
\overline{\tau_{n|f}} = \frac{\overline{\tau_n} - p_A^n\ n\ T}{1 - p_A^n}. \qquad (1)
$$

Performance of RFCS-I and RFCS-II schemes is compared with ROLLBACK-I and ROLLBACK-II schemes, respectively. Parameters for a hypothetical task, named Task 1, are listed below. Results presented here for Task 1 are also valid over a range of task parameters. For brevity, we have chosen only one set of parameter values.

T_u	t_{ch}	t_r	t_s	t_{cc}	t_{cp}	t_{pr}
50	0.50	0.30	0.30	0.70	0.30	0.40

Note that when coverage c is 0, ROLLBACK-I and ROLLBACK-II schemes become identical. Also, RFCS-I and RFCS-II schemes become identical.

Comparison of $\overline{\tau_{n|f}}$

$\overline{\tau_{n|f}}$ is the expected task completion time, given that at least one failure occurs during the execution of the task. In the absence of failures, RFCS and ROLLBACK schemes perform identically; $\overline{\tau_{n|f}}$ is a good measure of how a scheme performs when failures occur. The minimum possible task completion time given n checkpoints is $n(t_u + t_{ch})$. Numerical results indicated that for the RFCS schemes, $\overline{\tau_{n|f}}$ is closer to $n(t_u + t_{ch})$ compared to the ROLLBACK schemes. Essentially, this is because of the fact that the RFCS schemes avoid rollback in the presence of a single failure, and therefore complete the task in about the same time as a failure-free execution. Let g denote the "relative gain" in $\overline{\tau_{n|f}}$ achieved by an RFCS scheme with respect to the corresponding ROLLBACK scheme. Define

$$
g(\textbf{RFCS-I}) = \frac{\overline{\tau_{n|f}}(\textbf{ROLLBACK-I}) - \overline{\tau_{n|f}}(\textbf{RFCS-I})}{(T_u/n)}.
$$

g(RFCS-II) is defined similarly. In Table 1, relative gains of the two RFCS schemes are listed for various values of n. When c is large, the relative gain of RFCS-I seems to decrease; however, the relative gain of RFCS-II remains high. $\overline{\tau_{n|f}}$ is defined as the expected task completion time conditional to a failure occurring during task execution, and not conditional to *undetected* failures. When c is large, most faults tend to be detected. As ROLLBACK-I and RFCS-I both treat self-detected faults identically, the average performance of RFCS-I approaches that of ROLLBACK-I. Therefore, the relative gain decreases as c approaches 1. Observe that the performance of RFCS schemes remains better over a wide range of failure rate λ. Table 1 lists the relative gain for $\lambda = 10^{-3}, 10^{-6}, 10^{-12}$. However, to minimize the number of graphs the following assumes $\lambda = 10^{-3}$.

Mean and variance comparison

In Figure 10, variance v_n is plotted versus the mean completion time $\overline{\tau_n}$ for the example task. Each

$$v_n = \frac{q_D}{1+q_D}\left(\begin{array}{l} 2\,(q_C\,t_C + q_F t_F)\,\sum_{i=3}^{n}\left(\overline{r_i}\,\left[q_D^{-1} + (-q_D)^{n-i}\right]\right) + \\ 2(q_A t_A + q_B t_B + q_E t_E + q_G t_G)\sum_{i=2}^{n-1}\left(\overline{r_i}\left[q_D^{-1} + (-q_D)^{n-1-i}\right]\right) \\ +2\,q_D\,t_D\,\sum_{i=1}^{n-2}\left(\overline{r_i}\,\left[q_D^{-1} + (-q_D)^{n-2-i}\right]\right) \\ +s_m\,\left((n-2)q_D^{-1} + \frac{1-(-q_D)^{n-2}}{1+q_D}\right) \\ +S_1(q_D^{-1} + (-q_D)^{n-1}) + (S_2 - q_{ABEG}S_1)\,(q_D^{-1} + (-q_D)^{n-2}) \end{array}\right) - (\overline{r_n})^2$$

Figure 9: Expression for v_n, $n \geq 3$, for RFCS-I (see Section 5 for details)

Table 1: Relative gain achieved by RFCS schemes

Task 1

$c = 0$					
		n			
RFCS	λ	3	5	8	12
I,II	10^{-3}	.325	.590	.747	.834
I,II	10^{-6}	.331	.594	.738	.813
I,II	10^{-12}	.331	.594	.738	.813
$c = 0.8$					
		n			
RFCS	λ	3	5	8	12
I	10^{-3}	.062	.116	.148	.166
I	10^{-6}	.066	.118	.147	.162
I	10^{-12}	.066	.118	.147	.162
II	10^{-3}	.627	.814	.938	1.02
II	10^{-6}	.615	.790	.903	.984
II	10^{-12}	.615	.790	.903	.984

point on the mean-variance plot corresponds to a specific number of checkpoints. By varying the number of checkpoints, different mean and variance can be achieved. Observe that for any mean and variance pair achieved using a ROLLBACK scheme, a pair with lower mean and variance can be achieved using the corresponding RFCS scheme. For example, in Figure 10, observe that if ROLLBACK-II scheme with $n = 7$ is used, then one may use the RFCS-II scheme with $n = 5$, 6 or 7 and achieve lower mean completion time with lower variance. In general, the RFCS schemes can achieve a lower minimum average task completion time compared to the ROLLBACK schemes. For smaller values of λ, the absolute improvement achieved by RFCS schemes in $\overline{r_n}$ becomes smaller, as failures are less likely and all schemes perform equally well when there are no failures.

In Figure 10 note that the mean completion time is minimized when n is small. When the size of a task is much smaller compared to the mean time to module failure ($1/\lambda$), the mean completion time is minimized by using a small number of checkpoints. However, the variance is not minimized with small n. Additionally, the reliability is also not minimized with small n. In fact, reliability increases monotonically with increasing n. With smaller checkpoint intervals, the probability of multiple failures becomes smaller, increasing the likelihood of reliable outcome.

6. Reliability of RFCS and ROLLBACK Schemes

Our discussion has thus far ignored the possibility that two faulty modules with self-undetected faults may produce the same checkpoint. While the likelihood of this situation may be small, it is instructive to consider this possibility when comparing *reliability* of the RFCS and ROLLBACK schemes presented earlier. *Reliability* of recovery scheme M, denoted as $R(M)$, is defined as the likelihood that a task will complete correctly when recovery scheme M is used.

Our analysis assumes a symmetric error model. (This model may not always hold in practice; however, similar analysis can also be performed for other error models.) Let S_c be the set of all checkpoints a fault-free module may produce. Assume that a faulty module with a self-undetected failure produces each of the $(|S_c| - 1)$ incorrect checkpoints with equal likelihood, $\frac{1}{|S_c|-1}$. Then, the likelihood that two faulty modules with self-undetected failures (in the same duplex system) may produce the same checkpoint can be seen[2] to be $\frac{1}{|S_c|-1}$. Let $R_k(M)$ denote the probability that a task will complete its last k intervals reliably using scheme M. Then, $R_n(M)$ is the same as $R(M)$. Also, let χ denote $1/(|S_c| - 1)$.

Reliability of ROLLBACK-I Scheme:

ROLLBACK-I scheme produces an erroneous output if (a) during a checkpoint interval, both the modules have self-undetected failures and also produce the same checkpoint, or (b) one module has a self-detected failure and the other module has a self-undetected failure. The following recursion is obtained, where $r = p_A + 2c(1 - e^{-\lambda T})e^{-\lambda T}$.

$$R_1(\text{ROLLBACK-I}) = r +$$

[2]There are $(|S_c| - 1)$ incorrect checkpoints, and each module produces each with probability $1/(|S_c| - 1)$. Therefore, the probability that both modules produce the same incorrect checkpoint is $(|S_c| - 1) \times (1/(|S_c| - 1)) \times (1/(|S_c| - 1)) = 1/(|S_c| - 1)$.

$$\begin{pmatrix} 1-r-\chi(1-c)^2(1-e^{-\lambda T})^2 \\ -2c(1-c)(1-e^{-\lambda T})^2 \end{pmatrix} R_1(\text{ROLLBACK-I})$$

$R_n(\text{ROLLBACK-I}) = R_1^n(\text{ROLLBACK-I})$.

Solving the above recursion, we get $R_n(\text{ROLLBACK-I})$ equal to

$$\left(\frac{r}{r+\chi(1-c)^2(1-e^{-\lambda T})^2+2c(1-c)(1-e^{-\lambda T})^2)}\right)^n$$

Reliability of ROLLBACK-II Scheme:
ROLLBACK-II scheme produces an erroneous output only if, during a checkpoint interval, both the modules have self-undetected failures and also produce the same checkpoint. The following recursion is obtained.

$$R_1(\text{ROLLBACK-II}) = p_A + (1-p_A-\chi(1-c)^2(1-e^{-\lambda T})^2)R_1(\text{ROLLBACK-II})$$

$R_n(\text{ROLLBACK-II}) = R_1^n(\text{ROLLBACK-II})$.

Solving the above recursion, we get

$$R_n(\text{ROLLBACK-II}) = \left(\frac{p_A}{p_A+\chi(1-c)^2(1-e^{-\lambda T})^2}\right)^n.$$

Observe that when $0 < c < 1$ and $|S_c| = \infty$, i.e. $\chi = 0$, $R_n(\text{ROLLBACK-II}) = 1$ while $R_n(\text{ROLLBACK-I}) < 1$.

Reliability of RFCS-I Scheme: For $X = A, \cdots, I$, define r_X = Probability that situation X occurs and is completed reliably. Let $f(t) = e^{-\lambda t}$ and $g(t) = 1 - f(t)$. Then,

$$\begin{aligned} r_z &= p_z, \text{ for } z = A, C, D, E \\ r_B &= 2c\,g(T)f(T) \end{aligned}$$

$$\begin{aligned} r_F &= (1-\chi)(1-c)^2g^2(T)f(t_{pr}+t_u+t_{cc}) \\ &+ (1-\chi)(1-c)^2g^2(T)\,c\,g(t_{pr}+t_u+t_{cc}) \\ &+ 2(1-c)g(T)f(T)\,c\,g(t_{pr}+t_u+t_{cc}) \\ &+ 2(1-c)g(T)f(T)\,(1-c)g(t_{pr}+t_u+t_{cc})(1-\chi) \\ &+ (1-c)^3g^2(T)g(t_{pr}+t_u+t_{cc})\tfrac{(|S_c|-2)(|S_c|-3)}{(|S_c|-1)^2} \end{aligned}$$

$$r_G = 2(1-c)\,g(T)\,f(T)\,f(t_{pr}+t_u+t_{cc}) \times \begin{pmatrix} (1-c)g(T)f(t_u+t_{cc}) + f(T)g(t_u+t_{cc}) \\ +(1-c)^2g(T)g(t_u+t_{cc})(1-\chi) \\ +(1-c)g(T)c\,g(t_u+t_{cc}) \end{pmatrix}$$

Similar to the recursions for ROLLBACK schemes, the following recursion for RFCS-I scheme is obtained. Recall that ROLLBACK-I is used when failures occur during I_{n-1} or I_n.

$$\begin{aligned} R_1(\text{RFCS-I}) &= R_1(\text{ROLLBACK-I}) \\ R_2(\text{RFCS-I}) &= R_2(\text{ROLLBACK-I}) \end{aligned}$$

For $k \geq 3$, $R_k(\text{RFCS-I}) = (r_A + r_B + r_E + r_G)R_{k-1}(\text{RFCS-I}) + r_DR_{k-2}(\text{RFCS-I}) + (r_C + r_F)R_k(\text{RFCS-I})$. $R_n(\text{RFCS-I})$ is obtained by solving the recursion.

Reliability of RFCS-II Scheme: The probabilities r_A through r_G were stated above. Similarly, $r_H = p_H$ and

$$\begin{aligned} r_I &= 2c\,g(T)f(T)g(t_{pr}+t_u+t_{cc}) \\ &+2c\,g(T)(1-c)g(T)f(t_{pr}+t_u+t_{cc}) \\ &+2c\,g(T)(1-c)g(T)cg(t_{pr}+t_u+t_{cc}) \\ &+2cg(T)(1-c)g(T)(1-c)g(t_{pr}+t_u+t_{cc})(1-\chi) \end{aligned}$$

Also, define $r_{roll} = 2(1-c)g(T)f(T) + (1-c)^2g^2(T)(1-\chi)$. The following recursion for RFCS-II scheme is obtained. Recall that ROLLBACK-II is used when failures occur during I_n.

$$\begin{aligned} R_1(\text{RFCS-II}) &= R_1(\text{ROLLBACK-II}) \\ R_2(\text{RFCS-II}) &= (r_A+r_H)R_1(\text{RFCS-II}) \\ &\quad +(r_C+r_I)R_2(\text{RFCS-II}) \\ &\quad + r_{roll}R_2(\text{RFCS-II}) \end{aligned}$$

For $k \geq 3$:

$$\begin{aligned} R_k(\text{RFCS-II}) &= (r_A+r_E+r_G+r_H)R_{k-1}(\text{RFCS-II}) \\ &\quad +r_DR_{k-2}(\text{RFCS-II}) + \\ &\quad (r_C+r_F+r_I)R_k(\text{RFCS-II}) \end{aligned}$$

$R_n(\text{RFCS-II})$ is obtained using above recursion.

Reliability Comparison

The following observations can be made from the discussion and analysis presented above.

- When $|S_c| = \infty$ and $0 < c < 1$, $R(\text{ROLLBACK-II}) = R(\text{RFCS-II}) = 1$. Also, $R(\text{ROLLBACK-I}) < 1$ and $R(\text{RFCS-I}) < 1$. To analytically verify that $R(\text{RFCS-II}) = 1$, observe the following: Under the given conditions, $R_1(\text{ROLLBACK-II}) = R_2(\text{ROLLBACK-II}) = 1$ and $p_X = r_X$, for $X = A, C, D, E, F, G, H, I$. The rest follows from the recursion for $R_k(\text{RFCS-II})$.
- As c approaches 1, reliability of all the four schemes approaches 1.
- When $c = 0$, ROLLBACK-I and ROLLBACK-II schemes are identical, also RFCS-I and RFCS-II schemes are identical. Therefore, their reliabilities are also identical.

Figure 11 plots unreliabilities for $\lambda = 10^{-3}$. Similar curves are obtained for smaller values of λ, as well [8]. Note that *unreliability* of scheme M is defined as $1 - R(M)$, i.e. $1 - R_n(M)$. Two extreme values for $|S_c|$ have deliberately been chosen to demonstrate the effect of the variation in $|S_c|$. In practice, $|S_c|$ is expected to be large. The following observations can be made from Figure 11 and other unreliability plots omitted here due to lack of space.

- With $c > 0$, $R(\text{RFCS-II}) > R(\text{RFCS-I})$, i.e., RFCS-II is more reliable than RFCS-I.
- For most values of coverage c, $R(\text{RFCS-I}) \leq R(\text{ROLLBACK-I})$ and $R(\text{RFCS-II}) \leq R(\text{ROLLBACK-II})$.
- For large values of $|S_c|$, $R(\text{RFCS-I})$ is almost equal to $R(\text{ROLLBACK-I})$ while $R(\text{ROLLBACK-II})$ remains marginally better than $R(\text{RFCS-II})$. However, as $|S_c|$ approaches ∞, both $R(\text{ROLLBACK-II})$ and $R(\text{RFCS-II})$ approach 1.
- $R(\text{RFCS-I})$ and $R(\text{ROLLBACK-I})$ actually decrease as the coverage c increases from 0 and approaches 0.5. A similar phenomenon has been observed before with regards to safety in reconfigurable duplication [2]. This phenomenon is due to the fact that with moderately low fault coverage, the likelihood that an unreliable action is taken in situation (B) is high. That is, when one of the modules in the duplex system has a self-detected failure, the likelihood that the failure in the other module is self-undetected remains high (due to low coverage). This may result in the duplex system being restored to an incorrect state.
- For a range of parameters, $R(\text{ROLLBACK-II}) \geq R(\text{RFCS-II}) \geq R(\text{ROLLBACK-I}) \geq R(\text{RFCS-I})$.

One approach for improving the reliability of the RFCS schemes is using more than two modules to perform the task and multiple spares for concurrent retry, rather than just one. For example, one may use three modules to perform the task and require that all the three modules produce an identical checkpoint [6]. This choice of a conservative (safe) strategy for checkpoint comparison results in a higher reliability of the recovery scheme. The RFCS schemes presented in this chapter for duplex systems can be generalized, with reasonable hardware overhead, for all modular redundant systems by using one or more spares for concurrent retry, and two or more modules for normal execution.

7. Summary

This paper has presented two roll-forward checkpointing schemes (RFCS) for duplex systems. These schemes are useful in a multiprocessor environment where multiple duplex systems share a small number of spares to achieve recovery. The RFCS schemes improve performance of duplex systems by avoiding rollback, in most common fault scenarios.

A trade-off exists between performance and reliability achievable, given comparable resources – typically performance gain is achieved with a loss in reliability. Here, performance is measured in terms of the mean and variance of the task completion time, reliability defined as the probability that a task is completed correctly.

Because a small number of spares is shared by multiple duplex systems, the resource requirement of the RFCS schemes is comparable with ROLLBACK schemes. Comparison of the reliability and performance of RFCS checkpointing schemes and ROLLBACK schemes suggests that the RFCS schemes achieve significant improvements in performance by trading small amounts of reliability. Quantitative results indicate that the trade-off achieved by the RFCS schemes remains favorable provided that the likelihood of two faulty modules producing the same incorrect checkpoint is not large.

References

[1] P. Agrawal, "Fault tolerance in multiprocessor systems without dedicated redundancy," *IEEE Trans. Computers*, vol. 37, pp. 358–362, March 1988.

[2] B. W. Johnson, *Design and Analysis of Fault Tolerant Digital Systems.* Addison-Wesley, 1989.

[3] J. Long, W. K. Fuchs, and J. A. Abraham, "Forward recovery using checkpointing in parallel systems," in *Proc. Int. Conf. Parallel Proc.*, pp. 272–275, August 1990.

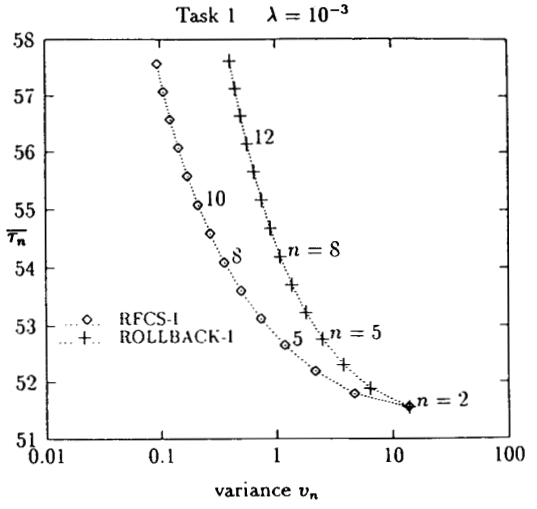
[4] J. Long, W. K. Fuchs, and J. A. Abraham, "Compiler-assisted static checkpoint insertion," in *Digest of papers: The 22nd Int. Symp. Fault-Tolerant Comp.*, pp. 58–65, July 1992.

[5] D. K. Pradhan, "Redundancy schemes for recovery," Tech. Rep. TR-89-CSE-16, ECE Department, Univ. of Massachusetts–Amherst, 1989.

[6] D. K. Pradhan and N. H. Vaidya, "Roll-forward checkpointing scheme: Concurrent retry with nondedicated spares," in *IEEE Workshop on Fault Tolerant Parallel and Distributed Systems*, pp. 166–174, July 1992.

[7] D. K. Pradhan and N. H. Vaidya, "Roll-forward checkpointing scheme: A novel fault-tolerant architecture," to appear in *IEEE Trans. Computers.*

[8] N. H. Vaidya, *Low-Cost Schemes for Fault Tolerance.* PhD thesis, University of Massachusetts-Amherst, February 1993. Available from UMI Dissertation Services, Ann Arbor, Michigan. Order number 9316722.



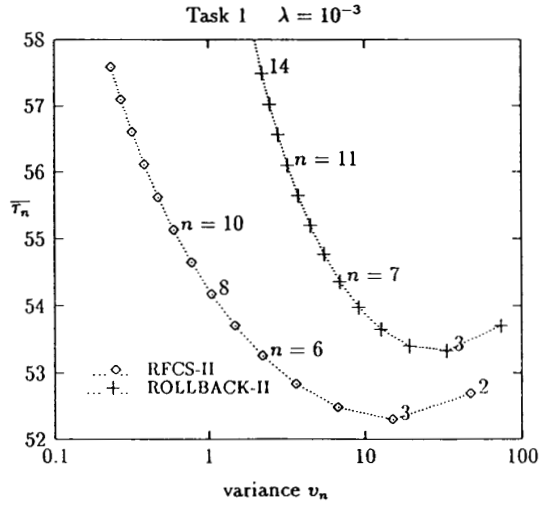
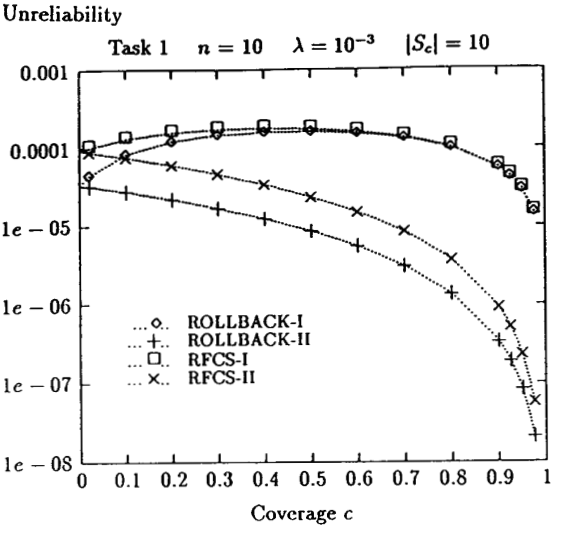


Figure 10: Mean completion time versus variance for task 1 with $\lambda = 10^{-3}$ and $c = 0.8$



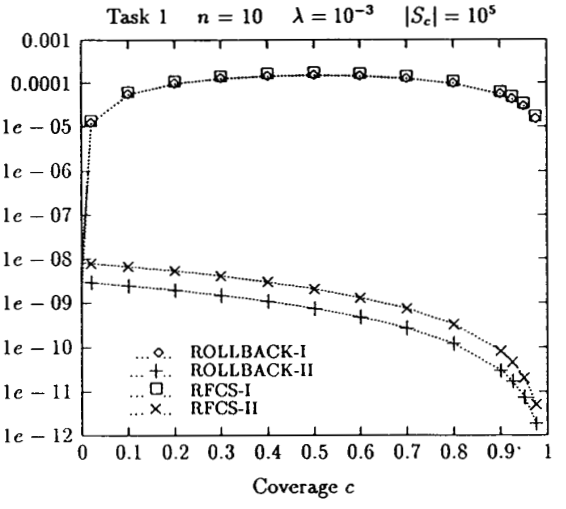


Figure 11: Unreliability versus coverage with $\lambda = 10^{-3}$.