# IMPLEMENTING AN APPROXIMATE PROBABILISTIC ALGORITHM FOR ERROR RECOVERY IN CONCURRENT PROCESSING SYSTEMS

**Silvia Heubach**
Dept. of Mathematics and Computer Science
California State University, Los Angeles

**Raj S. Pamula**
Dept. of Mathematics and Computer Science
California State University, Los Angeles

## ABSTRACT

We have developed a probabilistic algorithm for improved error recovery in a system of concurrent processes. Simulations for various lengths of checkpoint intervals have shown that in most cases the probabilistic method is more cost effective than the iterative rollback method. However, implementation of the probabilistic algorithm requires knowledge of the distribution function of the latency times between error occurrence and error detection. In this paper, we present a method for obtaining an approximate empirical distribution function for the latency times using the iterative rollback method. The cost effectiveness of the probabilistic method, when based on the approximate distribution function, is investigated for various parameters (number of data points collected, length of error interval). We show that using the probabilistic algorithm in conjunction with the approximate distribution function still leads to significant cost reduction over the iterative method, while not requiring knowledge of the theoretical distribution function, making this implementation universally applicable.

**Keywords:** Checkpoint, Rollback and Error Recovery, Error Latency, Acceptance Test, Concurrent Processing, Implementation, Probabilistic Algorithm.

## INTRODUCTION

A design of a reliable system requires a careful study of errors, causes of errors, and system response to overcome errors. Transient errors pose a difficult problem as all traces of their nature are long gone by the time the error is detected. A popular means of detecting errors is the use of acceptance tests (ATs) [1, 2]. After a failure has been detected, recovery is commonly attempted by rolling back to a previously established checkpoint [3-9]. Most of the research in the design of a reliable system to overcome transient failures is focussed on data collection, modeling and simulation [3, 6, 10].

In a concurrent processing system, inter-process communications can lead to propagation of an error from one process to another. Therefore, an error in process *i* may be detected by an AT in another process *k*, where processes *i* and *k* have communicated directly or indirectly after the error has occurred. The time between the occurrence and detection of an error, referred to as error latency, is a random variable and can span many checkpoint intervals. Consequently, several checkpoints have to be maintained to ensure error recovery. These checkpoints may have been established locally for each process (local checkpointing) [11, 12] or simultaneously across all processes at the same time (global checkpointing) [3, 8, 13].

We have described two algorithms, the iterative rollback algorithm and the probabilistic (= selective) rollback algorithm, for error recovery using global checkpointing [3]. The iterative rollback algorithm works its way backwards one checkpoint at a time, starting from the most recent one, attempting error recovery after each rollback. By contrast, the probabilistic rollback algorithm determines the checkpoint to which the system rolls back based on the error latency distribution. Simulations have shown that the probabilistic rollback algorithm performed better in most cases.

The problem in implementing the probabilistic roll back algorithm is the determination of the error latency distribution. Due to the dependence introduced by the inter-communications, a theoretical determination of this distribution is not an easy problem, even in the idealized case where the waiting times between failures, inter-process communications, and ATs can be modeled as independent exponential random variables. Furthermore, even if the theoretical distribution can be determined, it will depend on the parameters for the exponential waiting times which have to be estimated from the past behavior of the system. Unlike the occurrence of inter-process communications and ATs, errors cannot be observed (since errors are recognized only when they are detected), making the estimation of the latter parameters very difficult. If other types of distributions for the

waiting times between events are assumed or the independence assumption is relaxed, the theoretical derivation of the error latency distribution becomes even more difficult, if not impossible.

In this paper, we propose a mechanism to determine an approximate error latency distribution, independent of any assumptions about the individual distribution functions or the independence of the various events, making parameter estimation obsolete. Performance of the probabilistic rollback mechanism based on this approximate latency distribution will be analyzed with a simulation that uses the same parameters as in [3].

## DESCRIPTION OF ALGORITHMS

We assume that a fixed number of (global) checkpoints (CPs) are maintained throughout, labeled 1 (oldest CP) to $m$ (most recent CP). In the *iterative* method, recovery is initially attempted from CP $m$. If recovery from CP $k$ is unsuccessful, a new attempt is made from CP $k$-1. If the failure occurred after CP 1 was established, then error recovery will eventually be successful and will start from the checkpoint just prior to the checkpoint interval in which the failure originated. By contrast, the *probabilistic* method chooses the checkpoint from which recovery is attempted by using the error latency distribution. Pairs of consecutive checkpoints are compared for lower expected cost of recovery until a (local) minimum for the cost is found. If recovery from the selected checkpoint is unsuccessful, the probabilities for error occurrence in each checkpoint interval are updated (taking into account the unsuccessful recovery) prior to the next recovery attempt.

Let $C_0$ = time needed to save/load a checkpoint, $C$ = length of checkpoint interval, and $\tilde{C}$ = cycle time = $C_0 + C$. We assume that recovery is successful if the program passes the AT at which the failure was detected[1]. If *total* = min{$m$, # of checkpoints established so far}, and $d$ = time between last checkpoint and the detection of the error, then

$T_k$ = cost of recovery from CP $k$ = (*total* - $k$) $\tilde{C} + C_0 + d$

$P_k = P($ failure occured after CP $k) = P($ error latency $\le (total - k) \cdot C + d)$

$C_k^i$ = cost of successful recovery (iterative method) from CP $i$ = $\sum_{l=k}^{i} T_l$ (given recovery starts at CP $k > i$)

$EC_k$ = expected cost of recovery given unsuccessful recovery from CP $k$ = $\sum_{l=1}^{k-1} (P_l - P_{l+1}) \cdot C_l^{k-1}$

The probabilistic method can now be described as follows:

Step 1: $k$ = min{$m$, # of current checkpoints}; $P = 0$; $TOS = 0$;
Step 2: If $k = 1$, go to Step 4.
Step 3: If $(P_{k-1} - P) \cdot T_{k-1} + (1 - P_{k-1}) \cdot EC_{k-1} \le (P_k - P) \cdot T_k + (1 - P_k) \cdot EC_k$, set *k:= k-1*; go to Step 2.
Step 4: Roll back to CP $k$ and attempt recovery. Set $P = P_k$; $TOS = TOS + T_k$;
If recovery is successful, resume normal execution. Otherwise indicate system failure.

## APPROXIMATING THE LATENCY DISTRIBUTION

From the description of the probabilistic algorithm, it is clear that we need the distribution of the latency times to compute the expected values used in the determination of the checkpoint from which the system is to be restarted. The basic idea is to initially use the iterative algorithm to record approximate latency times, and then to switch to the probabilistic algorithm. Figure 1 shows how the approximate latency times are measured.

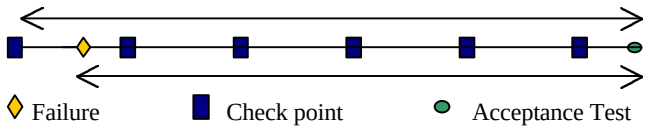


**Figure 1**

[1] This is a refined version of the assumption in [3] which does not affect the main results of [3].

The lower arrow ( « ) shows the actual latency time between occurrence of the failure and its detection. (We are only concerned with the time line, not at which individual process the failure occurred or in which process the failure was detected.) The upper arrow shows the approximate latency time resulting from the use of the iterative algorithm which can be measured easily from system records. The error made in this approximation is at most the length of the checkpoint interval and can thus be regulated by the user. Two questions arise: "How long should the checkpoint interval be during the initial phase?" and "How much data needs to be collected?" These two questions will be investigated with the help of a simulation.

## SIMULATION

The simulation consists of two parts: The first part of the simulation is used to "collect" data, both the exact latency times (possible only in a simulation), as well as the latency times as they would be measured for different lengths of the *error interval*s (= checkpoint intervals). The second part of the simulation uses the collected data to establish the number of checkpoints to be maintained, to determine the checkpoint for rollback in the probabilistic algorithm, and to compute the cost for both iterative and probabilistic algorithms.

Even though the method works for any kind of distribution, we use the same setup as in our previous paper [3], namely independent exponential waiting times between events (inter-connections between processes, failures and acceptance tests). The ranges for the average times (in hours) between inter-process communications, failures, and ATs were taken to be (0.25, 1), (10, 40), and (1, 2), respectively [8, 11, 13].

For the data collection phase, 10 simulations were executed, each for a time period of 100 hours. For each simulation, the parameters for the exponential waiting times were computed as the reciprocal of a randomly chosen number from the respective interval, for each of the 4 processes. This resulted in a total of 231 detected failures, and consequently, 231 arrays of exact and approximate failure times. The error intervals used for the approximate latency times were chosen to be 0.1 (= $20\,C_0$ ), 0.15, 0.2, and 0.25. Smaller values do not make sense as the frequent saving of the resulting large number of checkpoints would be too disruptive for the system operation. Furthermore, the iterative and probabilistic methods performed virtually identical for checkpoint interval lengths of 0.45 (~ median of the latency time distribution for the chosen simulation parameters) [3], so 0.25 was chosen as the largest data collection interval.

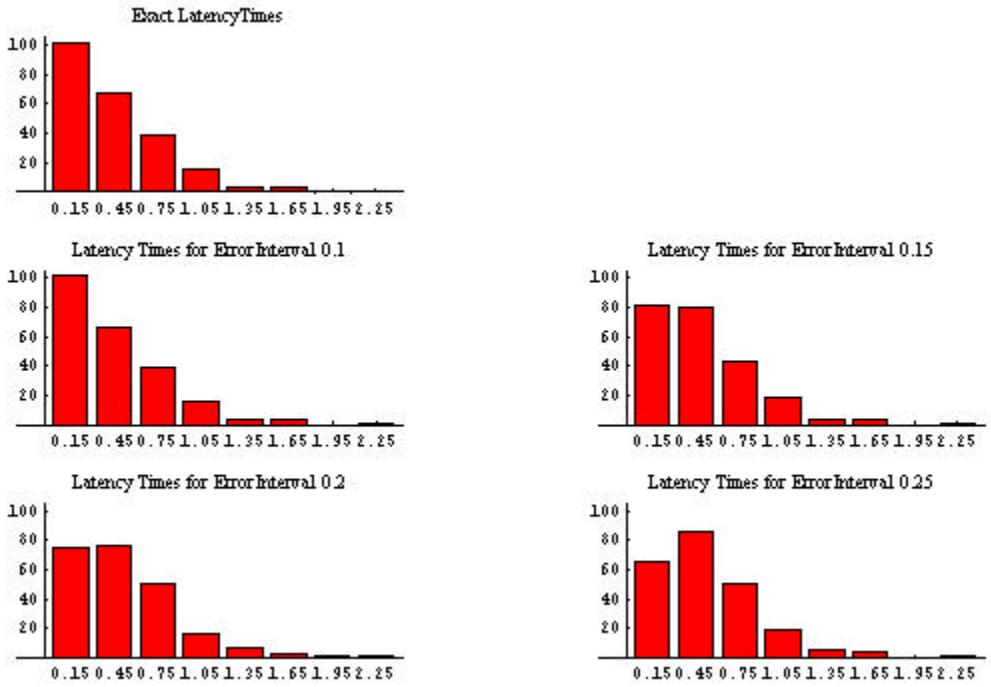


**Figure 2**

Figure 2 shows bar charts for the exact and approximate latency distributions. Notice how the structure of the distribution function changes as the error interval becomes longer. However, for $C = 0.1$, the exact and the approximate latency distribution functions have essentially the same shape.

For the second part of the simulation, in which the recovery process and its associated cost were simulated, we considered the following parameters: checkpoint interval length ($C$ = 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, and 0.45), probability of recovery ($R$ = 90% and 95%), and number of data points ($N$ = 25, 50, 75, 100, 150, and 200). For each combination of the three parameters, 30 simulations (each up to time 100) were executed, where $N$ data points were randomly selected from the 231 records. The resulting exact and approximate empirical distribution functions were then used to 1) determine the number of concurrent checkpoints needed to ensure a 90% or 95% level of recovery, and 2) compute the conditional probabilities required for the probabilistic algorithm. The cost of both the iterative and the probabilistic rollback algorithm were recorded for the five empirical distribution functions simultaneously, allowing for a comparison with regard to the length of the error interval for given values of $C$, $R$, and $N$.

## RESULTS

To measure the efficiency of the (approximate) probabilistic over the iterative algorithm, we computed the cost for the probabilistic method as a percentage of the cost for the iterative method for each simulation. Since the probabilistic method may roll back further than necessary, it can occasionally incur a larger cost than the iterative algorithm; on the other hand, the probabilistic method tends to perform much better when the latency time is large. Since it is a probabilistic method, it needs to be judged by its average performance (which is why the average was taken over the 30 simulations). The results are similar for both levels of recovery, so we will restrict ourselves to recovery level $R$ = 90% in the discussion below.

For each of the tables in Figures 3 - 5, the entries are percentages. (For easier comparison across different error intervals (= different tables), the percentages have been grouped according to the efficiency brackets indicated in the legend given in Figure 3.) The columns indicate the number of data points used to compute the distribution function, whereas the rows relate to the lengths of the checkpoint intervals used in the cost simulation. Figure 3 shows the results when using the exact empirical distribution function. Note that as the checkpoint interval length approaches 0.45, the probabilistic and the iterative methods become virtually identical; furthermore, in each case, the probabilistic method performs better or equal to the iterative method.

| | | ***N*** | | | | | |
|---|---|---|---|---|---|---|---|
| | | **25** | **50** | **75** | **100** | **150** | **200** |
| | **0.10** | 56.5 | 50.5 | 46.7 | 49.6 | 54.2 | 49.1 |
| ≤ 59.99 | **0.15** | 71.7 | 65.5 | 63.8 | 64.0 | 65.7 | 63.4 |
| 60 - 69.99 | **0.20** | 82.5 | 75.5 | 72.5 | 74.2 | 73.5 | 76.2 |
| 70 - 79.99 | ***C*** **0.25** | 88.8 | 83.1 | 82.4 | 81.3 | 80.7 | 80.4 |
| 80 - 89.99 | **0.30** | 94.4 | 88.9 | 87.2 | 86.5 | 89.1 | 87.6 |
| | **0.35** | 93.6 | 95.8 | 95.2 | 91.7 | 94.3 | 96.5 |
| | **0.40** | 97.0 | 97.9 | 94.7 | 99.3 | 97.6 | 96.7 |
| **>100.5** | **0.45** | 99.2 | 99.2 | 97.3 | 99.2 | 96.9 | 97.1 |

**Figure 3: Percentage Cost for Exact Latency Times**

Figure 4 and Figure 5 show the corresponding results when using the approximate empirical distribution function for the various lengths of the error intervals (and thus, the potential errors). With the error introduced by the approximate distribution function, we now have cases in which the iterative method is better than the probabilistic method. However, these primarily occur when the checkpoint interval is $C$ = 0.45, where the probabilistic method would usually not be used. The other occurrences of a percentage above 100 are in the columns for N = 25 (with one exception). This indicates that at least 50 data points should be used.

We will now look at each error interval separately, starting with error interval 0.1 (left table in Figure 4). If we disregard the column $N = 25$ and restrict our attention to checkpoint intervals with $C \leq 0.25$, then we can achieve the same efficiency brackets as with the exact latency times: $C = 0.1$ leads to at least a 40% reduction, $C = 0.15$ leads to at least a 30% reduction, $C = 0.2$ leads to at least a 20% reduction, and finally $C = 0.25$ can achieve at least a 10% reduction in cost over the iterative method. The individual percentages for the entries in the left table of Figure 4 are slightly higher than those in Figure 3, but the brackets are the same. This indicates that for an error interval of size 0.1 and $N > 25$, the approximate probabilistic algorithm performs comparably to the probabilistic algorithm based on the exact latency times for $C \leq 0.25$.

| | | N | | | | | |
|---|---|---|---|---|---|---|---|
| | | **25** | **50** | **75** | **100** | **150** | **200** |
| C | **0.1** | 54.9 | 53.3 | 53.1 | 52.1 | 57.2 | 51.6 |
| | **0.15** | 79.6 | 69.8 | 65.1 | 67.2 | 69.7 | 66.9 |
| | **0.2** | 85.2 | 76.6 | 72.8 | 78.4 | 76.1 | 75.8 |
| | **0.25** | 93.6 | 80.8 | 85.1 | 88.9 | 84.3 | 81.7 |
| | **0.3** | 97.5 | 97.4 | 87.6 | 87.1 | 98.8 | 94.0 |
| | **0.35** | **100.8** | 95.1 | 94.4 | 97.7 | 93.7 | 93.5 |
| | **0.4** | 99.7 | 95.6 | 95.7 | 94.6 | 95.9 | 96.5 |
| | **0.45** | **100.7** | **108.9** | 99.1 | 99.5 | **107.4** | **109.6** |

| | | N | | | | | |
|---|---|---|---|---|---|---|---|
| | | **25** | **50** | **75** | **100** | **150** | **200** |
| C | **0.1** | 60.5 | 50.6 | 50.3 | 52.0 | 57.1 | 51.6 |
| | **0.15** | 79.6 | 68.1 | 64.9 | 65.3 | 70.9 | 67.8 |
| | **0.2** | 87.7 | 77.0 | 72.9 | 82.1 | 80.2 | 76.1 |
| | **0.25** | 95.9 | 83.9 | 85.5 | 89.0 | 86.8 | 82.5 |
| | **0.3** | **100.6** | 91.4 | 88.0 | 87.9 | 99.2 | 94.1 |
| | **0.35** | **102.6** | 96.0 | 96.4 | 94.0 | 94.9 | 95.3 |
| | **0.4** | **102.6** | 95.7 | 96.5 | 95.7 | 96.1 | 97.0 |
| | **0.45** | **107.0** | 97.9 | 100.0 | 99.6 | **109.1** | **110.2** |

**Figure 4: Percentage Cost for Error Interval 0.1 (left table) and 0.15 (right table)**

As the error interval gets larger, the efficiency varies more. For error interval 0.15 (right table in Figure 4), we can still achieve at least a 40% reduction with C = 0.1. However, there is now one case ($N = 150$), where for C = 0.15, the cost reduction is not quite 30%. If we relax our efficiency requirement by 2% (e.g. from 60% to 62%), then we have the same structure as with error interval 0.1. If the efficiency brackets remain the same as before, then we have to use a smaller checkpoint interval to achieve the same brackets: $C = 0.1$ for at least a 30% reduction, $C = 0.15$ for at least a 20% reduction, and $C = 0.2$ or $C = 0.25$ for at least a 10% cost reduction.

The observations made for error interval 0.15 also apply in the case of error interval 0.2 (left table in Figure 5), except now we would have to adjust our efficiency brackets by 3.5% to achieve the brackets of the exact distribution function.

| | | N | | | | | |
|---|---|---|---|---|---|---|---|
| | | **25** | **50** | **75** | **100** | **150** | **200** |
| C | **0.1** | 62.0 | 52.3 | 52.4 | 57.4 | 58.1 | 52.7 |
| | **0.15** | 81.1 | 70.8 | 67.5 | 70.1 | 71.1 | 68.2 |
| | **0.2** | 90.5 | 77.2 | 75.0 | 83.5 | 78.5 | 76.1 |
| | **0.25** | 94.0 | 83.8 | 84.9 | 89.6 | 86.4 | 82.6 |
| | **0.3** | **102.0** | 97.9 | 88.0 | 88.5 | 98.1 | 93.3 |
| | **0.35** | **103.9** | 92.2 | 97.7 | 99.3 | 95.2 | 94.0 |
| | **0.4** | **101.7** | 93.7 | 96.3 | 93.9 | 96.5 | 97.4 |
| | **0.45** | **101.8** | **109.4** | 100.1 | **101.4** | **109.5** | **110.1** |

| | | N | | | | | |
|---|---|---|---|---|---|---|---|
| | | **25** | **50** | **75** | **100** | **150** | **200** |
| C | **0.1** | 65.5 | 58.5 | 52.5 | 54.0 | 62.6 | 53.3 |
| | **0.15** | 88.9 | 77.5 | 68.5 | 68.7 | 75.8 | 69.9 |
| | **0.2** | 98.9 | 85.1 | 75.1 | 83.5 | 85.1 | 78.2 |
| | **0.25** | **100.9** | 89.4 | 89.1 | 89.2 | 93.2 | 83.5 |
| | **0.3** | **108.5** | 97.6 | 92.2 | 88.5 | **101.3** | 94.1 |
| | **0.35** | **107.1** | 100.1 | 98.2 | 99.7 | 94.4 | 93.2 |
| | **0.4** | **109.3** | 94.3 | 96.6 | 94.4 | 96.1 | 97.2 |
| | **0.45** | **107.3** | **108.6** | 99.7 | 100.1 | **109.6** | **109.6** |

**Figure 5: Percentage Cost for Error Interval 0.2 (left table) and 0.25 (right table)**

Finally, for error interval 0.25 (right table in Figure 5), we can no longer achieve at least a 40% cost reduction with $C = 0.1$. In general, the changes have become quite large compared to the results for exact latency times, indicating that a smaller error interval should be chosen.

Overall, the cost coefficients displayed in Figure 4 and Figure 5 suggest that with error interval 0.1, the approximate probabilistic method performs almost as well as the probabilistic method based on the exact distribution function. This can be achieved with a modest amount of data collection, namely $N = 50$. For larger error intervals (0.15 and 0.2), the user may have to choose a smaller checkpoint interval or collect more data points to achieve the same efficiency bracket. Finally, the probabilistic method based on an error interval of 0.25 does not consistently outperform the iterative algorithm, even for $N > 25$ and $C < 0.45$, indicating that the error interval should be less than 0.25.

One possibly surprising result, namely that performance does not always increase with increased data collection, can be explained by the fact that for each combination of parameter values for $C$, $N$, and $R$, a new set of parameters for the exponential waiting times was selected. This variation in parameters carries through to the cost coefficients. In a future simulation, the cost will be computed simultaneously not only for the different error intervals, but also for the different values of $N$.

## CONCLUSION

We have analyzed the performance of the probabilistic algorithm based on an approximate empirical distribution function for the latency times. This approximate distribution function was computed by initially employing the iterative rollback algorithm. The simulation showed that for an error interval of 0.1, with a modest number of data, $N = 50$, the algorithm based on the approximate distribution function achieved the same efficiency brackets as the one based on the exact latency times. Depending on the lengths of the checkpoint intervals used with the probabilistic algorithm, a cost reduction of up to 40% (over the iterative algorithm) can be achieved. This method of computing the approximate empirical distribution function does not depend on the distribution functions of the waiting times between inter-connections, failures and ATs, nor on independence assumptions; therefore, this approximate probabilistic rollback method is universally applicable.

## REFERENCES


[1] A.K. Somai and N.H. Vaidya, *Understanding Fault Tolerance and Reliability*, Computer, 1997.

[2] B. Randell, *System Structure for Software Fault Tolerance*, IEEE Transactions on Software Engineering, June 1995.

[3] S. Heubach and R. Pamula, *Modeling and Simulation of Error Recovery in a Concurrent Processing System*, Proceedings of the 2nd International IASTED Conference: European Parallel and Distributed Systems (Euro-PDS'98), IASTED/ACTA Press, pp. 29-35, 1998

[4] R. Koo and S. Toueg, *Checkpointing and Rollback -Recovery for Distributed Systems*, IEEE Transactions on Software Engineering, January 1987.

[5] K. Mani Chandi and Leslie Lamport, *Distributed Snapshots: Determining Global States of Distributed Systems*, ACM Transactions on Computer Systems, February 1985.

[6] K.L. Wu et al., *Error Recovery in Shared Memory Multiprocessors Using Private Caches*, IEEE Transactions on Parallel and Distributed Systems, April 1990.

[7] K.M. Chandy et al., *Analytic Models For Rollback And Recovery Strategies In Database Systems*, IEEE Trans. on Software Engineering, March 1975.

[8] Krishna Kant, *A Model For Error Recovery With Global Checkpointing*, Information Sciences, No. 30, 1978.

[9] J. Gray and D. Siewiorek, *High Availability Computer Systems*, Computer, Sep. 1991.

[10] Y.K. Malaiya and S.Y.H. Su, *Reliability Measure of Hardware Redundancy Fault-Tolerant Digital Systems with Intermittent Faults*, IEEE Trans. on Computers, Aug. 1981.

[11] K.G. Shin and Y.H. Lee, *Error Detection Process - Model, Design, and Impact on Computer Performance*, IEEE Trans. on Computers, June 1984.

[12] G.S. Kang and Y.H. Lee, *Analysis Of Backward Error Recovery For Concurrent Processes With Recovery Blocks*, Proceedings of International Conference on Parallel Processes, June 1983.

[13] R.S. Pamula et al., *Global Checkpointing For A Concurrent Processing System*, Int. Journal of Systems Sciences, 1990.