Estimating error-probability and its application for optimizing Roll-back Recovery with Checkpointing

Dimitar Nikolov[†], Urban Ingelsson[†], Virendra Singh[‡] and Erik Larsson[†]

Department of Computer Science[†]
Linköping University
Sweden

Supercomputer Education and Research Centre[‡]
Indian Institute of Science
India

Abstract[1]

The probability for errors to occur in electronic systems is not known in advance, but depends on many factors including influence from the environment where the system operates. In this paper, it is demonstrated that inaccurate estimates of the error probability lead to loss of performance in a well known fault tolerance technique, Roll-back Recovery with checkpointing (RRC). To regain the lost performance, a method for estimating the error probability along with an adjustment technique are proposed. Using a simulator tool that has been developed to enable experimentation, the proposed method is evaluated and the results show that the proposed method provides useful estimates of the error probability leading to near-optimal performance of the RRC fault-tolerant technique.

I. Introduction

The rapid development in semiconductor technologies makes it possible to fabricate integrated circuits (ICs) that contain billions of transistors. The constant need of performance, which traditionally has been met by higher clock frequencies, is today also increasingly met by concurrency whereas a number of processor cores are implemented on the same silicon die; these ICs are often referred to as multi-processor system-on-chips (MPSoCs).

The drawback of the rapid development in semiconductor technologies is the increasing susceptibility to soft errors [4], [6]. Roll-back recovery with checkpointing (RRC) is a technique designed to detect and recover from soft errors. A checkpoint is a snapshot of the state of a processor node. From the checkpoint, the job execution can restart (roll-back) if an error is detected. Errors can be detected by comparing checkpoints for two processor nodes that are executing the same job. The combination of detection of errors in jobs and re-execution when errors are detected provides fault tolerance at the expense of not only employing two processor nodes, but also comparing checkpoints and re-executing.

If the error probability is known, it is possible to find an optimal number of checkpoints so that the overhead of the RRC technique is minimized [8]. However, error probability is not known in advance and is difficult to estimate. In this paper, it is demonstrated that inaccurate estimates of error probability lead to loss of performance and two novel techniques are proposed to provide on-line estimation of this probability to regain the lost performance. The proposed techniques are (1) Periodic Probability Estimation (PPE) and (2) Aperiodic Probability Estimation (APE). Thorough analysis on the proposed techniques is presented. Furthermore, a simulator tool is presented which has been developed to enable experimentation with the proposed estimation techniques. The obtained results show that both proposed approaches lead to near-optimal average execution time.

While most work addressing RRC [1], [7], [8], [9], [10], [11], [12], [13], [14], [15], focuses on finding strategies and presenting analysis for optimal checkpoint placements, these strategies depend on failure rate, which corresponds to error probability. However, to the extent of the authors knowledge, no work has jointly addressed the problem of error probability estimation with on-line checkpoint adjustment to optimize RRC.

This paper is organized as follows. In Section II we present the preliminaries of our work. Section III demonstrates the need of accurate error probabilities. Two estimation techniques are presented in Section IV. Section V elaborates on the developed simulator, which is used for presenting experimental results later in section VI. We conclude with Section VII.

II. Preliminaries

We assume an MPSoC architecture, described in Figure 1, which consists of n processor nodes, a shared memory, and a compare & control unit. The processor nodes include private memory, and the shared memory, which is common for all processor nodes, is used for communication between processors. The compare & control unit, added for fault tolerance, detects whether errors have occurred by comparing the contexts (checkpoints) of two processors executing the same job. We address errors that occur in the processors, and we assume that errors that occur elsewhere (buses and memories) can be handled by other techniques such as error correction codes.

[1]The research is partially supported by The Swedish Foundation for International Cooperation in Research and Higher Education (STINT) by an Institutional Grant for Younger Researchers.

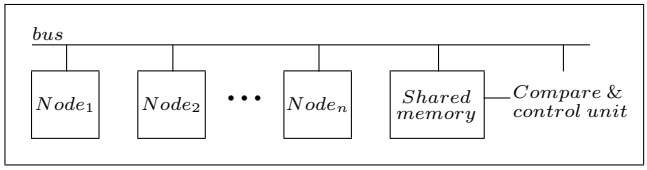

Figure 1: MPSoC architecture with n processor nodes, a shared memory and a compare & control unit

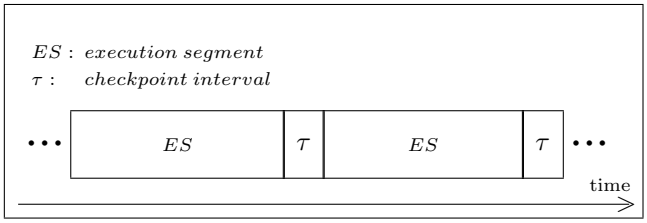

Figure 2: Execution segments and checkpoint intervals

In RRC, each job is executed concurrently on two processors and a number of checkpoints are inserted to detect errors. A given job is divided into a number of *execution segments* and between every execution segment there is a *checkpoint interval*. The checkpoint interval represents the time required to take a checkpoint. Figure 2 illustrates the execution segments and the inserted checkpoint intervals. When a job is executed and a checkpoint is reached, both processors send their respective contexts to the compare & control unit. The compare & control unit compares the contexts. If the contexts differ, meaning that an error has occurred during the last execution segment, this execution segment is to be re-executed. In the case that the contexts of the processors do not differ, meaning that there is no error, the execution proceeds with the next execution segment.

There is a trade off when choosing the number of checkpoints. Smaller number will increase the time spent in re-execution and larger number will increase the overall time due to the time required to process checkpoints. Väyrynen *et al.* addressed this problem considering the average execution time (AET), which is the expected time for a job to complete for a given error probability [8]. They proposed a mathematical framework for the analysis of AET, and presented an equation for computing the optimal number of checkpoints. The AET when applying RRC on a job is given as:

$$AET(P,T) = \frac{T + n_c \times (2 \times \tau_b + \tau_c + \tau_{oh})}{\sqrt[n_c]{(1-P)^2}} \qquad (1)$$

where P is the error probability per time unit, T is the fault-free execution time, n_c is the number of checkpoints, and τ_b, τ_c and τ_{oh} are time parameters due to checkpoint overhead. Given Eq. (1), Väyrynen *et al.* showed that the optimal number of checkpoints (n_c) is given as:

$$n_c(P,T) = -\ln(1-P) + \sqrt{(\ln(1-P))^2 - \frac{2 \times T \times \ln(1-P)}{2 \times \tau_b + \tau_c + \tau_{oh}}} \qquad (2)$$

Using the optimal number of checkpoints, n_c, (Eq. (2)), the optimal AET can be calculated with Eq. (1).

The computation of optimal number of checkpoints requires the following parameters: error probability (P), fault-free execution time (T), and parameters for checkpoint overhead (τ_b, τ_c and τ_{oh}). The parameters for checkpoint overhead can be estimated at design time; however it is difficult to accurately estimate error probability. The real error probability cannot be known at design time, it is different for different ICs, and it is not constant through the lifetime of an IC due to for example aging and the environment where the IC is to be used [1] [2] [3] [5].

III. Importance of accurate error probability estimates

In this section we demonstrate the importance of having accurate error probability estimates by presenting the impact of inaccurate error probability estimates on the number of checkpoints and the resulting AET. The optimal number of checkpoints (n_c) depends on error probability. However, the *real* (actual) error probability is not known at design time and further it can vary over the product's life time (time in operation). Because of this fact, it is common that the *initial* chosen error probability used for calculating the optimal number of checkpoints, and thus obtain the optimal AET, will differ from the *real* error probability. The *initial* chosen error probability value is an inaccurate error probability estimate.

The *inaccurate* estimate on error probability, results in a value for n_c which will differ from the optimal, and thus lead to an AET larger than the optimal. Eq. (3) denotes AET when the estimated error probability p is used to obtain the number of checkpoints n_c (Eq. (2)).

$$AET_{est_p}(P,T,p) = \frac{T + (2 \times \tau_b + \tau_c + \tau_{oh}) \times n_c(p,T)}{\sqrt[n_c(p,T)]{(1-P)^2}} \qquad (3)$$

It should be noted in Eq. (3) that the AET is equal to the optimal AET when estimated error probability, p, is equal to the *real* error probability, P, and thus $AET_{est_p}(P,T,P) = AET(P,T)$.

To quantify the impact of inaccurate error probability we use:

$$AET_{dev} = \frac{AET_{est_p}(P,T,p) - AET(P,T)}{AET(P,T)} \times 100\% \qquad (4)$$

where P is the *real* error probability and p is the *estimated* error probability. This equation represents the relative deviation in AET compared to the optimum, when estimate on error probability is used for obtaining the number of checkpoints.

To illustrate the impact of inaccurate estimation of error probability we have taken three jobs, all with the fault-free execution time T=1000 time units but different *real* error probabilities, that is P to be 0.5, 0.2 and 0.1. Figure 3

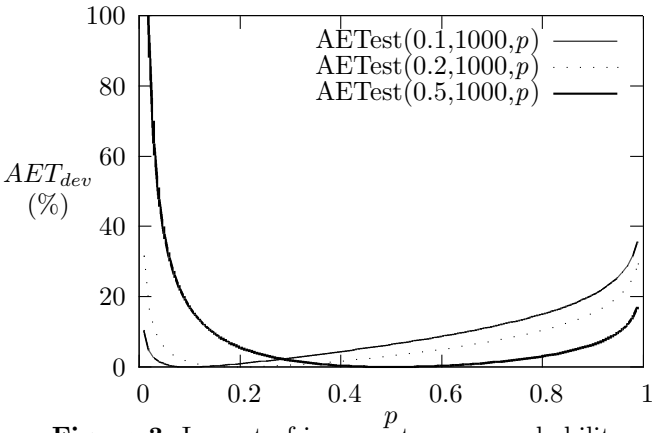

Figure 3: Impact of inaccurate error probability estimation relative to optimal AET (%)

shows the three cases at various estimated error probabilities versus the performance degradation (AET_{dev}). The x-axis represents the estimated error probabilities and the y-axis shows the relative deviation in AET (Eq. (4)). Each curve shows no deviation in AET when the estimated error probability (p) is equal to the *real* error probability (P). However, as soon as $p \neq P$, AET_{dev} is increased. This means that assuming an error probability other than the real one, leads to AET which is not the optimal. The increase in AET due to inaccurate error probability estimation represents the loss of performance.

IV. Approaches for error probability estimation and corresponding adjustment

In this section we present approaches that estimate error probability with the aim to adjust and optimize RRC during operation. To make use of the estimates on error probability, we need to estimate error probability during operation. One way to achieve this is to extend the architecture described earlier (Figure 1) by employing a history unit that keeps track on the number of successful (no error) executions of execution segments (n_s) and the number of erroneous execution segments (execution segments that had errors) (n_e). Having these statistics, error probability can be estimated during time, periodically or aperiodically. Thus we come up with one periodic approach, which we address as Periodic Probability Estimation (PPE), and one aperiodic, which we address as Aperiodic Probability Estimation (APE). For both approaches we need some initial parameters, *i.e.* initial estimate on error probability and adjustment period. It should be noted, that the adjustment period is kept constant for PPE, while for APE it is tuned over time.

A. Periodic Probability Estimation

PPE assumes a fixed T_{adj} and elaborates on p_{est} as:

$$p_{est} = \frac{n_e}{n_e + n_s} \tag{5}$$

where n_s is the number of successful (no error) executions of execution segments and n_e is the number of erroneous

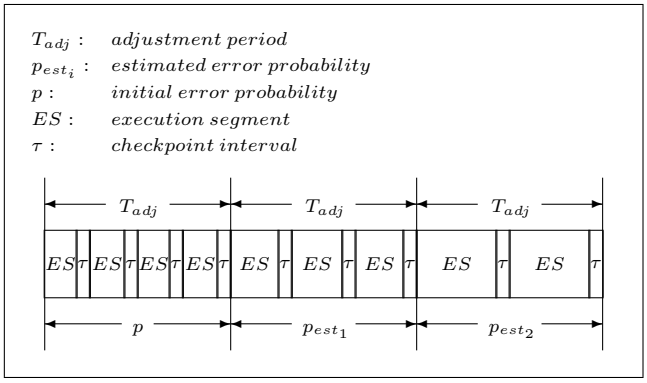

Figure 4: Graphical presentation of PPE

execution segments. As can be seen from Figure 4 estimates on error probability, p_{est}, are calculated periodically at every T_{adj}. The value of p_{est} is used to obtain the optimal number of checkpoints, n_c. During an adjustment period, n_c equidistant checkpoints are taken. So the checkpoint frequency, *i.e.* number of checkpoints during time interval, changes according to the changes of the error probability estimates.

B. Aperiodic Probability Estimation

APE elaborates on both T_{adj} and p_{est}. The idea for this approach comes from the following discussion. As this approach is an estimation technique, it is expected that during operation the estimates will converge to the real values, so we should expect changes on the estimated error probability during time. These changes can guide how to change the checkpointing scheme. If the estimates on error probability start decreasing, that implies that less errors are occurring and then we want to do less frequent checkpointing, so we increase the adjustment period. On the other hand, if the estimates on error probability start increasing, that implies that errors occur more frequently, and to reduce the time spent in re-execution we want more frequent checkpointing, so we decrease the adjustment period.

If the control & compare unit encounters that error probability has not changed in two successive adjustment periods, it means that during both adjustment periods the system has done a number of checkpoints which is greater than the optimal one. This can be observed by:

$$2 \times n_c(P, T_{adj}) > n_c(P, 2 \times T_{adj}) \tag{6}$$

In APE, error probability is estimated in the same manner as PPE, *i.e.* using Eq. (5). What distinguishes this approach from PPE, is that the adjustment period is updated during time. Eq. (7) describes the scheme for updating the adjustment period.

$$\begin{aligned} &if \quad p_{est_{i+1}} > p_{est_i} \quad then \\ &\quad T_{adj_{i+1}} = T_{adj_i} - T_{adj_i} \times \alpha \\ &else \\ &\quad T_{adj_{i+1}} = T_{adj_i} + T_{adj_i} \times \alpha \end{aligned} \tag{7}$$

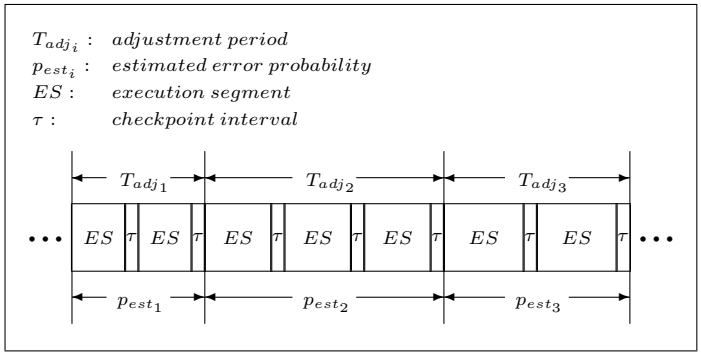

Figure 5: Graphical presentation of APE

The APE approach is illustrated in Figure 5. After every T_{adj} time units, the control & compare unit, computes a new error probability estimate ($p_{est_{i+1}}$) using the Eq. (5). The latest estimate ($p_{est_{i+1}}$) is then compared against the recent value (p_{est_i}). If estimation of error probability increases, meaning that during the last adjustment period, (T_{adj_i}), more errors have occurred, the next adjustment period, ($T_{adj_{i+1}}$), should be decreased to avoid expensive re-executions. However, if the estimation of error probability decreases or remains the same, meaning that less or no errors have occurred during the last adjustment period, (T_{adj_i}), the next adjustment period, ($T_{adj_{i+1}}$), should be increased to avoid excessive checkpointing.

V. Experimental Setup

To conduct experiments, we have developed a simulator that emulates the execution of a job. There are two types of inputs to the simulator, *designer* inputs and *environmental* inputs. The designer inputs refer to inputs that initialize the system, *i.e.* the initial estimated error probability, p, and the adjustment period, T_{adj}. Environmental inputs refer to the real error probability, P, and the expected fault-free execution time, T. The real error probability is modeled as a function that can change over time as error probability is not constant. This input is used for generating errors while simulating the approaches. The output of the simulator is the AET.

VI. Experimental Results

We have simulated three approaches: Periodic Probability Estimation (PPE) IV-A, Aperiodic Probability Estimation (APE) IV-B, and Baseline Approach (BA). The BA takes the designer inputs, *i.e.* the initial estimated error probability, p, and the adjustment period, T_{adj}, and computes an optimal number of checkpoints, n_c, for these inputs using Eq. (2). Further, it takes checkpoints at a constant frequency n_c/T_{adj}, and no adjustments are done during execution.

We made experiments to determine an appropriate value for α parameter in APE. The experiment was repeated for different values for the real error probability and for the adjustment period T_{adj}, and it was found that out of the considered values, $\alpha = 0.15$ provided the best results, *i.e.* the lowest deviation from optimal AET.

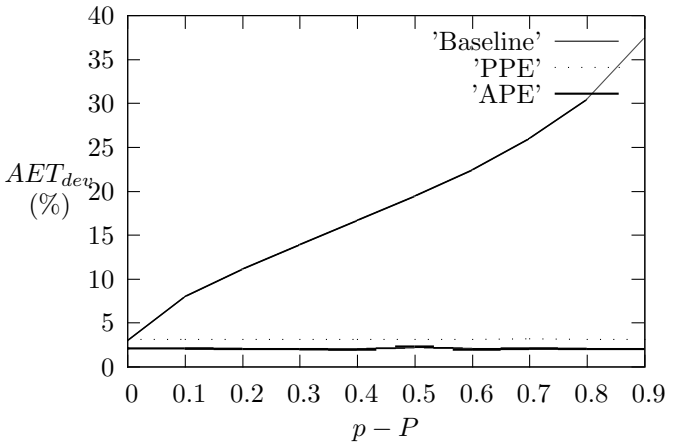

Figure 6: Relative deviation from optimal AET (%) for constant *real* error probability $P = 0.01$

We conducted two sets of experiments. In the first set, we have examined the behavior of the approaches when the real error probability is constant during time, while in the second set, the real error probability changes over time, following a predefined profile. Each approach is simulated for 1000 times with the same inputs.

In the first set of experiments, we compare the three simulated approaches: PPE, APE and BA against the optimal solution in terms of AET (%). The optimal solution is obtained by using the equations proposed by Väyrynen *et al.* [8] and using the environmental inputs as inputs for these equations. We have made several experiments, by varying both the designer and environmental inputs. In Figure 6 we present (1) on the y-axis the deviation of the AET, from the simulated approaches, relative to the optimal AET in %, and (2) on the x-axis the difference between the initial estimated error probability, p, and the real error probability, P. We assume a constant *real* error probability, $P = 0.01$, and fault-free execution time $T = 1000000$ time units. We choose the adjustment period to be $T_{adj} = 1000$ time units, and then simulate the approaches with different values for the initial estimated error probability, p. One can observe from Figure 6 that APE and PPE do not depend significantly on the initial estimated error probability, p. Both APE and PPE always perform better than the BA approach. The small deviation in the AET for PPE and APE, relative to the optimal AET, shows that both approaches make a good estimation on the *real* error probability. Furhter, Figure 6 shows that APE performs slightly better than PPE.

In the second set of experiments, we show how the approaches behave when *real* error probability changes over time. For this purpose, we define different error probability profiles showing how error probability changes over time, and then we run simulations for each of these profiles. Three probability profiles are presented in Table I. We as-

$$P1(t) = \begin{cases} 0.01, & 0 \le t < 200000 \\ 0.02, & 200000 \le t < 400000 \\ 0.03, & 400000 \le t < 600000 \\ 0.02, & 600000 \le t < 800000 \\ 0.01, & 800000 \le t < 1000000 \end{cases}$$

$$P2(t) = \begin{cases} 0.02, & 0 \le t < 350000 \\ 0.01, & 350000 \le t < 650000 \\ 0.02, & 650000 \le t < 1000000 \end{cases}$$

$$P3(t) = \begin{cases} 0.01, & 0 \le t < 90000 \\ 0.10, & 90000 \le t < 100000 \end{cases}$$

Table I: Error probability profiles

Probability Profile	Approaches		
	Baseline	PPE	APE
P1	55.93%	4.50%	2.84%
P2	50.69%	4.53%	2.74%
P3	56.02%	4.65%	2.50%

Table II: Relative deviation from fault-free execution time (%) for variable real error probability

sume that the probability profiles are repeated periodically over time. The results in Table II present the deviation of the AET, from the simulated approaches, relative to the fault-free execution time in %. For these simulations, we choose the adjustement period to be $T_{adj} = 1000$ time units and the initial estimated error probability to be equal to the real error probability at time 0, *i.e.* $p = P(0)$. We assume fault-free execution time of $T = 1000000$ time units. As can be seen from Table II, both PPE and APE perform far better than BA, with a very small deviation in average execution time relative to the fault-free execution time. Again we notice that APE gives slightly better results than PPE approach.

VII. Conclusion

Fault tolerance becomes a challenge with the rapid development in semiconductor technologies. However, many fault tolerance techniques have a negative impact on performance. For one such technique, Roll-back Recovery with Checkpointing, which inserts checkpoints to detect and recover from errors, the checkpointing frequency is to be optimized to mitigate the negative impact on performance. However, the checkpointing frequency depends on error probability which cannot be known in advance.

In this paper we have analyzed the impact of error probability estimates on the performance, and we have proposed two techniques to estimate error probability with the aim to reduce the average execution time. These two techniques are a periodic approach that continuously collects information and periodically estimates the error probability given an adjustment period, and an aperiodic approach where the adjustment period is tuned and the error probability is estimated after every adjustment period. To perform experiments we have implemented a simulator. The simulator runs the proposed approaches given designer inputs, (the initial estimated error probability, and the adjustment period), and environmental inputs, (the real error probability and the expected fault-free execution time). To realistically model the real error probability, we model it as a function that may vary over time. Using the simulator, we have conducted experiments that demonstrate the benefits of the proposed approaches, which are:

- Given an initial error probability estimate, which differs from the real error probability, both approaches achieve results comparable to the theoretical optimum,
- Both approaches perform well even in the scenario where real error probability changes over time.

From the results we also notice, that the proposed aperiodic approach gives slightly better results than the periodic approach, in terms of average execution time.

References

[1] I. Koren and C. M. Krishna, "Fault-Tolerant Systems", Morgan Kaufman, 1979.

[2] E.H. Cannon, A. KleinOsowski, R. Kanj, D. D. Reinhardt, and R. V. Joshi, "The Impact of Aging Effects and Manufacturing Variation on SRAM Soft-Error Rate", *IEEE Trans. Device and Materials Reliability*, vol. 8, no. 1,pp. 145-152, March 2008

[3] V. Lakshminarayanan, "What causes semiconductor devices to fail?", Centre for development of telematics, Bangalore, India – *Test & Measurement World*, 11/1/1999.

[4] V. Chandra, and R. Aitken, " Impact of Technology and Voltage Scaling on the Soft Error Susceptibility in Nanoscale CMOS", *IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems*, pp. 114-122, Oct. 2008

[5] T. Karnik, P. Hazucha, and J. Patel, "Characterization of Soft Errors Caused by Single Event Upsets in CMOS Processes", *IEEE Trans. on Dependable and secure computing*, vol. 1, no. 2, April-June 2004

[6] J. Borel, "European Design Automation Roadmap", 6th Edition, March 2009

[7] D. K. Pradhan and N. H. Vaidya, "Roll-Forward Checkpointing Scheme: A Novel Fault-Tolerant Architecture", *IEEE Transactions on computers*, vol. 43, no. 10, pp. 1163-1174, October 1994

[8] M. Väyrynen, V. Singh, and E. Larsson, " Fault-Tolerant Average Execution Time Optimization for General-Purpose Multi-Processor System-on-Chips", *Design Automation and Test in Europe (DATE 2009)*, Nice, France, April, 2009.

[9] Y. Ling, J. Mi and X. Lin, "A Variational Calculus Approach to Optimal Checkpoint Placement", *IEEE Transactions on computers*, vol. 50, no.7, pp. 699-708, July 2001.

[10] J.L. Bruno and E.G. Coffman,"Optimal Fault-Tolerant Computing on Multiprocessor Systems", *Acta Informatica*, vol. 34, pp. 881-904, 1997.

[11] E.G. Coffman and E.N. Gilbert, "Optimal Strategies for Scheduling Checkpoints and Preventive Maintenance", *IEEE Trans. Reliability*, vol. 39, pp. 9-18, Apr. 1990.

[12] P. L'Ecuyer and J. Malenfant, "Computing optimal checkpointing strategies for rollback and recovery systems", *IEEE Trans. Computers*, vol. 37, no. 4, pp. 491-496, 1988.

[13] E. Gelenbe and M. Hernandez, "Optimum Checkpoints with Age Dependent Failures", *Acta Informatica*, vol. 27, pp. 519-531, 1990.

[14] C.M. Krishna, K.G. Shin, and Y.H. Lee, "Optimization Criteria for Checkpoint Placements", *Comm. ACM*, vol. 27, no. 10, pp. 1008-1012, Oct. 1984.

[15] V.F. Nicola, "Checkpointing and the Modeling of Program Execution Time", *Software Fault Tolerance*, M.R. Lyu, ed., pp. 167-188, John Wiley&Sons, 1995.