Level of Confidence Evaluation and Its Usage for Roll-back Recovery with Checkpointing Optimization

Nikolov, Dimitar; Ingelsson, Urban; Singh, Virendra; Larsson, Erik

Published in:
2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W)

DOI:
10.1109/DSNW.2011.5958836

2011

Level of Confidence Evaluation and Its Usage for Roll-back Recovery with Checkpointing Optimization

Dimitar Nikolov[†] dimitar.nikolov@liu.se
Urban Ingelsson[†] urban.ingelsson@liu.se
Virendra Singh[‡] viren@serc.iisc.ernet.in
and
Erik Larsson[†] erik.larsson@liu.se

[†]Department of Computer Science
Linköping University, Sweden

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

Abstract

Increasing soft error rates for semiconductor devices manufactured in later technologies enforces the use of fault tolerant techniques such as Roll-back Recovery with Checkpointing (RRC). However, RRC introduces time overhead that increases the completion (execution) time. For non-real-time systems, research have focused on optimizing RRC and shown that it is possible to find the optimal number of checkpoints such that the average execution time is minimal. While minimal average execution time is important, it is for real-time systems important to provide a high probability that deadlines are met. Hence, there is a need of probabilistic guarantees that jobs employing RRC complete before a given deadline. First, we present a mathematical framework for the evaluation of level of confidence, the probability that a given deadline is met, when RRC is employed. Second, we present an optimization method for RRC that finds the number of checkpoints that results in the minimal completion time while the minimal completion time satisfies a given level of confidence requirement. Third, we use the proposed framework to evaluate probabilistic guarantees for RRC optimization in non-real-time systems.

I. Introduction

As semiconductor technologies are increasingly susceptible to soft errors, it is for computer systems (both real-time and non-real-time) becoming important to employ fault-tolerant techniques to detect and recover from soft errors. However, fault tolerance comes at a cost and usually degrades the performance of the system. To minimize the performance degradation it is important to analyze and optimize the usage of fault tolerance such that the performance degradation is minimized. In this paper we study Roll-back Recovery with Checkpointing (RRC).

Instead of executing the complete job and in case of errors, re-execute the complete job, RRC makes use of checkpoints such that if an error is detected, a job is rolled back from the most recently saved checkpoint. Saving checkpoints, introduces a time overhead that depends on the number of checkpoints. A high number of checkpoints leads to early error detection, and thus the penalty of re-execution from the recently saved checkpoint becomes less expensive in time. However, a high number of checkpoints causes more time overhead due to checkpointing, which increases the total execution time for the job. It is a problem to find the optimal number of checkpoints.

RRC has been the subject of research for both non-real-time [1], [2] [3], [4] and real-time systems [5], [6], [7], [8], [9]. While for non-real-time systems, it is important to minimize the average execution time when RRC is applied, it is for real-time systems important to maximize the probability that a job meets a given deadline, [10]. When using RRC in real-time systems, both hard and soft, it is important to provide a reliability metric that indicates the probability of meeting deadlines. However, to the extent of our knowledge no such reliability metrics have been presented. The contribution of this paper is three-fold. First, we derive for real-time systems an expression to evaluate the probability that a job employing RRC meets a given deadline, *i.e.* the level of confidence. Second, as time overhead is to be minimized we propose an optimization method that finds the optimal number of checkpoints that results in a minimal completion time that satisfies a given level of confidence requirement. Third, we evaluate probabilistic guarantees for non-real-time system optimization of RRC, using our mathematical framework.

II. System model

In this section we detail the Roll-back Recovery with Checkpointing (RRC) scheme and we provide some basic assumptions regarding the occurrence of soft errors.

The RRC scheme that we adopt assumes that a job is duplicated and concurrently executed on two processors (illustrated in Figure 1). During the execution of a job, a number of checkpoints are taken and compared against each other. If the checkpoints match, they are saved as a safe point from which a job can be restarted. If the checkpoints mismatch, this indicates that errors have occurred and therefore the job is restarted in both processors from the most recently saved checkpoint. In the scheme, RRC provides fault tolerance at expense of hardware redundancy, *i.e* two processors execute the same job, and time redundancy, *i.e.* taking and comparing checkpoints introduces a time overhead. We define checkpointing overhead, τ (see Figure 1), as the time required to carry out checkpoint operations, *i.e.* to load/store a checkpoint and compare the checkpoints from the two processors. We assume constant τ for each checkpoint. We assume that RRC handles soft errors that occur in the processors, while for errors that occur elsewhere other fault-tolerant techniques are used.

978-1-4577-0375-1/11/$26.00 ©2011 IEEE

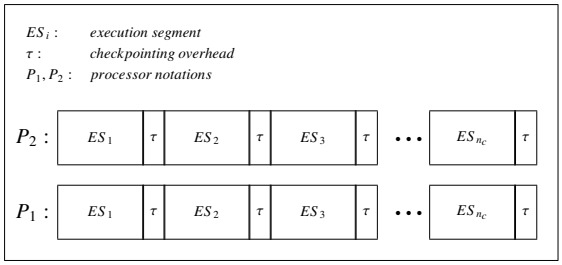


Figure 1: Graphical presentation of RRC scheme

We define the portion of a job's execution between two successive checkpoints as an execution segment (see Figure 1). We refer to an execution segment as successful execution segment if no errors have occurred during the execution in both processors, or erroneous execution segment otherwise.

For a job, we assume given is the processing time, T, which is the time required for a job to complete when RRC is not used and no errors have occurred during the execution of the job. When RRC is employed, a number of checkpoints are taken, n_c. Having n_c checkpoints, implies n_c execution segments and each segment is of length of $\frac{T}{n_c}$.

Next, we elaborate on the occurrence of soft errors. We assume that occurrence of soft errors is an independent event. In our work, given is the probability, P_T, that no errors occur in a processor within an interval equal to the processing time of the job, T. Due to the fact that the occurrence of soft errors is an independent event, we calculate P_ϵ, the probability of successful execution segment, with the following expression:

$$P_\epsilon = \sqrt[n_c]{P_T} \cdot \sqrt[n_c]{P_T} = \sqrt[n_c]{{P_T}^2} \tag{1}$$

Eq. 1 takes into account that no errors occur within an interval of length $\frac{T}{n_c}$ in both processors.

III. Evaluation of level of confidence

In this section we provide analysis and derive an expression to evaluate the level of confidence that a job that employs RRC meets a given deadline. The level of confidence, with respect to a given deadline D, is the probability that a job completes before D. The level of confidence is determined as the sum of intermediate terms that represent the probability that a job completes at a given discrete point in time. These terms are calculated according to a probability distribution function. Thus, to compute the level of confidence we need to derive an expression for the probability distribution function.

To derive the probability distribution function, we start by analyzing the expected completion time when RRC is employed. The expected completion time can be described by a discrete variable due to the fact that an integer number of execution segments (each followed by a checkpointing overhead) must be executed before a job completes. Assuming that n_c checkpoints are to be taken, a job can complete only when n_c successful execution segments have been executed. Thus, in the best case scenario, when no errors have occurred, a job completes after n_c executions segments. Each execution segment is of length $\frac{T}{n_c}$ plus the checkpointing overhead, τ. We denote the case when zero erroneous execution segments are executed with t_0 and it is defined as:

$$t_0 = n_c \cdot (\frac{T}{n_c} + \tau) = T + n_c \cdot \tau \tag{2}$$

If errors occur, and these errors only affect *one* execution segment, this segment will be re-executed. There will be $n_c + 1$ execution segments executed (one erroneous and n_c successful execution segments). We denote the case when one execution segment is re-executed with t_1 and it is defined as:

$$t_1 = (n_c + 1) \cdot (\frac{T}{n_c} + \tau) = T + n_c \cdot \tau + (\frac{T}{n_c} + \tau) \tag{3}$$

In general, when there are k erroneous execution segments, t_k denotes the expected completion time which is defined as:

$$t_k = T + n_c \cdot \tau + k \cdot (\frac{T}{n_c} + \tau) \tag{4}$$

Next, we analyze the number of cases that a job completes exactly at time t_k. First, let us study the case that a job completes at time t_0. This can happen if and only if all the execution segments were successful, that is no errors have occurred. This is the only possible alternative for a job to complete at time t_0. Now, let us assume that a job completes at time t_1. If a job completes at time t_1, a single execution segment has been re-executed. This can be any of the n_c different execution segments. Thus, there are n_c possible cases that a job completes at time t_1. If a job completes at time t_2, two execution segments have been re-executed. It can either be that two out of all n_c different execution segments were re-executed, or a single execution segment was re-executed twice (an error was detected after the first re-execution). In general, if a job completes at time t_k, a total of $n_c + k$ execution segments have been executed, that is n_c successful execution segments and k erroneous execution segments. Note that the last execution segment among all $n_c + k$ execution segments must have been a successful execution segments otherwise it contradicts the assumption that the job has completed at t_k. Hence, the k erroneous execution segments are any of the $n_c + k - 1$ (any execution segment except for the last one). Therefore, the number of different cases that exists such that a job completes at time t_k is the number of all the combinations of k execution segments out of $n_c + k - 1$ execution segments. $N(t_k)$ denotes the number of possible cases that a job completes at time t_k, and $N(t_k)$ is defined as:

$$N(t_k) = \binom{n_c + k - 1}{k} \tag{5}$$

In Figure 2(a) we illustrate $N(t_k)$ (see Eq. 5) for $n_c = 3$ and $t_k \in [t_0, t_5]$. For example, $N(t_1) = 3$ shows that there are three cases that a job completes at t_1, since any one of the three execution segments ($n_c = 3$) could have been re-executed.

Next, to calculate the probability that a job completes at time t_k, we need a probability metric for each case (t_k). This probability metric is closely related to the probability of successful execution segment, P_ϵ (Eq. 1). When a job completes at time t_k, $n_c + k$ execution segments were executed, n_c *successful* and k *erroneous* execution segments. Since P_ϵ represents the probability of successful execution segment, the probability of erroneous execution segment is evaluated as $(1 - P_\epsilon)$. Since execution segments are independent, the probability of having n_c successful execution segments is $P_\epsilon^{n_c}$, and the probability of

having k erroneous execution segments is $(1-P_\epsilon)^k$. Combining these two probabilities, probability of n_c successful and k erroneous execution segments, results in $P_\epsilon^{n_c}\cdot(1-P_\epsilon)^k$, which is the probability metric per possible case when a job completes at time t_k. In Figure 2(b), we illustrate the probability metric per possible case, $P_\epsilon^{n_c}\cdot(1-P_\epsilon)^k$, for $n_c = 3$, $P_T = 0.5$ and $t_k \in [t_0, t_5]$. From Figure 2(b) it can be observed that the probability metric, $P_\epsilon^{n_c}\cdot(1-P_\epsilon)^k$, has the highest value at t_0 and it is evaluated as $P_\epsilon^{n_c} = \left(\sqrt[n_c]{P_T{}^2}\right)^{n_c} = {P_T}^2 = 0.25$. The probability metric per case, $P_\epsilon^{n_c}\cdot(1-P_\epsilon)^k$, drops rapidly by increasing t_k.

To calculate the probability that a job completes at time t_k, we need to multiply the number of possible cases, $N(t_k)$, with the probability metric per case. We denote the probability that a job completes at time t_k with $p(t_k)$, and it is defined as

$$p(t_k) = N(t_k)\cdot P_\epsilon^{n_c}\cdot(1-P_\epsilon)^k = \binom{n_c+k-1}{k}\cdot P_\epsilon^{n_c}\cdot(1-P_\epsilon)^k \quad (6)$$

Eq.6 defines the probability distribution function. In Figure 2(c) we illustrate the probability distribution function for $n_c = 3$, $P_T = 0.5$, and $t_k \in [t_0, t_5]$.

To compute the level of confidence it is required to sum all terms from the probability distribution function $p(t_k)$ for which the discrete variable t_k has a value which is lower or equal to the given deadline, D. We denote the level of confidence of meeting the deadline, D, with $\Lambda(D)$, and it is computed as:

$$\Lambda(D) = \sum_{k=0}^{t_k \le D} p(t_k) = \sum_{k=0}^{t_k \le D}\binom{n_c+k-1}{k}\cdot P_\epsilon^{n_c}\cdot(1-P_\epsilon)^k \quad (7)$$

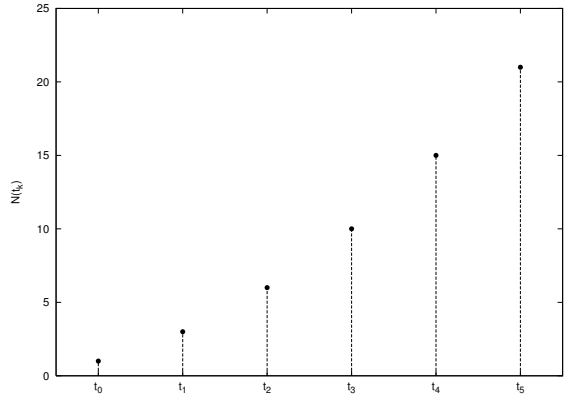
IV. Optimization method

In this section we propose an optimization method for RRC where the optimization goal is to find an optimal number of checkpoints for a job such that minimal completion time is reached under the constraint that the minimal completion time is guaranteed with a given level of confidence requirement.

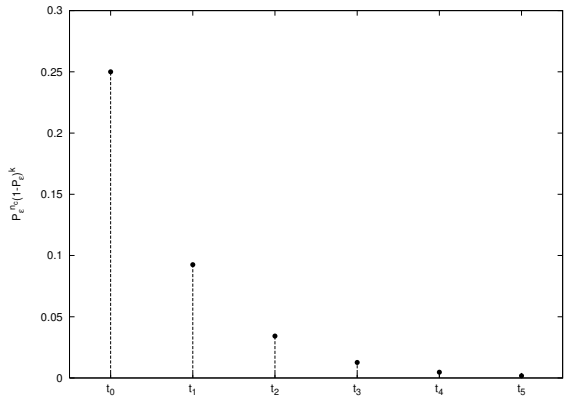
We introduce the term *guaranteed completion time*, GTC_δ, which is a completion time that is guaranteed with a given level of confidence, δ. For the guaranteed completion time, the following expression holds:

$$\Lambda(GCT_\delta) \ge \delta \quad (8)$$

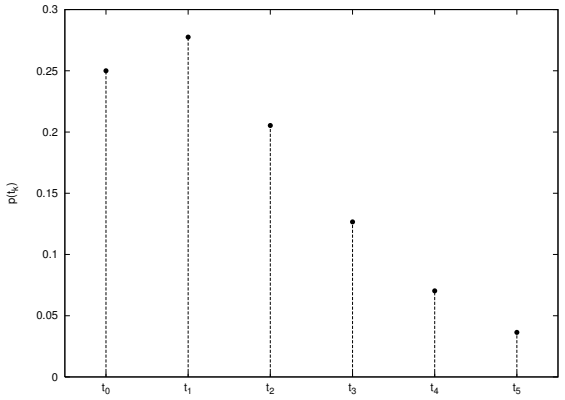
GCT_δ depends on the number of checkpoints, n_c, and there exists an optimal number of checkpoints, n_c^*, that leads to the minimal GCT_δ. To demonstrate that there exist an n_c^*, we consider the following scenario: given is a job with processing time $T = 1000$t.u.(time units), a checkpointing overhead $\tau = 20$t.u., a probability that no errors will occur in the processors in interval of time equal to the processing time $P_T = 0.99999$ and a required level of confidence $\delta = 1-10^{-10}$. In Figure 3 we plot GCT_δ for different number of checkpoints n_c. For instance, when the number of checkpoints is $n_c = 1$, GCT_δ is 3060t.u., and it includes a safe margin for two re-executions. As can be seen from the Figure 3, increasing the number of checkpoints, up till a certain point ($n_c = 10$), results in a decrease of GCT_δ. However, by increasing the number of checkpoints further, GCT_δ starts to increase. Thus, it becomes



(a) Number of cases N(t_k)



(b) Probability metric per case $P_\epsilon^{n_c}(1-P_\epsilon)^k$



(c) Probability distribution function $p(t_k)$

Figure 2: Illustration of functions for $n_c = 3$ and $P_T = 0.5$

important to find the optimal number of checkpoints that leads to the minimal GCT_δ.

Next, we present how to determine the optimal number of checkpoints, n_c^*. We start from Eq. 4 that represents the completion time. As one can observe from Eq. 4, the completion time depends on the number of checkpoints (n_c) and the number of erroneous execution segments (k). Depending on the number of erroneous execution segments (k) a job can complete only at discrete instances in time (t_k), and the distance between two subsequent instances depends on the number of checkpoints (n_c), *i.e.* $t_k - t_{k-1} = \frac{T}{n_c} + \tau$. This is illustrated in Figure 4, *e.g.* the distance between t_1 and t_0 for

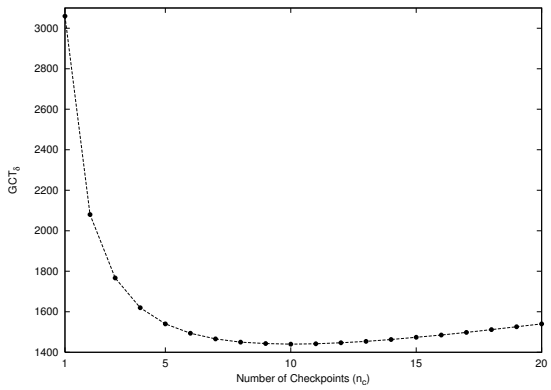


Figure 3: GCT_δ dependence on n_c, for given $\delta = 1 - 10^{-10}$, T = 1000t.u., τ = 20t.u. and $P_T = 0.99999$

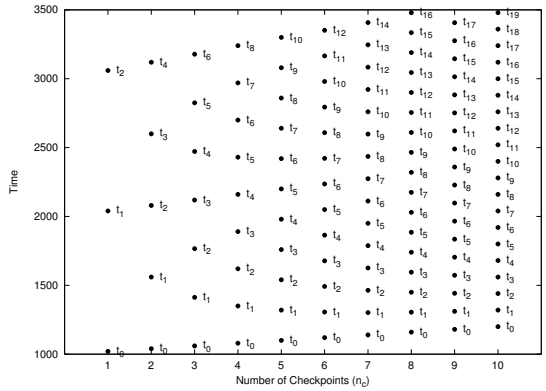


Figure 4: Dependence of completion time(t_k) on number of checkpoints (n_c) for T = 1000t.u. and τ = 20t.u.

$n_c = 1$ is larger than the distance between t_1 and t_0 for $n_c = 10$. Also illustrated in Figure 4, is that when the completion time does not include execution of erroneous execution segments, *i.e.* $k = 0$, the completion time is increasing linearly with the number of checkpoints n_c (observe t_0 at different number of checkpoints in Figure 4). When the completion time includes at least one execution of erroneous execution segment, *i.e.* $k \geq 1$, the completion time decreases by increasing the number of checkpoints up till a certain point. However, increasing the number of checkpoints further results in an increase in the completion time. This is also presented in Figure 4. If we focus on the values of t_1 in Figure 4, we observe that t_1 has the highest value (2040t.u.) at $n_c = 1$, but t_1 decreases as the number of checkpoints is increased up to $n_c = 7$ where the value for t_1 is 1302t.u. Increasing n_c above 7, results in a higher value for t_1, *e.g.* the value for t_1 at $n_c = 10$ is 1320t.u. From Eq. 4, one can observe that for a given n_c, the completion time depends linearly on the number of erroneous execution segments (k), and thus obtaining a low completion time requires a low value for k. On the other hand, for a given number of erroneous execution segments (k), there exists an optimal number of checkpoints for which the completion time is minimal. To find the optimal number of checkpoints, we take the first derivative of Eq. 4 and set it to be equal to zero, hence we have:

$$\frac{\partial t_k}{\partial n_c} = 0 \Rightarrow n_c = \sqrt{k \cdot \frac{T}{\tau}} \tag{9}$$

As can be observed from Eq. 9, the optimal number of checkpoints depends on k. Since each erroneous execution segment requires a re-execution, we define k as the number of re-executions to be included in the completion time. This parameter k is tightly related to the given level of confidence requirement, δ. To satisfy δ, GCT_δ must include a safe margin allowing a number of re-executions k. To find the minimal GCT_δ, requires finding the lowest value of k and the optimal number of checkpoints. Note that a higher value of k would also meet the required level of confidence, δ, but then GCT_δ would be unnecessarily high.

The optimization method works as follow. We first set $k = 1$, and then compute the number of checkpoints n_c using Eq. 9. Next, the level of confidence is calculated by summing the terms of the probability distribution function for $k = 0$ and $k = 1$, while using the recently calculated value for n_c (see Eq.7 for evaluation of level of confidence). If the computed level of confidence is higher than the given level of confidence requirement, δ, the calculated n_c is reported to be the optimal number of checkpoints, otherwise a new iteration follows. In each iteration k is incremented, the value of n_c is updated (Eq. 9), and the level of confidence is computed for the new values of k and n_c. The iteration terminates when the required level of confidence is reached. Then the recent calculated n_c is reported to be the optimal number of checkpoints, n_c^*, which leads to the minimal $GCT_\delta^* = T + n_c^*\tau + k(\frac{T}{n_c^*} + \tau)$.

Next, we demonstrate the method for the example introduced earlier in this section. Setting $k = 1$, leads to $n_c = \sqrt{k \cdot \frac{T}{\tau}} = 7$. Hence, we compute the level of confidence for $n_c = 7$ by summing the first two terms from the probability distribution function (for $k = 0$ and $k = 1$). This results in a level of confidence value, $\Lambda(t_1)$, that is lower than the given level of confidence requirement, δ, so we continue with another iteration. In this iteration, k is incremented, ($k = 2$), which implies $n_c = 10$. Next, we compute the level of confidence for $n_c = 10$ by summing the terms from the probability distribution function for $0 \leq k \leq 2$. The obtained result satisfies the given requirement δ, and therefore we report the optimal number of checkpoints to be $n_c^* = 10$, which leads to minimal $GCT_\delta^* = 1000 + 10 \cdot 20 + 2 \cdot (\frac{1000}{10} + 20) = 1440$t.u.

V. Results

In this section we present results for these three problems:

- P1: evaluation of the level of confidence with respect to a given deadline D;
- P2: finding an optimal number of checkpoints, n_c^*, that minimizes the guaranteed completion time, GCT_δ, for a given level of confidence requirement δ, and
- P3: evaluation of probabilistic guarantees for RRC optimization for non-real-time systems.

For each problem we use two input scenarios, Scenario A and Scenario B, (Table I). For each scenario, the following inputs are given:T, τ, and P_T.

For P1, we assume given is a deadline D = 1500t.u. The results in Table II and Table III show the computed level of

Scenario A	Scenario B
T = 1000t.u.	T = 1000t.u.
τ = 20t.u.	τ = 20t.u.
P_T = 0.99999	P_T = 0.9

Table I: Input Scenarios

confidence, $\Lambda(D)$, at different number of checkpoints, n_c. For each n_c, we first calculate K, the number of re-executions that can be accommodated within the interval $[t_0, D]$, and then we sum all terms from the probability distribution function (Eq. 6) for $t_k \in [t_0, t_K]$. As can be seen from Table II and Table III, the level of confidence, $\Lambda(D)$, for meeting a given deadline, D, depends on the number of checkpoints, n_c. When the number of checkpoints is low, the level of confidence is also low. The level of confidence increases as the number of checkpoints increases. However, at a certain number of checkpoints, increasing the number of checkpoints further results in decreased level of confidence or even leads to a zero level of confidence. The reason is that when the number of checkpoints is low, the execution segments are longer, which means that it is difficult to accommodate many re-executions while meeting the deadline. This implies that only a small number of terms from the probability distribution function (Eq. 6) will be summed and therefore the level of confidence (Eq. 7) is low. Increasing the number of checkpoints, decreases the length of the execution segments and thus allows more re-executions to be accommodated before the deadline on one hand, but increases the total checkpointing overhead on the other hand. Having a high number of checkpoints may result in a zero level of confidence. As t_0, the case when zero erroneous execution segments are executed, depends on the number of checkpoints, n_c, (Eq. 2), having a high number of checkpoints may result in that t_0 violates the deadline D, *i.e.* $t_0 > D$. For example, for the given input scenarios when $n_c = 26$, $t_0 = 1000 + 26 \cdot 20 = 1520$ and the level of confidence $\Lambda(D) = 0$, (see Table II and Table III). With the results obtained from solving P1, we want to point out that it is useful to have a framework to calculate the level of confidence because it makes it possible to optimize the RRC scheme such that an optimal number of checkpoints that results in the highest level of confidence can be obtained. From the presented results in Table II and Table III, we note that the number of checkpoints that provides the highest level of confidence $\Lambda(D)$ is $n_c = 17$ for both Scenario A and Scenario B. However, $\Lambda(D)$ for Scenario A is much higher than $\Lambda(D)$ for Scenario B due to the different P_T values.

For P2, we assume given is a level of confidence requirement $\delta = 1 - 10^{-10}$. We compute the guaranteed completion time, GCT_δ, for different number of checkpoints, n_c. The results are presented in Table IV and Table V. For each n_c, we initialize $k = 1$, and check the following inequality $\Lambda(t_1) \geq \delta$. If the inequality is not satisfied, then k is incremented until $\Lambda(t_k) \geq \delta$. This implies that for the last value of k, we get a t_k that is a guaranteed completion time (GCT_δ) for the given n_c. In Table IV and Table V for each n_c we present the values for GCT_δ and the number of re-executions, k, that are included in GCT_δ. As can be seen from Table IV and Table V, having a

D = 1500			
n_c	$\Lambda(D)$	n_c	$\Lambda(D)$
1	0.999980000100000000	14	0.999999999999998367
2	0.999980000100000000	15	0.999999999999998388
3	0.999999999733334814	16	0.999999999999998406
4	0.999999999750001250	**17**	**0.999999999999998422**
5	0.999999999760001120	18	0.999999999788889670
6	0.999999999999997925	19	0.999999999789474459
7	0.999999999999998040	20	0.999999999790000770
8	0.999999999999998125	21	0.999999999790476955
9	0.999999999999998189	22	0.999980000100000000
10	0.999999999999998240	23	0.999980000100000000
11	0.999999999999998280	24	0.999980000100000000
12	0.999999999999998314	25	0.999980000100000000
13	0.999999999999998343	26	0

Table II: Level of confidence, $\Lambda(D)$, for Scenario A, at various number of checkpoints, n_c

D = 1500			
n_c	$\Lambda(D)$	n_c	$\Lambda(D)$
1	0.810000000000000000	14	0.998386333221060871
2	0.810000000000000000	15	0.998405709197021325
3	0.974827503159636872	16	0.998422589149847735
4	0.976266114316335439	**17**	**0.998437425722750770**
5	0.977137362167560214	18	0.979688847172390437
6	0.997980204415657095	19	0.979741032210778210
7	0.998085015474654920	20	0.979788017059326005
8	0.998162202793752259	21	0.979830542116846522
9	0.998221387037794418	22	0.810000000000000000
10	0.998268194669895683	23	0.810000000000000000
11	0.998306132813719019	24	0.810000000000000000
12	0.998337499909652013	25	0.810000000000000000
13	0.998363864473716882	26	0

Table III: Level of confidence, $\Lambda(D)$, for Scenario B, at various number of checkpoints, n_c

$\delta = 1 - 10^{-10}$					
n_c	k	GCT_δ	n_c	k	GCT_δ
1	2	3060	11	2	1442
2	2	2080	12	2	1447
3	2	1767	13	2	1454
4	2	1620	14	2	1463
5	2	1540	15	2	1474
6	2	1494	16	2	1485
7	2	1466	17	2	1498
8	2	1450	18	2	1512
9	2	1443	19	2	1526
10	**2**	**1440**	20	2	1540

Table IV: GCT_δ and the number of re-executions, k, included in GCT_δ for Scenario A, at various n_c

$\delta = 1 - 10^{-10}$					
n_c	k	GCT_δ	n_c	k	GCT_δ
1	13	14280	12	8	2066
2	11	6760	13	8	2036
3	10	4594	14	8	2012
4	9	3510	15	8	1994
5	9	3080	16	8	1980
6	9	2800	17	8	1971
7	8	2443	18	8	1965
8	8	2320	19	8	1962
9	8	2229	**20**	**8**	**1960**
10	8	2160	21	8	1961
11	8	2108	22	8	1964

Table V: GCT_δ and number of re-executions, k, included in GCT_δ for Scenario B, at various n_c

low number of checkpoints leads to high GCT_δ. Increasing the number of checkpoints up till a certain point, decreases GCT_δ. However, by increasing the number of checkpoints further we observe an increase in the guaranteed completion time. We explain this behavior with the following reasoning. To satisfy the given level of confidence requirement, GCT_δ must include some safe margin, *i.e.* some number of re-executions should be allowed. When the number of checkpoints is low, the execution segments are large and therefore the re-executions are more expensive in time, which leads to high GCT_δ. By increasing the number of checkpoints, the execution segments become shorter and this leads to a decrease in GCT_δ. However, when the number of checkpoints becomes sufficiently high, the checkpointing overhead becomes the dominant part of GCT_δ and it is the checkpointing overhead that is responsible for the observed increase in GCT_δ. From Table IV we observe that the minimal GCT_δ is 1440t.u. and it is achieved when $n_c = 10$ for Scenario A, which adheres to the results that we obtained from the proposed optimization method presented in Section IV. In Table V we observe that for Scenario B, the minimal GCT_δ is 1960t.u. with $n_c = 20$. The same result is obtained when the presented optimization method is used. The results presented in Table IV and Table V are acquired by running all combinations of values for n_c and k, while the results from the optimization method only require two iterations for Scenario A, and eight iterations for Scenario B (observe the number of re-executions, k, in Table IV and Table V for $n_c = 10$ and $n_c = 20$ respectively).

For P3, we consider an RRC optimization approach for a non-real-time system that provides an optimal number of checkpoints, n_c^*, that leads to minimal average execution time (AET), [4]. For Scenario A, the approach, [4], computes $n_c^* = 1$ and a minimal AET = 1020t.u., while for Scenario B, the approach computes $n_c^* = 3$ and a minimal AET = 1138t.u. There are two interesting problems when evaluating probabilistic guarantees for RRC optimization for non-real-time systems, and thus we divide P3 into two subproblems:

- P3A: evaluation of the level of confidence with respect to the minimal AET,
- P3B: evaluation of the level of confidence with respect to a given deadline D, when n_c is optimized towards AET

For P3A, *i.e.* evaluation of $\Lambda(AET)$, we present the level of confidence that a job completes within an interval that is equal to the minimal average AET, while assuming that optimal number of checkpoints (n_c^*) are used. By computing the level of confidence for the calculated minimal AET, we observe that $\Lambda(1020) = 0.99998$ for Scenario A, and $\Lambda(1138) = 0.81$ for Scenario B, which may be acceptable for non-real-time system, but not for a real-time system where a high level of confidence is required.

For P3B, *i.e.* evaluation of $\Lambda(D)$ when n_c is optimized towards AET, we assume given is a deadline D = 1500t.u. As shown earlier, the optimization approach, [4], computed $n_c^* = 1$, for Scenario A, and $n_c^* = 3$, for Scenario B. Relying on this optimization implies the following results:

- for Scenario A, $\Lambda(D) = 0.99998$ (see Table II for $n_c = 1$)
- for Scenario B, $\Lambda(D) \leq 0.975$ (see Table III for $n_c = 3$).

However, we observed earlier (see Table II and Table III) that the highest level of confidence that can be achieved is:

- for Scenario A, $\Lambda(D) \geq 0.999999999999998$ for $n_c = 17$
- for Scenario B, $\Lambda(D) \geq 0.99843$ for $n_c = 17$.

From presented results for P3 (P3A and P3B), we conclude that relying on RRC optimization for non-real-time systems results in poor probabilistic guarantees.

VI. Conclusion

In this paper we have focused on analyzing RRC in the real-time system scenario. There are three main contributions that we have presented in the paper.

First, we presented a mathematical framework to evaluate the level of confidence that a job employing RRC meets a given deadline. This mathematical framework is important not only for computing the level of confidence and thus getting a reliability metric, but also it is useful to acquire knowledge on how to adjust the RRC scheme, *i.e.* adjust the number of checkpoints such that the level of confidence is maximized.

Second, we presented an optimization method where the optimization goal was to find the optimal number of checkpoints that minimizes the completion time, while with a given level of confidence requirement we can guarantee that the job will complete within this minimal completion time.

Third, by using the proposed mathematical framework, we evaluated probabilistic guarantees for non-real-time RRC optimization.We have shown that having the minimal AET is not a sufficient guarantee and as such is not very useful in the real-time scenario. Further, we demonstrated that relying on RRC optimization for non-real-time systems can significantly reduce the level of confidence for meeting a given deadline.

References

[1] D. Nikolov, U. Ingelsson, V. Singh, and E. Larsson, "Estimating Error-probability and its Application for Optimizing Roll-back Recovery with Checkpointing", delta, pp.281-285, 2010 Fifth IEEE International Symposium on Electronic Design, Test & Applications, 2010.

[2] A. Ziv and J. Bruck, "Analysis of Checkpointing Schemes with Task Duplication", *IEEE Trans. on computers*, vol. 47, no.2, February 1998.

[3] A. Ziv and J. Bruck, "An On-Line Algorithm for Checkpoint Placement", *IEEE Trans. on computers*, vol. 46, no.9, September 1997.

[4] 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.

[5] S. Punnekkat, A. Burns, and R. Davis "Analysis of Checkpointing for Real-Time Systems",*The International Journal of Time-Critical Computing Systems*,20, pp.83-102, 2001.

[6] Y. Zhang and K. Chakrabarty, "Fault Recovery Based on Checkpointing for Hard Real-Time Embedded Systems", *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'03)*, 2003.

[7] K. G. Shin, T. Lin, and Y. Lee, "Optimal Checkpointing of Real-Time Tasks", *IEEE Trans. on computers*, vol. C-36, no.11, November 1987.

[8] V. Grassi, L. Donatiello and S. Tucci, "On the Optimal Checkpointing of Critical Tasks and Transaction-Oriented Systems", *IEEE Trans. on software engineering*, vol. 18, no.1, January 1992.

[9] S. W. Kwak, B. J. Choi, and B. K. Kim Ling, "An Optimal Checkpointing-Strategy for Real-Time Control Systems Under Transient Faults", *IEEE Trans. on reliability*, vol. 50, no.3, September 2001.

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