A different re-execution speed can help

Anne Benoit, Aurélien Cavelan, Valentin Le Fèvre, Yves Robert, Hongyang Sun

A different re-execution speed can help

Anne Benoit*, Aurélien Cavelan*, Valentin Le Fèvre*, Yves Robert*†, Hongyang Sun*
*Ecole Normale Supérieure de Lyon & Inria, France
†University of Tennessee Knoxville, USA

***Abstract*—We consider divisible load scientific applications executing on large-scale platforms subject to silent errors. While the goal is usually to complete the execution as fast as possible in expectation, another major concern is energy consumption. The use of dynamic voltage and frequency scaling (DVFS) can help save energy, but at the price of performance degradation. Consider the execution model where a set of K different speeds is given, and whenever a failure occurs, a different re-execution speed may be used. Can this help? We address the following bi-criteria problem: how to compute the optimal checkpointing period to minimize energy consumption while bounding the degradation in performance. We solve this bi-criteria problem by providing a closed-form solution for the checkpointing period, and demonstrate via a comprehensive set of simulations that a different re-execution speed can indeed help.**

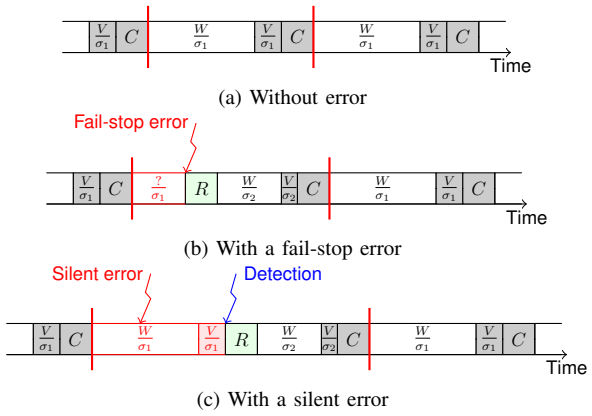
***Index Terms*—resilience, silent errors, speeds, re-execution, checkpointing, verification.**

I. Introduction

This paper investigates whether changing speed for re-execution, after a silent error has been detected, can help save energy while matching a prescribed performance bound. We revisit the well-known formula by Young [19] and Daly [10] to compute the optimal periodic checkpointing interval. We extend their formula to this bi-criteria energy/performance problem, where the application can be executed, or re-executed, using a set of K different speeds.

The motivation for this work is twofold. The first concern is that large-scale platforms are increasingly subject to errors. The frequent striking of silent errors (or SDCs, for Silent Data Corruptions) has been recently identified as one of the major challenges for exascale [8]. There are several causes of silent errors, such as cosmic radiation, packaging pollution, among others. In contrast to a fail-stop error whose detection is immediate, a silent error is identified only when the corrupted data leads to an unusual application behavior. Such a detection latency raises a new challenge: if the error struck before the last checkpoint, and is detected after that checkpoint, then the checkpoint is corrupted and cannot be used for rollback. In order to avoid corrupted checkpoints, an effective approach consists in employing some verification mechanism and combining it with checkpointing [5], [4]. This verification mechanism can be general-purpose (e.g., based on replication [11] or even triplication [13]) or application-specific (e.g., based on Algorithm-based fault tolerance (ABFT) [7], on approximate re-execution for ODE and PDE solvers [6], or on orthogonality checks for Krylov-based sparse solvers [9], [17]).

We address silent errors by taking *verified checkpoints*, which correspond to performing a verification just before each checkpoint. Note that this approach is agnostic of the nature of the verification mechanism. If the verification succeeds, then one can safely store the checkpoint. If the verification fails, it means that a silent error has struck since the last checkpoint, which was duly verified, and one can safely recover from that checkpoint to resume the execution of the application. It is not difficult to compute the optimal checkpointing period T with silent errors and verified checkpoints. For fail-stop errors, the Young/Daly formula writes $T = \sqrt{2C/\lambda}$, where C is the time to checkpoint and λ the error rate (hence the platform MTBF is $\mu = 1/\lambda$). For silent errors, the formula becomes $T = \sqrt{(V+C)/\lambda}$, where V is the time to verify. We simply replace C by $V + C$ in the formula, and the missing factor 2 can be explained as follows (see Figure 1): fail-stop errors are detected, on average, at half the period, while silent errors are always detected at the end of the period (by the verification mechanism). For a given T, the expected re-execution time is $T/2$ for fail-stop errors and T for silent errors, so the optimal period is shorter for silent errors (see [12] for details).



(a) Without error

(b) With a fail-stop error

(c) With a silent error

Figure 1. A periodic checkpointing pattern (highlighted in red) first executed at speed σ_1 and re-executed at speed $\sigma_2 = 2\sigma_1$ in case of error. The first figure shows the execution without any error, so only the first speed is used. The second figure shows that the execution is stopped immediately when a fail-stop error strikes, in which case the pattern is re-executed at speed σ_2 after a recovery. The third figure shows that the execution continues after a silent error strikes, and it is detected by the verification at the end of the pattern. In this case, the pattern is also re-executed at speed σ_2 after a recovery.

The second concern is energy consumption. The power requirement of current petascale platforms is that of a small town, hence measures must be taken to reduce the energy consumption of future platforms [8]. A popular technique is dynamic voltage and frequency scaling (DVFS): modern processors can run at different speeds, and lower speeds induce

bigger savings in energy consumption. In a nutshell, this is because the dynamic power consumed when computing at speed σ is proportional to σ^3 [18], [2], while execution time is proportional to $1/\sigma$. As a result, computing energy (which is the product of time and power) is proportional to σ^2. However, static power must also be accounted for, and it is paid throughout the duration of the execution, which calls for a shorter execution (at higher speeds). Overall, there are trade-offs to be found, but in most practical settings, using lower speeds reduces the global energy consumption. Unfortunately, using lower speeds also increases execution time, and a realistic approach calls for minimizing energy while guaranteeing a performance bound.

Altogether, we face a challenging problem: given a platform subject to silent errors striking at rate λ, given the cost C of a checkpoint and the cost V of a verification, and given a set $\mathcal{S} = \{s_1, s_2, \ldots, s_K\}$ of K possible speeds, what is the optimal length of the execution period T? More precisely, because we allow the use of different speeds, what is the optimal amount of work W within a pattern, and what are the optimal speeds $\sigma_1, \sigma_2 \in \mathcal{S}$ to use? Here, we allow the first execution at a speed σ_1 and all following possible re-executions (in case of error) at speed σ_2. As mentioned above, *optimality* is defined as minimizing energy subject to a performance bound ρ, thus the target optimization problem BICRIT is formally stated as

$$\text{MINIMIZE } \frac{\mathcal{E}(W,\sigma_1,\sigma_2)}{W} \text{ S.T. } \frac{\mathcal{T}(W,\sigma_1,\sigma_2)}{W} \leq \rho, \quad (1)$$

where $\mathcal{E}(W,\sigma_1,\sigma_2)$ is the expected energy consumed to execute W units of work at speed σ_1, with eventual re-executions at speed σ_2, thus $\frac{\mathcal{E}(W,\sigma_1,\sigma_2)}{W}$ is the expectation of the energy consumed per work unit. Similarly, $\frac{\mathcal{T}(W,\sigma_1,\sigma_2)}{W}$ is the expected execution time per work unit, and ρ is a performance bound, or admissible degradation factor.

The main contribution of this paper is to solve this difficult optimization problem. We are able to extend the Young/Daly formula and to provide a closed-form analytical expression for the optimal solution, up to first-order approximation. We show that using different speeds can indeed induce significant savings in energy, both through theoretical analysis and via a comprehensive set of simulations conducted with realistic parameters. Finally, we explore how to extend this work to deal with both fail-stop and silent errors by reporting our preliminary findings. An interesting particular case is the following: when the platform is subject to fail-stop errors only, and the re-execution speed is twice the initial speed, the optimal checkpointing period is not in the order of the square root of platform MTBF, a striking novelty!

Due to lack of space, further related work is discussed in the companion research report [3]. We simply point out that this paper is the first (to the best of our knowledge) to tackle the bi-criteria problem for HPC applications using different re-execution speeds (our previous work [1] deals with real-time tasks, which are assumed to be re-executed only once).

The rest of the paper is organized as follows. Section II introduces the model and notations. Section III presents our main contributions and shows how to determine the optimal pattern. Simulation results are provided in Section IV. We consider extensions (with fail-stop errors) in Section V. Finally, Section VI provides concluding remarks and future directions.

II. MODEL

We consider a divisible load application executing on a platform subject to silent errors. The execution is partitioned into periodic patterns that repeat over time. Each pattern consists of a computational chunk of W units of work followed by a verification and a checkpoint (see Figure 1). The size of the chunk W can be freely determined (this is what we mean by a *divisible load* application; we can insert verifications and checkpoints at any time step during the computation).

A. Parameters

The application executes on a large scale platform with the following parameters:

- Silent errors follow an exponential distribution of parameter λ, hence the platform MTBF is $\mu = 1/\lambda$. The probability of an error striking during T time units is then $p(T) = 1 - e^{-\lambda T}$.
- The platform can be operated under a set $\mathcal{S} = \{s_1, s_2, \ldots, s_K\}$ of K possible speeds (each speed is the aggregated speed of all processors in the platform).
- Without errors, the time to execute W units of work at speed $\sigma \in \mathcal{S}$ is $\frac{W}{\sigma}$, and the energy consumed is $\frac{W}{\sigma}(P_{\text{idle}} + P_{\text{cpu}}(\sigma))$. Here, P_{idle} is the static power consumed when the platform is on (even idle), and $P_{\text{cpu}}(\sigma) = \kappa\sigma^3$ is the dynamic power spent by operating the platform at speed σ [18], [2].
- The time to checkpoint is C and the time to recover is R. The corresponding energy consumptions are $C(P_{\text{idle}} + P_{\text{io}})$ and $R(P_{\text{idle}} + P_{\text{io}})$, where P_{io} is the dynamic power spent by I/O transfers.
- The time to perform a verification at speed σ is $\frac{V}{\sigma}$ and the corresponding energy consumption is $\frac{V}{\sigma}(P_{\text{idle}} + P_{\text{cpu}}(\sigma))$.

We assume that all speeds in S are within normal operational range (i.e., without overscaling or underscaling), so the error rate λ is not affected by the choice of the speed.

B. Execution model

We consider the following execution model:

- The first execution is done at speed σ_1, which is freely chosen from $\mathcal{S}$.
- At the end of the pattern, we verify the result. If it is correct, we can safely checkpoint. Otherwise, if the verification detects an error, we recover from the last checkpoint (or from the initial data for the first pattern), and re-execute the pattern. This re-execution, and all the subsequent re-executions if needed till success, are all done at speed σ_2, which is also freely chosen from $\mathcal{S}$.

For a pattern of size W, let $\mathcal{T}(W,\sigma_1,\sigma_2)$ be the expected time to execute the pattern in this model, and let $\mathcal{E}(W,\sigma_1,\sigma_2)$ be the corresponding expected energy consumption.

C. Optimization problem BICRIT

Let W_{base} denote the total amount of work of the application. Also, let $\mathcal{T}_{\text{total}}$ and $\mathcal{E}_{\text{total}}$ denote, respectively, the expected makespan and expected energy consumption of the application when executed using periodic patterns. For a pattern size W, there are approximately $\frac{W_{\text{base}}}{W}$ patterns (for a long-lasting application), with each pattern having an expected execution time of $\mathcal{T}(W, \sigma_1, \sigma_2)$. Hence, we have $\mathcal{T}_{\text{total}} \approx \frac{\mathcal{T}(W,\sigma_1,\sigma_2)}{W} \cdot W_{\text{base}}$ and $\mathcal{E}_{\text{total}} \approx \frac{\mathcal{E}(W,\sigma_1,\sigma_2)}{W} \cdot W_{\text{base}}$.

Minimizing the expected makespan of the application is therefore equivalent to minimizing the time overhead $\frac{\mathcal{T}(W,\sigma_1,\sigma_2)}{W}$ of a single pattern and, similarly, minimizing the expected energy consumption is equivalent to minimizing the energy overhead $\frac{\mathcal{E}(W,\sigma_1,\sigma_2)}{W}$. The optimization problem BICRIT is to minimize the expected energy consumption (or equivalently the energy overhead) of the application subject to a bound on the expected makespan (or equivalently the makespan overhead), as presented formally in Equation (1).

III. Optimal pattern size and speeds

In this section, we compute the optimal pattern size W and speed pair (σ_1, σ_2) for the BICRIT problem. We first compute $\mathcal{T}(W, \sigma, \sigma)$, i.e., the expected time to execute the pattern with a single speed σ.

Proposition 1. *For the* BICRIT *problem with a single speed,*

$$\mathcal{T}(W, \sigma, \sigma) = C + e^{\frac{\lambda W}{\sigma}} \left(\frac{W+V}{\sigma}\right) + \left(e^{\frac{\lambda W}{\sigma}} - 1\right) R.$$

Proof. The recursive equation to compute $\mathcal{T}(W, \sigma, \sigma)$ writes:

$$\mathcal{T}(W, \sigma, \sigma) = \frac{W+V}{\sigma} + p(W/\sigma)\left(R + \mathcal{T}(W, \sigma, \sigma)\right) + (1 - p(W/\sigma))C,$$

where $p(W/\sigma) = 1 - e^{-\frac{\lambda W}{\sigma}}$. The reasoning is as follows:

- We always execute W units of work followed by the verification, in time $\frac{W+V}{\sigma}$;
- With probability $p(W/\sigma)$, a silent error occurred and is detected, in which case we recover and start anew;
- Otherwise, with probability $1 - p(W/\sigma)$, we simply checkpoint after a successful execution.

Solving this equation leads to the expected execution time. □

We are now ready to compute $\mathcal{T}(W, \sigma_1, \sigma_2)$, the expected time to execute the pattern with two speeds.

Proposition 2. *For the* BICRIT *problem,*

$$\mathcal{T}(W, \sigma_1, \sigma_2) = C + \frac{W+V}{\sigma_1} + \left(1 - e^{-\frac{\lambda W}{\sigma_1}}\right) e^{\frac{\lambda W}{\sigma_2}} \left(R + \frac{W+V}{\sigma_2}\right).$$

Proof. With a similar reasoning as in the preceding proof, the recursive equation to compute $\mathcal{T}(W, \sigma_1, \sigma_2)$ writes:

$$\mathcal{T}(W, \sigma_1, \sigma_2) = \frac{W+V}{\sigma_1} + p(W/\sigma_1)\left(R + \mathcal{T}(W, \sigma_2, \sigma_2)\right) + (1 - p(W/\sigma_1))C,$$

where $p(W/\sigma_1) = 1 - e^{-\frac{\lambda W}{\sigma_1}}$. Plugging back the expression of $\mathcal{T}(W, \sigma_2, \sigma_2)$ from Proposition 2 and rearranging terms, we readily obtain the expression of $\mathcal{T}(W, \sigma_1, \sigma_2)$ as claimed. □

The following shows the expected energy consumption to execute the pattern with two speeds.

Proposition 3. *For the* BICRIT *problem,*

$$\begin{aligned}\mathcal{E}(W, \sigma_1, \sigma_2) = &\left(C + \left(1 - e^{-\frac{\lambda W}{\sigma_1}}\right) e^{\frac{\lambda W}{\sigma_2}} R\right)(P_{\text{io}} + P_{\text{idle}}) \\ &+ \frac{W+V}{\sigma_1}(\kappa\sigma_1^3 + P_{\text{idle}}) \\ &+ \frac{W+V}{\sigma_2}(1 - e^{-\frac{\lambda W}{\sigma_1}}) e^{\frac{\lambda W}{\sigma_2}}(\kappa\sigma_2^3 + P_{\text{idle}}).\end{aligned}$$

Proof. The power spent during checkpoint or recovery is $P_{\text{io}} + P_{\text{idle}}$, and that spent during computation and verification at speed σ is $P_{\text{cpu}}(\sigma) + P_{\text{idle}} = \kappa\sigma^3 + P_{\text{idle}}$. Hence, from Proposition 2, we get the expression of $\mathcal{E}(W, \sigma_1, \sigma_2)$. □

Formally, the BICRIT problem writes:

$$\min_{\sigma_1,\sigma_2} \min_{W} \frac{\mathcal{E}(W, \sigma_1, \sigma_2)}{W} \text{ s.t. } \frac{\mathcal{T}(W, \sigma_1, \sigma_2)}{W} \leq \rho.$$

In order to get a closed-form expression for the optimal value of W, we first derive the following first-order approximation formulas, using Taylor expansion $e^{\lambda W} = 1 + \lambda W + O(\lambda^2 W^2)$:

$$\begin{aligned}\frac{\mathcal{T}(W, \sigma_1, \sigma_2)}{W} &= \frac{1}{\sigma_1} + \frac{\lambda W}{\sigma_1\sigma_2} + \frac{\lambda R}{\sigma_1} + \frac{\lambda V}{\sigma_1\sigma_2} \\ &\quad + \frac{C + V/\sigma_1}{W} + O(\lambda^2 W). \qquad (2)\\ \frac{\mathcal{E}(W, \sigma_1, \sigma_2)}{W} &= \frac{\kappa\sigma_1^3 + P_{\text{idle}}}{\sigma_1} + \frac{\lambda W}{\sigma_1\sigma_2}(\kappa\sigma_2^3 + P_{\text{idle}}) \\ &\quad + \frac{\lambda R}{\sigma_1}(P_{\text{io}} + P_{\text{idle}}) + \frac{\lambda V}{\sigma_1\sigma_2}(\kappa\sigma_1^3 + P_{\text{idle}}) \\ &\quad + \frac{C(P_{\text{io}} + P_{\text{idle}}) + V(\kappa\sigma_1^3 + P_{\text{idle}})/\sigma_1}{W} + O(\lambda^2 W). \qquad (3)\end{aligned}$$

Both Equations (2) and (3) are of the form $x + \frac{y}{W} + zW + O(\lambda^2 W)$, where x, y and z are positive constants. Such an expression is minimized when $W = \sqrt{\frac{y}{z}} = \Theta(\lambda^{-1/2})$, a result similar to the Young/Daly formula and which shows the accuracy of the first-order approximation via Taylor expansion, since $\lambda W = \Theta(\lambda^{1/2})$ tends to 0 with λ. Here, because we consider Equations (2) and (3) simultaneously, we have to resort to a case study to determine the optimal solution.

Theorem 1. *Given σ_1, σ_2 and ρ, consider the equation $aW^2 + bW + c = 0$, where $a = \frac{\lambda}{\sigma_1\sigma_2}$, $b = \frac{1}{\sigma_1} + \lambda\left(\frac{R}{\sigma_1} + \frac{V}{\sigma_1\sigma_2}\right) - \rho$ and $c = C + \frac{V}{\sigma_1}$.*

- *If there is no positive solution to the equation, i.e., $b > -2\sqrt{ac}$, then* BICRIT *has no solution.*
- *Otherwise, let W_1 and W_2 be the two solutions of the equation with $W_1 \leq W_2$ (at least W_2 is positive and possibly $W_1 = W_2$). Then, the optimal pattern size is*

$$W_{\text{opt}} = \min(\max(W_1, W_e), W_2), \qquad (4)$$

$$\text{where } W_e = \sqrt{\frac{C(P_{\text{io}} + P_{\text{idle}}) + \frac{V}{\sigma_1}(\kappa\sigma_1^3 + P_{\text{idle}})}{\frac{\lambda}{\sigma_1\sigma_2}(\kappa\sigma_2^3 + P_{\text{idle}})}}. \quad (5)$$

Proof. Neglecting lower-order terms, Equation (3) is minimized when $W = W_e$ given by Equation (5). However, this minimum value may well lead to a time overhead exceeding ρ. Enforcing $\frac{\mathcal{T}(W,\sigma_1,\sigma_2)}{W} \leq \rho$ is equivalent to having $aW^2 + bW + c \leq 0$, where a, b and c are given in Theorem 1. Either this latter expression has no positive solution, in which case BICRIT has no solution, or W must lie within the interval $\left[W_1 = \frac{-b-\sqrt{b^2-4ac}}{2a}; W_2 = \frac{-b+\sqrt{b^2-4ac}}{2a}\right]$ (where at least W_2 is positive). Because the energy overhead is a convex function of W, we deduce that $W_{\text{opt}} = \min(\max(W_1, W_e), W_2))$. □

For any speed pair (s_i, s_j), with $1 \leq i, j \leq K$, we define $\rho_{i,j}$ to be the minimum performance bound for which the BICRIT problem with $\sigma_1 = s_i$ and $\sigma_2 = s_j$ admits a solution. From the analysis of Theorem 1, we get that $\rho_{i,j}$ is obtained when $b = -2\sqrt{ac}$, which leads to

$$\rho_{i,j} = \frac{1}{s_i} + 2\sqrt{\left(C + \frac{V}{s_i}\right)\frac{\lambda}{s_i s_j}} + \lambda\left(\frac{R}{s_i} + \frac{V}{s_i s_j}\right). \quad (6)$$

Hence, to solve the BICRIT problem for any given performance bound ρ, we can use the following simple procedure:

1) For each speed pair (s_i, s_j), compute $\rho_{i,j}$ from Equation (6), and discard those pairs with $\rho < \rho_{i,j}$;
2) For each remaining speed pair (σ_1, σ_2), compute W_{opt} from Equation (4), and the energy overhead $\frac{\mathcal{E}(W_{\text{opt}},\sigma_1,\sigma_2)}{W_{\text{opt}}}$ from Equation (3).
3) Select the best speed pair (σ_1^*, σ_2^*) that minimizes the energy overhead above.

This procedure takes $O(K^2)$ time to run, where K is the total number of available speeds. Since K is usually small (e.g., 5) and can be considered as a constant, the optimal solution to the BICRIT problem can be computed in constant time.

IV. SIMULATIONS

In this section, we conduct simulations to evaluate our model. Section IV-A describes the parameters used in the simulations. Section IV-B investigates whether using a different re-execution speed can indeed help save energy. Section IV-C shows the impact of different parameters on the optimal solution and the associated energy overhead.

A. Simulation setup

This section describes the parameters used in the simulations. First, the recovery time is set to be the same as the checkpointing time, i.e., $R = C$, which is reasonable because a read operation (i.e., recovery) takes the same amount of time as a write operation (i.e., checkpointing) [15]. Also, the verification time at full speed is set to be the same as the cost of a local checkpoint (i.e., memory copy), as a complete verification checks all the data in memory. The parameters λ, C and V are set according to the real values of four platforms [14], which are reported in Table I. The power parameters $P_{\text{cpu}}, P_{\text{io}}, P_{\text{idle}}$, as well as the set $\mathcal{S}$ of available speeds, are determined by the processor used. Table II gives the values of these parameters for two processors reported in [16]. Simulations are then conducted based on eight virtual configurations, each of which combines one platform and one processor type. In the simulations, the default value of P_{io} is set to be equivalent to the power used when the CPU runs at the lowest speed, and the performance bound is set as $\rho = 3$. All of these parameters as well as the values of C, V and λ will be varied in the simulations to evaluate their impact on performance.

Table I. Platform parameters (verification time is w.r.t. full speed).

Platform	λ (error/second)	C (second)	V (second)
Hera	3.38e-6	300	15.4
Atlas	7.78e-6	439	9.1
Coastal	2.01e-6	1051	4.5
Coastal SSD	2.01e-6	2500	180.0

Table II. Processor parameters.

Processor	Normalized speeds	$P(\sigma)$ (mW)
Intel Xscale	$0.15, 0.4, 0.6, 0.8, 1$	$1550\sigma^3 + 60$
Transmeta Crusoe	$0.45, 0.6, 0.8, 0.9, 1$	$5756\sigma^3 + 4.4$

B. A different re-execution speed does help

Does using two different speeds indeed help reduce the energy overhead compared to using one speed alone? This section provides affirmative answer to the question above with concrete examples. Specifically, the following tables present the results for the Hera/XScale configuration (similar results are also observed on other platform/processor configurations [3]), with a different bound ρ for every table. For each initial speed σ_1, the best second speed σ_2 that gives the smallest energy overhead while satisfying the bound ρ is reported, with the corresponding W_{opt} and energy overhead. The overall best speed pair is highlighted in bold.

σ_1	**Best** σ_2	W_{opt}	$\frac{\mathcal{E}(W_{\text{opt}},\sigma_1,\sigma_2)}{W_{\text{opt}}}$
0.15	0.4	1711	466
0.4	**0.4**	2764	416
0.6	0.4	3639	674
0.8	0.4	4627	1082
1	0.4	5742	1625

$\rho = 8$

σ_1	**Best** σ_2	W_{opt}	$\frac{\mathcal{E}(W_{\text{opt}},\sigma_1,\sigma_2)}{W_{\text{opt}}}$
0.15	-	-	-
0.4	**0.4**	2764	416
0.6	0.4	3639	674
0.8	0.4	4627	1082
1	0.4	5742	1625

$\rho = 3$

σ_1	**Best** σ_2	W_{opt}	$\frac{\mathcal{E}(W_{\text{opt}},\sigma_1,\sigma_2)}{W_{\text{opt}}}$
0.15	-	-	-
0.4	-	-	-
0.6	**0.8**	4251	690
0.8	0.4	4627	1082
1	0.4	5742	1625

$\rho = 1.775$

σ_1	**Best** σ_2	W_{opt}	$\frac{\mathcal{E}(W_{\text{opt}},\sigma_1,\sigma_2)}{W_{\text{opt}}}$
0.15	-	-	-
0.4	-	-	-
0.6	-	-	-
0.8	**0.4**	4627	1082
1	0.4	5742	1625

$\rho = 1.4$

From these tables, we see that in many cases the best solution is composed of two different speeds. In fact, it is possible, for a well-chosen ρ, to have almost any speed pair as the optimal solution (except the pairs with very low speeds). This is because, as ρ decreases, the number of speed pairs that satisfy the performance bound also decreases. Hence, if the energy increases with (σ_1, σ_2), then for any speed pair we could find values of ρ such that it gives the best solution. However, as the speed becomes too slow, the energy will increase as we are likely to have more errors and hence re-executions. This is shown in the first table, where ρ is big enough such that the pair $(0.15, 0.15)$ gives a feasible solution but has a higher energy overhead than $(0.4, 0.4)$, which is the optimal speed pair. It turns out, in this configuration, that all speed pairs except the ones containing 0.15 can be the optimal solution, depending on the value of ρ specified. This is confirmed by the simulations shown in the next section.

C. Impact of various parameters

This section evaluates the impact of various parameters on the behaviors of the optimal pattern and the associated energy overhead for the Atlas/Crusoe configuration. Due to lack of space, the results for the other platform/processor configurations are omitted here and can be found in the companion research report [3].

1) Impact of C and V: Figure 2 shows that, as the checkpointing time C is increased, the optimal pattern size W also increases till it is constrained by the performance bound, in which case the execution speeds are adapted (first σ_2 and then σ_1) to prevent the energy overhead (as well as the pattern size) from growing too fast. In this configuration, the optimal speed pair starts at (0.45, 0.45) when C is small and reaches (0.45, 0.8) when C is increased to 5000 seconds. Compared to the optimal solution that uses only one speed (shown in dotted line), using two speeds achieves up to 35% improvement in the energy overhead. Similar savings are also observed when the verification time V (with respect to the maximum speed) is varied, and the results are shown in Figure 3. In this case, the optimal speed pair stabilizes at $(0.6, 0.45)$ when V is increased to 5000 seconds.

2) Impact of λ and ρ: Figure 4 shows the evolution of the optimal solution as a function of the error rate λ. The optimal pattern size W reduces with increasing λ while the execution speeds increase (first σ_2 and then σ_1 till both reach the maximum value), so that errors can be detected earlier (due to decreased W) and sooner (due to increased speeds) in the computation. To satisfy a stricter performance bound, i.e., as ρ is reduced, the speeds increase in a similar fashion, as shown in Figure 5. However, for a fixed error rate λ, the optimal pattern size increases with increased σ_1 and decreases with σ_2, as dictated by Equation (5). In both experiments, we can see that using two execution speeds allows the application to checkpoint less frequently while providing significant energy savings over its one-speed counterpart.

3) Impact of P_{idle} and P_{io}: Figures 6 and 7 show, respectively, the impact of the idle power P_{idle} and dynamic I/O power P_{io} on the performance of the optimal solution. In both cases, the optimal energy overhead as well as the pattern size increase with the power consumption. However, the execution speeds increase (σ_1 first and then σ_2) with P_{idle} but are not affected by P_{io}. This is because increasing the speeds helps counterbalance the increase in energy overhead as P_{idle} becomes larger (see Equation (3)), which is not true for the I/O power since its dominating term does not contain P_{io}. Furthermore, since the optimal re-execution speed σ_2 is (almost always) the same as the initial speed σ_1, the same performance can be achieved by using one speed alone.

4) Summary: Overall, the simulations presented in this section demonstrate the benefit of using two execution speeds for the BICRIT problem. The results show that, under various parameter settings, up to 35% of the energy consumption can be saved by using a different re-execution speed while meeting a prescribed performance constraint.

V. EXTENSIONS

In this section, we show how to compute $\mathcal{T}(W, \sigma_1, \sigma_2)$ and $\mathcal{E}(W, \sigma_1, \sigma_2)$ when the platform is subject to both fail-stop and silent errors. Unfortunately, the first-order approximation is valid only when σ_2 is not too large with respect to σ_1, and we are no longer able to provide a general closed-form solution to the BICRIT problem.

A. Dealing with fail-stop and silent errors

Suppose that the platform is subject to two (independent) error sources: in addition to silent errors, whose rate is now denoted as λ^s, there are also fail-stop errors (such as process crashes) striking with rate λ^f. The probability of having a fail-stop error during T units of time is $p^f(T) = 1 - e^{-\lambda^f T}$ (and that of a silent error remains $p^s(T) = 1 - e^{-\lambda^s T}$). We assume that fail-stop errors can occur during computation and verification, but not during checkpointing and recovery. Also, a fail-stop error is detected immediately after striking, so that the time lost since the last checkpoint when executing at speed σ gets smaller on average than $\frac{W+V}{\sigma}$ (see Figure 1(b)). In fact, we can compute its expectation, as shown in the proposition below, which derives $\mathcal{T}(W, \sigma_1, \sigma_2)$ with two error sources.

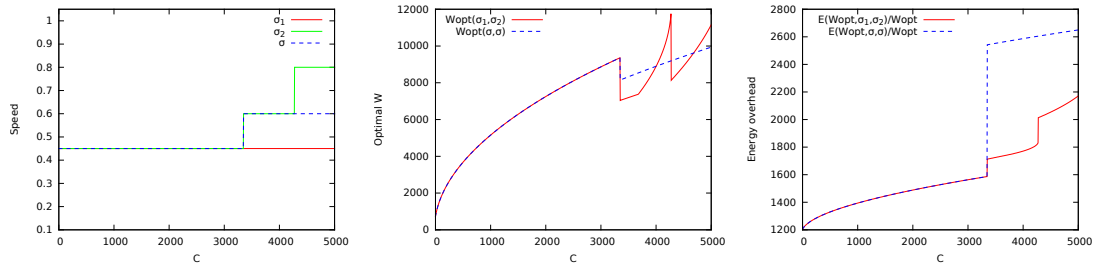


Figure 2. The optimal solution (speed pair, pattern size, and energy overhead) as a function of the checkpointing time C in Atlas/Crusoe configuration.

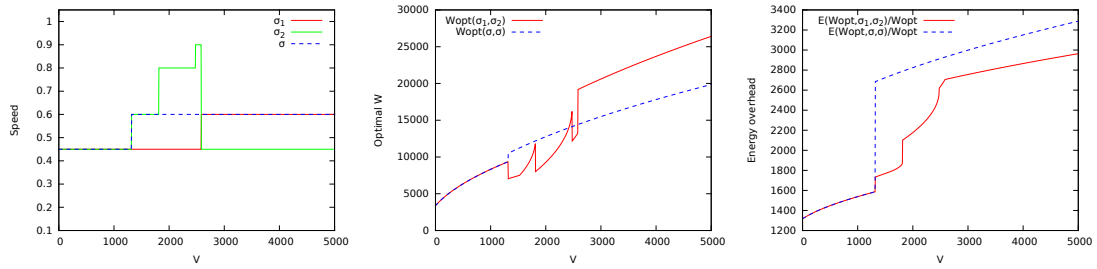


Figure 3. The optimal solution (speed pair, pattern size, and energy overhead) as a function of the verification time V in Atlas/Crusoe configuration.

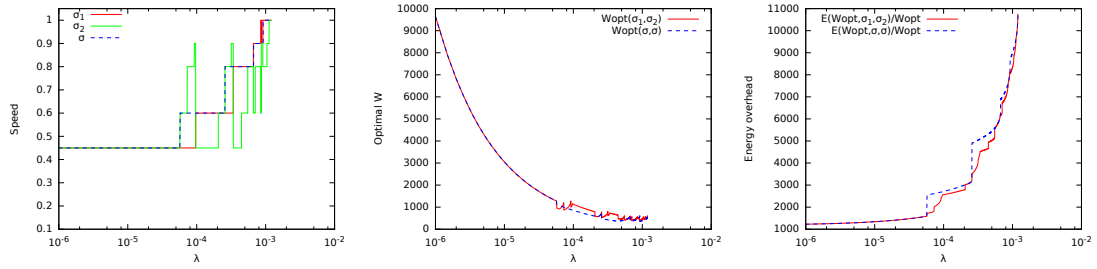


Figure 4. The optimal solution (speed pair, pattern size, and energy overhead) as a function of the error rate λ in Atlas/Crusoe configuration.

Proposition 4. *With fail-stop and silent errors,*

$$\mathcal{T}(W,\sigma_1,\sigma_2) = C + \left(1 - e^{-\frac{\lambda^f(W+V)+\lambda^s W}{\sigma_1}}\right) e^{\frac{\lambda^f(W+V)+\lambda^s W}{\sigma_2}} R + \left(1 - e^{-\frac{\lambda^f(W+V)+\lambda^s W}{\sigma_1}}\right) e^{\frac{\lambda^s W}{\sigma_2}} \frac{V}{\sigma_2} + \frac{1}{\lambda^f}\left(1 - e^{-\frac{\lambda^f(W+V)}{\sigma_1}}\right) + \frac{1}{\lambda^f}\left(1 - e^{-\frac{\lambda^f(W+V)+\lambda^s W}{\sigma_1}}\right) e^{\frac{\lambda^s W}{\sigma_2}} \left(e^{\frac{\lambda^f(W+V)}{\sigma_2}} - 1\right).$$

Proof. The analysis becomes more involved with two error sources. The following gives the recursive equation for computing the expected execution time $\mathcal{T}(W,\sigma_1,\sigma_2)$:

$$\mathcal{T}(W,\sigma_1,\sigma_2) = p^f\left(\frac{W+V}{\sigma_1}\right)\left(\mathcal{T}_{\text{lost}}(W+V,\sigma_1) + R + \mathcal{T}(W,\sigma_2,\sigma_2)\right) + \left(1 - p^f\left(\frac{W+V}{\sigma_1}\right)\right)\left(\frac{W+V}{\sigma_1} + p^s\left(\frac{W}{\sigma_1}\right)\left(R + \mathcal{T}(W,\sigma_2,\sigma_2)\right) + \left(1 - p^s\left(\frac{W}{\sigma_1}\right)\right) C\right). \quad (7)$$

Here is the case analysis:

- If a fail-stop error occurs, we lose in expectation $\mathcal{T}_{\text{lost}}(W+V,\sigma_1)$ time. Then, we recover and re-execute at speed σ_2.
- Otherwise, we retrieve the former case analysis with silent errors only (see Propositions 1 and 2).

From [12], we get that $\mathcal{T}_{\text{lost}}(W+V,\sigma) = \frac{1}{\lambda^f} - \frac{(W+V)/\sigma}{e^{\lambda^f(W+V)/\sigma}-1}$. Solving Equation (7) and rearranging terms lead to the expression of $\mathcal{T}(W,\sigma_1,\sigma_2)$ as claimed. □

The following proposition shows the expected energy consumption $\mathcal{E}(W,\sigma_1,\sigma_2)$ with two error sources. The proof is similar to that of Proposition 4 and is omitted.

Proposition 5. *With fail-stop and silent errors,*

$$\mathcal{E}(W,\sigma_1,\sigma_2) = C(P_{\text{io}} + P_{\text{idle}}) + \left(1 - e^{-\frac{\lambda^f(W+V)+\lambda^s W}{\sigma_1}}\right) e^{\frac{\lambda^f(W+V)+\lambda^s W}{\sigma_2}} R(P_{\text{io}} + P_{\text{idle}}) + \left(1 - e^{-\frac{\lambda^f(W+V)+\lambda^s W}{\sigma_1}}\right) e^{\frac{\lambda^s W}{\sigma_2}} \frac{V}{\sigma_2}(\kappa\sigma_2^3 + P_{\text{idle}}) + \frac{1}{\lambda^f}\left(1 - e^{-\frac{\lambda^f(W+V)+\lambda^s W}{\sigma_1}}\right) e^{\frac{\lambda^s W}{\sigma_2}} \left(e^{\frac{\lambda^f(W+V)}{\sigma_2}} - 1\right)(\kappa\sigma_2^3 + P_{\text{idle}}) + \frac{1}{\lambda^f}\left(1 - e^{-\frac{\lambda^f(W+V)}{\sigma_1}}\right)(\kappa\sigma_1^3 + P_{\text{idle}}).$$

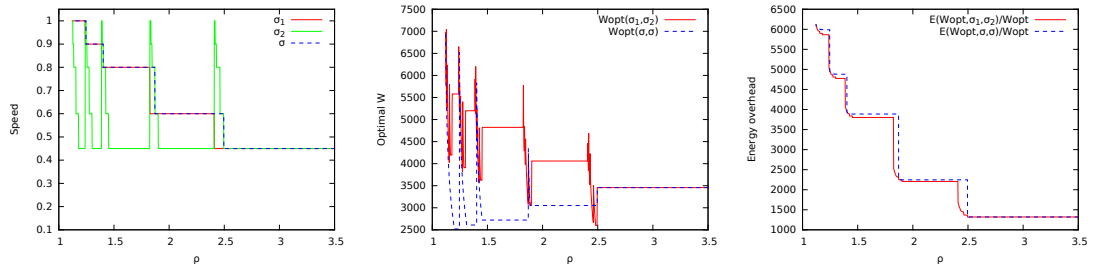


Figure 5. The optimal solution (speed pair, pattern size, and energy overhead) as a function of the performance bound ρ in Atlas/Crusoe configuration.

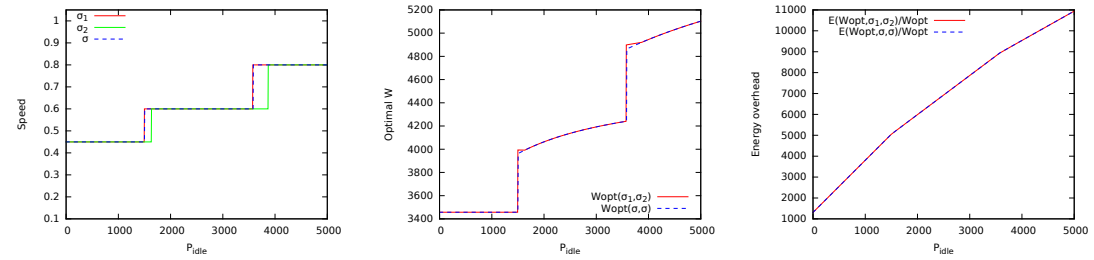


Figure 6. The optimal solution (speed pair, pattern size, and energy overhead) as a function of the idle power P_{idle} in Atlas/Crusoe configuration.

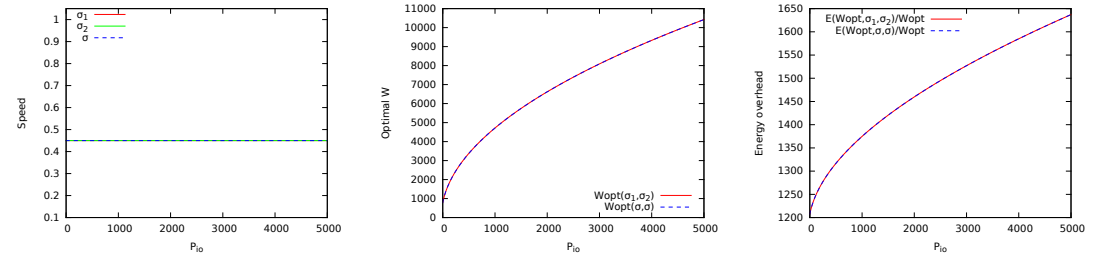


Figure 7. The optimal solution (speed pair, pattern size, and energy overhead) as a function of the I/O power P_{io} in Atlas/Crusoe configuration.

B. Limits of the first-order approximation

Let $\lambda = 1/\mu$ denote the total error rate when accounting from both error sources, and suppose f fraction of the total number of errors are fail-stop and the remaining $s = 1 - f$ fraction are silent. Then, the arrival rates of fail-stop and silent errors are $\lambda^f = f\lambda$ and $\lambda^s = s\lambda$, respectively. After some tedious manipulations, we can derive, from Propositions 4 and 5, first-order approximations for the time and energy overheads, as shown in the following proposition.

Proposition 6. *With fail-stop and silent errors,*

$$\frac{\mathcal{T}(W,\sigma_1,\sigma_2)}{W} = \frac{C+V/\sigma_1}{W} + \left(\frac{(f+s)}{\sigma_1\sigma_2} - \frac{f}{2\sigma_1^2}\right)\lambda W + \frac{(f+s)\lambda(R+V/\sigma_2)+1-f\lambda V/\sigma_1}{\sigma_1} + O(\lambda^2 W). \quad (8)$$

$$\frac{\mathcal{E}(W,\sigma_1,\sigma_2)}{W} = \frac{C(P_{\text{io}}+P_{\text{idle}})+V(\kappa\sigma_1^3+P_{\text{idle}})/\sigma_1}{W} + \left(\frac{(f+s)(\kappa\sigma_2^3+P_{\text{idle}})}{\sigma_1\sigma_2} - \frac{f(\kappa\sigma_1^3+P_{\text{idle}})}{2\sigma_1^2}\right)\lambda W + \frac{(f+s)\lambda\left(R(P_{\text{io}}+P_{\text{idle}})+V(\kappa\sigma_2^3+P_{\text{idle}})/\sigma_2\right)}{\sigma_1} + \frac{(1-f\lambda V/\sigma_1)(\kappa\sigma_1^3+P_{\text{idle}})}{\sigma_1} + O(\lambda^2 W). \quad (9)$$

Consider Equation (8) and compare it to Equation (2): it still is of the form $x + \frac{y}{W} + zW + O(\lambda^2 W)$, where x and y are positive constants, but now the sign of $z = \frac{(f+s)\lambda}{\sigma_1\sigma_2} - \frac{f\lambda}{2\sigma_1^2}$ depends on the ratio $\frac{\sigma_2}{\sigma_1}$. If $\frac{\sigma_2}{\sigma_1} < 2(1+\frac{s}{f})$, then z is positive and Equation (8) is minimized when $W = \sqrt{\frac{y}{z}} = \Theta(\lambda^{-1/2})$, just as with silent errors only. But if $\frac{\sigma_2}{\sigma_1} > 2(1+\frac{s}{f})$, then z becomes negative and the time overhead is a strictly decreasing function of W. The minimum would then be achieved for arbitrarily large pattern size W. This is not allowed because, for the Taylor expansion to be valid, we need $\lambda W = \varepsilon(\lambda)$ where ε is a function s.t. $\lim_{\lambda\to 0}\varepsilon(\lambda) = 0$. There is an interesting case limit when $\frac{\sigma_2}{\sigma_1} = 2(1+\frac{s}{f})$, which we will discuss in Section V-C.

Looking at Equation (9), we get a similar result: the first-order approximation leads to a valid solution only when $z > 0$, which leads to the constraint $\frac{\sigma_2}{\sigma_1} < 2(1+\frac{s}{f})\frac{\kappa\sigma_2^3+P_{\text{idle}}}{\kappa\sigma_1^3+P_{\text{idle}}}$. For the sake of simplification, assume that $P_{\text{idle}} = 0$. The latter condition then translates into $\frac{\sigma_2}{\sigma_1} > \left(2(1+\frac{s}{f})\right)^{-1/2}$. Altogether, the first-order approximation will lead to a solution to the BiCrit problem if and only if

$$\left(2(1+\frac{s}{f})\right)^{-1/2} < \frac{\sigma_2}{\sigma_1} < 2(1+\frac{s}{f}).$$

While the interval defined by the above condition is never empty, it bounds the ratio $\frac{\sigma_2}{\sigma_1}$, thereby limiting the applicability of the first-order approximation.

C. Second-order approximation

Since the first-order approximation has shown its limits, we could envision resorting to the second-order approximation. Unfortunately, despite all our efforts, the second-order approximation cannot help solve the BICRIT problem in the general case. However, the approach enables us to derive the optimal checkpointing period when $\sigma_2 = 2\sigma_1$ with the optimization of expected execution time.

As the issue is caused by fail-stop errors, for the sake of simplification, we assume that $s = 0$ (hence $f = 1$, which means no silent errors, only fail-stop errors).

Theorem 2. *When considering only fail-stop errors with rate λ, the optimal pattern size W to minimize the time overhead $\frac{\mathcal{T}(W,\sigma,2\sigma)}{W}$ is*

$$W_{\text{opt}} = \sqrt[3]{\frac{12C}{\lambda^2}}\sigma.$$

This result is really striking: it is the first resilience framework (to the best of our knowledge) in which the optimal checkpointing size W_{opt} is not in the order of the square root of MTBF; instead of having $W_{\text{opt}} = \Theta(\lambda^{-1/2})$ as in Young/Daly's formula, we have $W_{\text{opt}} = \Theta(\lambda^{-2/3})$ when we re-execute twice faster. We stress that this result is not directly related to the BICRIT problem, but applies to the classical optimization problem of finding the best checkpointing period to minimize the expected execution time.

Proof. We start by deriving the second-order expression of the time overhead given below:

Proposition 7. *With fail-stop errors only,*

$$\frac{\mathcal{T}(W,\sigma_1,\sigma_2)}{W} = \frac{1}{\sigma_1} + \frac{C}{W} + \left(\frac{1}{\sigma_1\sigma_2} - \frac{1}{2\sigma_1^2}\right)\lambda W + \frac{\lambda R}{\sigma_1} + \left(\frac{1}{6\sigma_1^3} - \frac{1}{2\sigma_1^2\sigma_2} + \frac{1}{2\sigma_1\sigma_2^2}\right)\lambda^2 W^2 + O(\lambda^3 W^2). \quad (10)$$

The proof of Proposition 7 is straightforward. We use it with $\sigma_1 = \sigma$ and $\sigma_2 = 2\sigma$. The coefficient of W becomes zero in Equation (10), and we derive $\frac{\mathcal{T}(W,\sigma,2\sigma)}{W} = \frac{1}{\sigma} + \frac{C}{W} + \frac{\lambda^2 W^2}{24\sigma^3} + \frac{\lambda R}{\sigma}$ while neglecting low-order terms. When differentiating, we see that this expression is minimized for $W_{\text{opt}} = \sqrt[3]{\frac{12C}{\lambda^2}}\sigma$. □

VI. CONCLUSION

Silent errors and energy consumption are two major challenges towards exascale computing. In this paper, we have shown that using two different speeds (one for the first execution, and one for re-executions due to errors) can lead to significant energy savings while satisfying a performance constraint. For silent errors, we have extended the classical formula of Young/Daly for a divisible load application; by deriving a first-order approximation of the expected execution time and energy consumption, we have obtained a general, closed-form solution to get both the optimal speed pair and the associated optimal checkpointing period. Extensive simulations confirm that having a second speed can indeed help, with up to 35% savings in energy, and that many speed pairs can be potential candidates for the optimal solution, depending on the tightness of the performance bound. All these results shed a new light on the optimal checkpointing period for platforms whose processor speeds can be adjusted through DVFS.

Further work will aim at complementing our initial study for both silent and fail-stop errors. With both error sources, the first-order approximation is valid only when the second speed is not too large with respect to the first one, otherwise we are no longer able to provide a general closed-form solution. In particular, we have derived a striking result with fail-stop errors when re-execution is twice faster: in this case, the optimal checkpointing period is (surprisingly) in the order of $\Theta(\lambda^{-2/3})$ instead of $\Theta(\lambda^{-1/2})$ as in the Young/Daly formula. It seems that new methods are needed to capture the general case with two error sources and arbitrary speed pairs.

Acknowledgments: This research was funded in part by the European project SCoRPiO, by the LABEX MILYON (ANR-10-LABX-0070) of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX-0007) operated by the French National Research Agency (ANR), and by the PIA ELCI project. Yves Robert is with Institut Universitaire de France.

REFERENCES

[1] G. Aupy, A. Benoit, R. Melhem, P. Renaud-Goud, and Y. Robert. Energy-aware checkpointing of divisible tasks with soft or hard deadlines. In *IGCC*, 2013.

[2] N. Bansal, T. Kimbrel, and K. Pruhs. Speed scaling to manage energy and temperature. *Journal of the ACM*, 54(1):3:1–3:39, 2007.

[3] A. Benoit, A. Cavelan, V. Le Fèvre, Y. Robert, and H. Sun. A different re-execution speed can help. Research Report RR-8888, INRIA, 2016. Available at graal.ens-lyon.fr/~yrobert/rr8888.pdf.

[4] A. Benoit, A. Cavelan, Y. Robert, and H. Sun. Optimal resilience patterns to cope with fail-stop and silent errors. In *IPDPS*, 2016.

[5] A. Benoit, Y. Robert, and S. K. Raina. Efficient checkpoint/verification patterns. *Int. J. High Performance Computing Applications*, 2015.

[6] A. R. Benson, S. Schmit, and R. Schreiber. Silent error detection in numerical time-stepping schemes. *Int. J. High Performance Computing Applications*, 2014.

[7] G. Bosilca, R. Delmas, J. Dongarra, and J. Langou. Algorithm-based fault tolerance applied to high performance computing. *J. Parallel Distrib. Comput.*, 69(4):410–416, 2009.

[8] F. Cappello, A. Geist, W. Gropp, S. Kale, B. Kramer, and M. Snir. Toward exascale resilience: 2014 update. *Supercomputing frontiers and innovations*, 1(1), 2014.

[9] Z. Chen. Online-ABFT: An online algorithm based fault tolerance scheme for soft error detection in iterative methods. In *PPoPP*, 2013.

[10] J. T. Daly. A higher order estimate of the optimum checkpoint interval for restart dumps. *Future Generation Comp. Syst.*, 22(3):303–312, 2006.

[11] D. Fiala, F. Mueller, C. Engelmann, R. Riesen, K. Ferreira, and R. Brightwell. Detection and correction of silent data corruption for large-scale high-performance computing. In *Proc. SC'12*, page 78, 2012.

[12] T. Hérault and Y. Robert, editors. *Fault-Tolerance Techniques for High-Performance Computing*, Computer Communications and Networks. Springer Verlag, 2015.

[13] R. E. Lyons and W. Vanderkulk. The use of triple-modular redundancy to improve computer reliability. *IBM J. Res. Dev.*, 6(2):200–209, 1962.

[14] A. Moody, G. Bronevetsky, K. Mohror, and B. R. d. Supinski. Design, modeling, and evaluation of a scalable multi-level checkpointing system. In *Proc. SC'10*, pages 1–11, 2010.

[15] F. Quaglia. A cost model for selecting checkpoint positions in time warp parallel simulation. *IEEE Trans. Parallel Distrib. Syst.*, 12(4):346–362, 2001.
[16] N. B. Rizvandi, A. Y. Zomaya, Y. C. Lee, A. J. Boloori, and J. Taheri. Multiple frequency selection in DVFS-enabled processors to minimize energy consumption. In A. Y. Zomaya and Y. C. Lee, editors, *Energy-Efficient Distributed Computing Systems*. John Wiley, 2012.
[17] P. Sao and R. Vuduc. Self-stabilizing iterative solvers. In *ScalA*, 2013.
[18] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *FOCS*, 1995.
[19] J. W. Young. A first order approximation to the optimum checkpoint interval. *Comm. of the ACM*, 17(9):530–531, 1974.