Robustness of the Young/Daly formula for stochastic iterative applications

Yishu Du
yishu.du@inria.fr
Tongji University
Shanghai, China
LIP, ENS Lyon
Lyon, France

Loris Marchal
loris.marchal@inria.fr
LIP, ENS Lyon
Lyon, France

Guillaume Pallez (Aupy)
guillaume.pallez@inria.fr
Inria, Univ. of Bordeaux
Talence, France

Yves Robert
yves.robert@inria.fr
LIP, ENS Lyon
Lyon, France
Univ. Tenn. Knoxville
Knoxville, USA

ABSTRACT

The Young/Daly formula for periodic checkpointing is known to hold for a divisible load application where one can checkpoint at any time-step. In an nutshell, the optimal period is $\mathcal{P}_{YD} = \sqrt{2\mu_f C}$ where μ_f is the Mean Time Between Failures (MTBF) and C is the checkpoint time. This paper assesses the accuracy of the formula for applications decomposed into computational iterations where: (i) the duration of an iteration is stochastic, i.e., obeys a probability distribution law $\mathcal{D}$ of mean $\mu_{\mathcal{D}}$; and (ii) one can checkpoint only at the end of an iteration. We first consider static strategies where checkpoints are taken after a given number of iterations k and provide a closed-form, asymptotically optimal, formula for k, valid for any distribution $\mathcal{D}$. We then show that using the Young/Daly formula to compute k (as $k \cdot \mu_{\mathcal{D}} = \mathcal{P}_{YD}$) is a first order approximation of this formula. We also consider dynamic strategies where one decides to checkpoint at the end of an iteration only if the total amount of work since the last checkpoint exceeds a threshold W_{th} , and otherwise proceed to the next iteration. Similarly, we provide a closed-form formula for this threshold and show that $\mathcal{P}_{YD}$ is a first-order approximation of W_{th} . Finally, we provide an extensive set of simulations where $\mathcal{D}$ is either Uniform, Gamma or truncated Normal, which shows the global accuracy of the Young/Daly formula, even when the distribution $\mathcal{D}$ had a large standard deviation (and when one cannot use a first-order approximation). Hence we establish that the relevance of the formula goes well beyond its original framework.

ICPP '20, August 17–20, 2020, Edmonton, AB, Canada

ACM ISBN 978-1-4503-8816-0/20/08…$15.00
https://doi.org/10.1145/3404397.3404419

CCS CONCEPTS

• **Computer systems organization** → **Real-time systems**; *Reliability*; • **Theory of computation** → **Design and analysis of algorithms**; **Parallel algorithms**.

KEYWORDS

fault-tolerance, checkpoint, Young/Daly formula, iterative algorithm

ACM Reference Format:
Yishu Du, Loris Marchal, Guillaume Pallez (Aupy), and Yves Robert. 2020. Robustness of the Young/Daly formula for stochastic iterative applications. In *49th International Conference on Parallel Processing - ICPP (ICPP '20), August 17–20, 2020, Edmonton, AB, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3404397.3404419

1 INTRODUCTION

Large-scale platforms are increasingly subject to errors [7, 8]. Current computing platforms have millions of cores: the Summit system at the Oak Ridge National Laboratory (ORNL) is listed at number one in the TOP500 ranking [29], and it has more than two million cores. The Chinese Sunway TaihuLight (ranked as number 3) has even more than 10 million cores. These large computing systems are frequently confronted with failures, also called fail-stop errors (such as hardware failures or crashes) . Scale is the enemy here: assume that each computing resource is very reliable, with, say, a *Mean Time Between Failures* (MTBF) of ten years, meaning that each resource will experience an error only every 10 years on average. Then a platform composed of 100, 000 of such resources will experience a failure every 50 minutes; with 1 million resources, the platform is struck every five minutes [19]. Hence, fault-tolerance techniques to mitigate the impact of errors are required to ensure a correct and uninterrupted execution of the application [8].

The classical technique in High Performance Computing (HPC) to deal with failures consists of using a checkpoint-restart mechanism: the state of the application is periodically checkpointed, and when a failure occurs, one recovers from the last valid checkpoint and resumes the execution from that point on, rather than starting

the execution from scratch. The key for an efficient checkpointing policy is to decide how often to checkpoint. Indeed, checkpointing too often leads to spending too much time in checkpoints, as opposed as to executing useful work. On the contrary, checkpointing too infrequently leads to wasting too much time to re-execute work that has been lost after a failure. Young [32] and Daly [9] derived the well-known Young/Daly formula $\mathcal{P}_{YD} = \sqrt{2\mu_f C}$ for the optimal[1] checkpointing period, where μ_f is the platform MTBF and C is the checkpoint duration. Assuming unit speed, the time $\mathcal{P}_{YD}$ elapsed between two checkpoints is also the amount of work executed during each period. The Young/Daly formula applies to applications where one can checkpoint at any instant. Divisible-load applications [4, 26] are examples of such applications.

However, many scientific applications exhibit a more complicated behavior. In this work, we focus on *iterative applications* which we define as applications that are decomposed into computational *iterations*, and where one can checkpoint only at the end of an iteration. Indeed, for iterative applications, checkpointing is efficient, let alone possible, only at the end of an iteration, because the volume of data to checkpoint is dramatically reduced at that point. A wide range of applications fits in this framework. Iterative solvers for sparse linear algebra systems are a representative example [25, 27]. Moreover, the time of each iteration depends upon the several parameters (sparsity pattern of some vectors, communication contention, system jitter) and can vary significantly from one iteration to another. To illustrate the variability of linear algebra tasks, [23] shows a performance range from 30 to 80 Gflops. for the same multicore matrix-matrix multiplication kernel. This variability phenomenon is amplified in randomized iterative methods [15] where random vectors are generated as the application progresses. Another class of applications that are naturally decomposed into iterations of variable length are Bulk Synchronous Parallel (BSP) applications [14, 20] where one checkpoints at the end of each join operation. A typical example of a BSP sequence of fork-join operations is the n-body computation [6]. Due to the simplicity of the programming model, many BSP applications are deployed at scale [5].

As already mentioned, many iterative or BSP applications exhibit iterations of variable length, typically because each iteration is data-dependent. When considering an iterative application, we assume that the length of each iteration is not known a priori, but instead is drawn randomly from some probability distribution $\mathcal{D}$. Again, with unit speed, the length of the iteration is the amount of work within the iteration. The distribution $\mathcal{D}$ is usually acquired by sampling a few executions. This technique is illustrated in [13] where many distributions are identified from sampling medical image analysis tasks. In this paper, we use several usual distributions, such as Uniform, Gamma or Normal.

The main objective of this paper is to explore whether the Young/-Daly formula applies beyond divisible-load applications. To what extent can we use the formula for iterative applications whose length obey a probability distribution $\mathcal{D}$? We first consider static strategies where checkpoints are taken after a given number of iterations k, and we show that using the Young/Daly formula to compute k (as $k \cdot \mu_{\mathcal{D}} = \mathcal{P}_{YD}$) is asymptotically optimal among such strategies, and remains accurate even when the distribution $\mathcal{D}$ had a large standard deviation. Then we consider dynamic strategies where one decides to checkpoint at the end of an iteration only if the total amount of work since the last checkpoint exceeds a threshold W_{th} , and otherwise proceed to the next iteration; we show that an approximation of the optimal value of W_{th} is $\mathcal{P}_{YD}$. Finally, we provide an extensive set of simulations where $\mathcal{D}$ is either Uniform, Gamma or Normal, which shows the global accuracy of the Young/Daly formula and establish that its relevance goes well beyond its original framework.

The main contributions of this paper are the following:

- For static solutions, we derive a closed-form formula to compute the optimal checkpointing period, and we show that its first-order approximation corresponds to the Young/Daly formula. The derivation is quite technical, and constitutes a major extension of the deterministic case.
- For dynamic solutions, we derive a closed-form formula to compute the threshold at which one decides either to checkpoint or to execute more work, and we show that its first-order approximation also corresponds to the Young/Daly formula. Again, the derivation is complicated and required to use a simplified objective, using the ratio of expectations of actual time over useful time, instead of the expectation of these ratios (see Section 5 for details).
- We conduct an extensive set of experiments with classic probability distributions (Uniform, Gamma, Normal) and we conclude that the Young/Daly formula remains accurate and useful in a stochastic setting.

The paper is organized as follows. We briefly review existing work on checkpointing parallel applications in Section 2. We formally state the model for iterative applications in Section 3. Section 4 is the core of the paper to state the static strategy. We state the dynamic strategy in Section 5. Section 6 is devoted to simulations. Finally, we conclude and give hints for future work in Section 7.

2 RELATED WORK

We survey related work in this section. We start with checkpointing in Section 2.1. Then we discuss iterative applications in Section 2.2.

2.1 Checkpointing

Checkpoint-restart is one of the most used strategy to deal with fail-stop errors, and several variants of this policy have been studied, see [19] for a survey. The natural strategy is to checkpoint periodically, and one must then decide how often to checkpoint, hence derive the optimal checkpointing period. For a divisible-load application, results were first obtained by Young [32] and Daly [9], who showed how to derive the optimal checkpointing period. This periodic strategy has been extended to deal with a multi-level checkpointing scheme [3, 10, 24], or by using SSD or NVRAM as secondary storage [8].

Going beyond divisible-load applications, some works target checkpointing strategies for workflows. Workflows are expressed in terms of directed acyclic graphs (DAGs) where vertices represent the computational tasks and edges represent dependences between

[1]The objective function is to minimize the expectation of the total execution time, see Section 3 for details.

tasks. Workflows are similar to iterative applications in that checkpointing is only possible right after the completion of a task. The simplest workflows are linear chains of tasks. If these tasks are parallel, we have as iterative application deployed on the platform, but whose iterations have deterministic execution times, namely the durations of the tasks. The problem of finding the optimal checkpoint strategy for a linear chain of tasks (determining which tasks to checkpoint), in order to minimize the expected execution time, has been solved by Toueg and Babaoglu [30] using a dynamic programming algorithm. For general workflows, finding an optimal solution is a #P-complete problem [16]. Recall that #P is the class of counting problems that correspond to NP decision problems [31], and that #P-complete problems are at least as hard as NP-complete problems. Several heuristics to decide which tasks to checkpoint are proposed and evaluated in [17].

2.2 Iterative applications

Iterative methods are popular for solving large sparse linear systems, which have a wide range of applications in several scientific and industrial problems. There are many classic iterative methods including stationary iterative methods like the Jacobi method, the Gauss-Seidel method and the Successive Overrelaxation method (SOR), and non-stationary iterative methods like the Conjugate Gradient method (CG), the Generalized Minimum Residual method (GMRES) and the Biconjugate Gradient Stabilized method (BICGSTAB) [27]. In recent years, randomized iterative methods have been much more popular. For example, the randomized Kaczmraz method [28] and the greedy randomized Kaczmarz method [1] for solving consistent linear system, the randomized coordinate descent method [21, 22] and the greedy randomized coordinate descent method [2] for solving least square problems. For these iterative methods, it is economic to set checkpoints at the end of the iterations since the volume of data need to be stored is dramatically reduced at that point. Furthermore, in all these methods, the time spent per iteration is not constant: for classic iterative methods, the amount of flops is usually the same per iteration but the communication volume and the amount of contention varies from one iteration to another. The variation becomes more important for randomized applications, where random vectors are generated as the application progresses and the amount of flops per iteration changes according to the sparsity pattern [15].

Another class of iterative applications arises from the Bulk Synchronous Parallel (BSP) model, which was originally suggested as a possible 'bridging' model to serve as a standard interface between the architecture levels and language in parallel computations [14, 20]. The representative n-body computations [6] have a number of important applications in fields such as molecular dynamics, fluid dynamics, computer graphics, and even astrophysics [18]. A BSP computation consists of a sequence of parallel super-steps, composed of fork-join operations with independent threads executed in parallel. It is economical to set up checkpoints at the end of the super-steps which naturally fit the definition of iterations. BSP applications that are deployed at scale [5] are composed of a large number of super-steps whose lengths are data dependent and can adequately be modeled as drawn from some probability distribution.

3 FRAMEWORK

We first introduce all model parameters in Section 3.1. Then we formally state the optimization problem, as well as the static and dynamic scheduling strategies in Section 3.2.

3.1 Model

Platform. We consider a parallel platform subject to failures. We assume that the failure inter-arrival times follow an Exponential distribution $\textsc{Exp}(\lambda)$ of parameter λ, whose PDF (Probability Density Function) is $f(x) = \lambda e^{-\lambda x}$ for $x \geq 0$. The MTBF is $\mu_f = \frac{1}{\lambda}$. Note that scale is accounted for by the value of the MTBF $\mu_f = 1/\lambda$. Here λ is the failure rate for the whole platform. If the failure rate is λ_1 for a single processor, it becomes $\lambda = p\lambda_1$ for a platform with p processors (hence the MTBF is divided by p). In addition, when hit by a failure, the platform is unavailable during a downtime D.

Application. We consider an iterative application composed of n consecutive iterations. The execution time of each iteration is not known before execution but follows a probability distribution $\mathcal{D}$. The execution times of the iterations are thus modeled with random variables $X_1, \ldots, X_n$, where the X_i are IID (Independent and identically Distributed) variables following $\mathcal{D}$. Finally, we assume that the iterations are deterministic: the second execution for a given iteration has the same duration as the first one, that is to say, two executions of the same iteration take the same time. After each iteration, one can checkpoint the state of the application at a cost of C units of time. In case of a failure, it takes R units of time (after the downtime D) to recover from the last checkpoint.

Expected execution time of a given iteration. Consider an iteration of length W; we normalize performance so that the application has unit speed; then W also represents the amount of work performed within the iteration. We recall the following result [19, Proposition 1.1]: the expected execution time to perform a work of size W followed by a checkpoint of size C in the presence of failures (Exponential distribution of parameter λ), with a restart cost R and a downtime D is:

$$T_\lambda(W, C, D, R) = \left(\frac{1}{\lambda} + D\right) e^{\lambda R} \left(e^{\lambda(W+C)} - 1\right). \tag{1}$$

In Equation (1), one assumes that failures can strike during checkpoint and recovery, but not during downtime.

3.2 Objective function

Given an iterative application with n iterations, a *solution* is defined as a checkpointing strategy of the form $\mathcal{S} = (\delta_1, \ldots, \delta_n = 1)$ where $\delta_i = 1$ if and only if we perform a checkpoint after the i-th iteration of length X_i. Note that we always checkpoint at the end of the last iteration, because final results are saved on disk in many applications. However, checkpointing the last segment is not mandatory, and our approach can easily handle this. A solution with $m \leq n$ checkpoints writes $\mathcal{S} = (\delta_1, \ldots, \delta_n)$, with $1 \leq i_1 < i_2 < \cdots < i_m = n$ and $\delta_j = 1 \iff j \in \{i_1, \ldots, i_m\}$. We let $i_0 = 0$ and let $W_j = \sum_{l=i_{j-1}+1}^{i_j} X_l$ denote the work between the j-th checkpoint and the previous one (or the beginning of the execution if $j = 1$).

We are interested in minimizing the total execution time (makespan) of the application. This makespan is given by random variable:

$$MS(\mathcal{S}) = \sum_{j=1}^{m} T_\lambda(W_j, C, D, R). \tag{2}$$

For given values of iteration lengths (the X_i variables), the value of the makespan $MS(\mathcal{S})$ is the expected execution time over all failure scenarios, weighted with their probabilities to happen.

In this work, we present and analyze two different strategies to build a solution. In the *static* strategy, we decide before the execution which iterations to checkpoint. In other words, a static solution does not depend upon the value of the X_i variables, it is determined without knowing the iteration lengths. In that case, the optimization objective is easy to express: it is the expectation $\mathbb{E}[MS(\mathcal{S})]$ of the variable $MS(\mathcal{S})$ over the range of the X_i variables which are IID and follow $\mathcal{D}$. Formally:

$$\mathbb{E}[MS(\mathcal{S})] = \mathbb{E}\left[\sum_{j=1}^{m} T_\lambda(W_j, C, D, R)\right]. \tag{3}$$

In Section 4, we show how to design a solution that is asymptotically optimal (where the number of iterations n tends to infinity) among all static solutions.

Contrarily to static strategies, *dynamic* strategies decide which iterations to checkpoint on the fly during execution: at the end of each iteration, we add a checkpoint only if the total work since the last checkpoint (or the beginning of the execution if there was no previous checkpoint) exceeds a given threshold. Hence a dynamic solution may well insert different checkpoints for different values of the iteration lengths. Providing a closed-form formula of the expected makespan of a dynamic solution is complicated, because the values of the δ_i are now conditional to the values of the X_i. We circumvent this difficulty by minimizing the slowdown of a solution, where the slowdown is defined as the ratio of the actual execution time over the base time without any checkpoint nor failure. We refer to Section 5 for further details.

4 STATIC STRATEGIES

This section focuses on static strategies, where checkpoint decisions are made before the execution, based upon application and platform parameters, and do not depend on the actual lengths of the iterations. As stated in Equation (3), the objective is to minimize the expected makespan $\mathbb{E}[MS(\mathcal{S})]$.

Given an application with n iterations, static solutions decide which iterations to checkpoint. One can choose a solution to be periodic with period k, i.e., checkpoints are taken every k iterations, namely at the end of iterations number k, $2k$,... until the last iteration (which is always checkpointed by hypothesis, even if its number n is not a multiple of k). An optimal solution may well not be periodic. However, we prove in Section 4.1 that the periodic solution with period k_{static} given below is asymptotically optimal when n is large, and we show in Section 4.3 that the first-order approximation of the period length corresponds to the Young/Daly formula.

4.1 Asymptotic optimality

We first characterize the expected makespan of a static solution (possibly non-periodic):

Proposition 1. *Given a solution $\mathcal{S} = (\delta_1, \ldots, \delta_n)$ and its associated m checkpoint indices $i_1 < i_2 < \cdots < i_m = n$, let $k_j = i_j - i_{j-1}$ denote the number of iterations between the $j-1$-th checkpoint (or the beginning of the execution if $j = 1$) and the j-th checkpoint. Define*

$$C_{ind}(k) = \frac{e^{\lambda C}\mathbb{E}[e^{\lambda X}]^k - 1}{k}, \tag{4}$$

then the expected makespan is

$$\mathbb{E}[MS(\mathcal{S})] = e^{\lambda R}\left(\frac{1}{\lambda} + D\right)\sum_{j=1}^{m} k_j \cdot C_{ind}(k_j). \tag{5}$$

Proof. Recall that $W_j = \sum_{l=i_{j-1}+1}^{i_j} X_l$ in Equation (1). We have

$$\begin{aligned}
&\mathbb{E}\left[T_\lambda(W_j, C, D, R)\right] && (6)\\
&= \int_{\mathcal{D}_{W_j}} e^{\lambda R}\left(\frac{1}{\lambda} + D\right)\left(e^{\lambda(w+C)} - 1\right) f_{W_j}(w)dw \\
&= e^{\lambda R}\left(\frac{1}{\lambda} + D\right)\left(\int_{\mathcal{D}_{W_j}} e^{\lambda(w+C)} f_{W_j}(w)dw - 1\right) \\
&= e^{\lambda R}\left(\frac{1}{\lambda} + D\right)\left(e^{\lambda C}\mathbb{E}[e^{\lambda W_j}] - 1\right) && (7)\\
&= e^{\lambda R}\left(\frac{1}{\lambda} + D\right)\left(e^{\lambda C}\prod_{i=i_{j-1}+1}^{i_j}\mathbb{E}[e^{\lambda X_i}] - 1\right) && (8)\\
&= e^{\lambda R}\left(\frac{1}{\lambda} + D\right)\left(e^{\lambda C}\mathbb{E}[e^{\lambda X}]^{i_j - i_{j-1}} - 1\right). && (9)
\end{aligned}$$

In Equation (7), $f_{W_j}(w)$ denotes the probability density function of W_j, which is the convolution of $i_j - i_{j-1}$ IID random variables following $\mathcal{D}$. Equation (8) holds because the random variables X_i are independent, and Equation (9) holds because they are identically distributed. Using the number of iterations $k_j = i_j - i_{j-1}$ included in W_j, we rewrite the expected cost of $\mathcal{S}$ as:

$$\mathbb{E}[MS(\mathcal{S})] = e^{\lambda R}\left(\frac{1}{\lambda} + D\right)\sum_{j=1}^{m}\left(e^{\lambda C}\mathbb{E}[e^{\lambda X}]^{k_j} - 1\right).$$

□

Note that $\mathbb{E}[e^{\lambda X}]$ is easy to compute for well-known distributions, and we give examples below. Equation (5) provides a closed-form formula to compute the expected makespan of a static solution. Recall that the principal Lambert function $\mathcal{W}_0$ is defined for $x \geq -\frac{1}{e}$ by $\mathcal{W}_0(x) = y$ if $ye^y = x$. The asymptotically optimal solution is given by the following theorem;

Theorem 1. *The periodic solution checkpointing every k_{static} iterations is asymptotically optimal, where*

$$x_{static} = \frac{\mathcal{W}_0(-e^{-\lambda C - 1}) + 1}{\log\left(\mathbb{E}[e^{\lambda X}]\right)} \tag{10}$$

and k_{static} is either $\max(1, \lfloor x_{static}\rfloor)$ or $\lceil x_{static}\rceil$, whichever achieves the smaller value of $C_{ind}(k)$ (computed by Equation (4)).

Proof. We first show that the function $C_{\text{ind}}(x)$ reaches its minimum for $x = x_{\text{static}}$:

Lemma 1. *The function $x \mapsto C_{ind}(x)$ is decreasing on $[0, x_{static}]$ and increasing on $[x_{static}, \infty)$ where x_{static} is defined by Equation* (10).

Proof. We differentiate and study the variations of C'_{ind}. We get

$$C'_{\text{ind}}(x) = \frac{xe^{\lambda C}\log\left(\mathbb{E}[e^{\lambda X}]\right)\mathbb{E}[e^{\lambda X}]^x - \left(e^{\lambda C}\mathbb{E}[e^{\lambda X}]^x - 1\right)}{x^2}.$$

Letting $y = x\log\left(\mathbb{E}[e^{\lambda X}]\right) - 1$, we have $e^y = \mathbb{E}[e^{\lambda X}]^x e^{-1}$ and we obtain

$$C'_{\text{ind}}(x) = \frac{e^{\lambda C}y\mathbb{E}[e^{\lambda X}]^x + 1}{x^2} = \frac{e^{\lambda C+1}ye^y + 1}{x^2}.$$

We derive that $C'_{\text{ind}}(x) \le 0 \Leftrightarrow ye^y \le -e^{-\lambda C-1}$. The function ye^y is an increasing function of y (and hence of x), and the equality is reached for $y = \mathcal{W}_0(-e^{-\lambda C-1})$ where $\mathcal{W}_0$ is the principal Lambert function. Finally, when $y = \mathcal{W}_0(-e^{-\lambda C-1})$, we have $x = x_{\text{static}}$.

Therefore, the function $C'_{\text{ind}}(x)$ has a unique zero x_{static}, is negative on $[0, x_{\text{static}}]$ and is positive on $[x_{\text{static}}, \infty)$. This shows that the function $C_{\text{ind}}(k)$ for integer values of k reaches its minimum either for $\max(1, \lfloor x_{\text{static}}\rfloor)$ or $\lceil x_{\text{static}}\rceil$, and we retrieve the definition of k_{static}. This concludes the proof of Lemma 1. □

We consider Equation (5) again and re-write it as

$$\begin{aligned}\mathbb{E}[MS(\mathcal{S})] &= e^{\lambda R}\left(\frac{1}{\lambda} + D\right)\sum_{j=1}^{m} k_j \cdot C_{\text{ind}}(k)\\ &= e^{\lambda R}\left(\frac{1}{\lambda} + D\right)\sum_{k=1}^{\infty} n_k \cdot k \cdot C_{\text{ind}}(k), \qquad (11)\end{aligned}$$

where n_k is the number of inter-checkpoint intervals with k iterations. We let $n_k = 0$ if there is no interval with k iterations, hence the infinite sum is well-defined.

We now introduce the periodic solution $\mathcal{S}_p$ that checkpoints every k_{static} iterations until the end of the execution, as long as there are at least k_{static} iterations left, and then checkpoints every remaining iteration. Formally, with an Euclidean division, letting $n_{\text{div}} = \lfloor n/k_{\text{static}}\rfloor$ and $n_{\text{mod}} = n \bmod k_{\text{static}}$, we have $n = n_{\text{div}}k_{\text{static}} + n_{\text{mod}}$ and $0 \le n_{\text{mod}} < k_{\text{static}}$. Hence the solution $\mathcal{S}_p$ has n_{div} intervals of k_{static} iterations, and the few remaining n_{mod} iterations, if any, are checkpointed individually. The expected makespan of $\mathcal{S}_p$ is

$$\begin{aligned}\mathbb{E}[MS(\mathcal{S}_p)] &= e^{\lambda R}\left(\frac{1}{\lambda} + D\right)\left(n_{\text{div}}k_{\text{static}}C_{\text{ind}}(k_{\text{static}}) + n_{\text{mod}}C_{\text{ind}}(1)\right)\\ &\le e^{\lambda R}\left(\frac{1}{\lambda} + D\right)\left(nC_{\text{ind}}(k_{\text{static}}) + (k_{\text{static}} - 1)C_{\text{ind}}(1)\right).\end{aligned}$$

From Equation (11), and because $C_{\text{ind}}(k_{\text{static}})$ is minimum over all possible values of k, we get

$$\mathbb{E}[MS(\mathcal{S}_{\text{opt}})] \ge e^{\lambda R}\left(\frac{1}{\lambda} + D\right)nC_{\text{ind}}(k_{\text{static}}).$$

Hence we can bound the ratio as follows:

$$\begin{aligned}\frac{\mathbb{E}[MS(\mathcal{S}_p)]}{\mathbb{E}[MS(\mathcal{S}_{\text{opt}})]} &\le \frac{nC_{\text{ind}}(k_{\text{static}}) + (k_{\text{static}} - 1)C_{\text{ind}}(1)}{nC_{\text{ind}}(k_{\text{static}})}\\ &= 1 + \frac{k_{\text{static}} - 1}{n}\frac{C_{\text{ind}}(1)}{C_{\text{ind}}(k_{\text{static}})} = 1 + O\left(\frac{1}{n}\right).\end{aligned}$$

This shows the asymptotic optimality of solution $\mathcal{S}_p$ and concludes the proof of Theorem 1. □

4.2 Instantiation for some distribution laws

We recall the definition of some well-known distributions laws that we use for $\mathcal{D}$, and show how to compute x_{static} for each of them.

Uniform law. Let X (the random variable for an iteration length) obey an Uniform distribution law Uniform(a, b) on $[a, b]$, where $0 < a < b$. The PDF (Probability Density Function) is $f(x) = \frac{1}{b-a}$ for $x \in [a, b]$. We have $\mu_{\mathcal{D}} = \frac{a+b}{2}$ and $\mathbb{E}[e^{\lambda X}] = \frac{e^{\lambda b} - e^{\lambda a}}{\lambda(b-a)}$, hence $x_{\text{static}} = \frac{\mathcal{W}_0(-e^{-\lambda C-1})+1}{\log\left(\frac{e^{\lambda b}-e^{\lambda a}}{\lambda(b-a)}\right)}$.

Gamma law. Let X obey a Gamma law Gamma(α, β), where $\alpha, \beta > 0$. The PDF is $f(x) = \frac{\beta^\alpha x^{\alpha-1}e^{-\beta x}}{\Gamma(\alpha)}$ for $x \ge 0$. We have $\mu_{\mathcal{D}} = \frac{\alpha}{\beta}$ and $\mathbb{E}[e^{\lambda X}] = \left(\frac{\beta}{\beta-\lambda}\right)^\alpha$, hence $x_{\text{static}} = \frac{\mathcal{W}_0(-e^{-\lambda C-1})+1}{\alpha\log\left(\frac{\beta}{\beta-\lambda}\right)}$. Note that a Gamma law Gamma$(1, \beta)$ is an Exponential law of parameter β.

Normal law. Let X obey a Normal law Normal(μ, σ^2), where $\mu, \sigma > 0$. The PDF is $f(x) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$. We have $\mu_{\mathcal{D}} = \mu$ and $\mathbb{E}[e^{\lambda X}] = e^{\lambda\mu + \frac{\lambda^2\sigma^2}{2}}$, hence $x_{\text{static}} = \frac{\mathcal{W}_0(-e^{-\lambda C-1})+1}{\lambda\mu + \frac{\lambda^2\sigma^2}{2}}$.

Simulations. In the experiments in Section 6, we randomly sample $\mathcal{D}$ to compute the length of each iteration. For Normal distributions Normal(μ, σ^2), we take $\mu \gg 0$ and sample the distribution until we get a positive value.

4.3 First-order approximation

In this section, we show that the first-order approximation (i.e. when the failure rate is very low in front of the distribution parameters) of k_{static} leads to the Young/Daly formula. This result holds for all distributions with finite expectation $\mathbb{E}[e^{\lambda X}]$, hence for all classic distributions. More precisely, we have:

Proposition 2. *The first-order approximation k_{FO} of k_{static} obeys the equation*

$$k_{FO} \cdot \mu_{\mathcal{D}} = \sqrt{\frac{2C}{\lambda}}.$$

Proposition 2 shows that (the first order approximation of) the average period length of the optimal periodic solution, namely k_{FO} iterations of expected length $\mu_{\mathcal{D}}$, is equal to the Young/Daly period. Note that this result is not surprising but reassuring. Essentially it says that when the inter-arrival time between failure is large in front of the distribution parameters (mean, variance), this distribution can be approximated by a deterministic distribution of size $\mu_{\mathcal{D}}$ to compute the optimal interval size.

Proof. We use Taylor expansions to solve the equation giving the zero of the function $C_{\text{ind}}(k)$, namely

$$e^{\lambda C}\left(k\log\left(\mathbb{E}[e^{\lambda X}]\right) - 1\right)e^{k\log\left(\mathbb{E}[e^{\lambda X}]\right)} = -1. \qquad (12)$$

We successively derive that

$$\mathbb{E}[e^{\lambda X}] = 1 + \mathbb{E}[X]\,\lambda + \frac{1}{2}\mathbb{E}\left[X^2\right]\lambda^2 + o\left(\lambda^2\right), \tag{13}$$

$$\begin{aligned}\log \mathbb{E}\left[e^{\lambda X}\right] &= \lambda\mathbb{E}[X] + \frac{\lambda^2}{2}\mathbb{E}\left[X^2\right] - \frac{\left(\lambda\mathbb{E}[X] + \frac{\lambda^2}{2}\mathbb{E}\left[X^2\right]\right)^2}{2} + o\left(\lambda^2\right)\\ &= \mathbb{E}[X]\,\lambda + \frac{1}{2}\left(\mathbb{E}\left[X^2\right] - \mathbb{E}[X]^2\right)\lambda^2 + o\left(\lambda^2\right), \end{aligned}\tag{14}$$

$$\begin{aligned}e^{k\log\mathbb{E}[e^{\lambda X}]} &= \left(\mathbb{E}[e^{\lambda X}]\right)^k = \left(1 + \lambda\mathbb{E}[X] + \frac{\lambda^2}{2}\mathbb{E}\left[X^2\right]\right)^k + o\left(\lambda^2\right)\\ &= 1 + k\mathbb{E}[X]\,\lambda + \frac{k}{2}\left(\mathbb{E}\left[X^2\right] + (k-1)\mathbb{E}[X]^2\right)\lambda^2 + o\left(\lambda^2\right). \end{aligned}\tag{15}$$

By plugging Equations (14) and (15) in Equation (12), we have

$$\begin{aligned}&\left(1 + C\lambda + \tfrac{C^2}{2}\lambda^2\right)\left(k\mathbb{E}[X]\,\lambda + \tfrac{k\mathbb{E}[X^2]}{2}\lambda^2 - \tfrac{k\mathbb{E}[X]^2}{2}\lambda^2 - 1\right)\\ &\quad\times\left(1 + k\mathbb{E}[X]\,\lambda + \tfrac{k\mathbb{E}[X^2]}{2}\lambda^2 + \tfrac{k(k-1)\mathbb{E}[X]^2}{2}\lambda^2\right) = -1.\end{aligned}$$

After simplification, we obtain $\frac{k^2}{2}\mathbb{E}[X]^2\lambda^2 - C\lambda = o(\lambda)$, hence $k_{\text{FO}}\mathbb{E}[X] = \sqrt{\frac{2C}{\lambda}}$, which corresponds to the Young/Daly formula. □

Simulations. In the experiments in Section 6, we use

$$k_{\text{FO}} = \max\left(1, \text{round}\left(\frac{1}{\mu_{\mathcal{D}}}\sqrt{\frac{2C}{\lambda}}\right)\right), \tag{16}$$

where round(x) rounds x to the closest integer.

5 DYNAMIC STRATEGIES

In Section 4, we have studied static solutions where checkpoint locations are decided before the execution. These static decisions are made based upon the distribution $\mathcal{D}$ and the fault rate, but do not depend on the actual length of the iterations in a specific instance of the problem. However, when executing the application, we know on the fly whether some iteration has been much shorter, or much longer, than the average iteration length, and we could take this information into account to decide whether to checkpoint or not. In other words, we take dynamic decisions, at the end of each iteration, and these decisions are based upon the actual work executed since the last checkpoint.

5.1 Asymptotic optimality

The dynamic strategy discussed in this section can be stated as follows:

- We fix a threshold W_{th} for the amount of work since the last checkpoint.
- When iteration X_i finishes, if the amount of work since the last checkpoint is greater than W_{th}, then $\delta_i = 1$ (we checkpoint) otherwise $\delta_i = 0$ (we do not checkpoint).

The objective is to determine the value of W_{th} that minimizes the expected execution time of this strategy. However, the expected execution time is much harder to write than for static strategies since the δ_i are now conditional to the values of the X_i. Instead, we make dynamic decisions at the end of each iteration based upon the overhead of the decision (to checkpoint or not). For applications with a large number n of iterations, we minimize the overhead at each step by progressing this way, and always checkpoint the last iteration. This enforces the asymptotic optimality of the strategy when n tends to infinity.

The slowdown H is defined as the ratio

$$H = \frac{\text{actual execution time}}{\text{useful execution time}},$$

so that the slowdown is equal to 1 if there is no cost for fault-tolerance (checkpoints, and re-execution after failures). When an iteration is completed, we compute two values:

- The expected slowdown H_{ckpt} if a checkpoint is taken at the end of this iteration;
- The expected slowdown H_{no} if no checkpoint is taken at the end of this iteration.

The rationale is the following: If $H_{\text{ckpt}} < H_{\text{no}}$, it is better to checkpoint now than waiting for the end of the next iteration, and by induction, than waiting for the end of two or more following iterations. On the contrary, if $H_{\text{no}} < H_{\text{ckpt}}$, it is better not to checkpoint now, in which case we recompute the decision at the end of the next iteration.

We now show how to compute H_{ckpt} and H_{no}. We assume that we just finished an iteration, and that the total amount of work since the last checkpoint (including the last iteration) is w_{dyn}, and write $H_{\text{ckpt}}(w_{\text{dyn}})$ and $H_{\text{no}}(w_{\text{dyn}})$ for the two slowdowns:

Computing H_{ckpt}. Recall that Equation (1) gives $T_\lambda(W, C, D, R)$, the expected execution time to perform a work of size W followed by a checkpoint of size C, with downtime D and recovery R. The expected time to compute w_{dyn} was $T(w_{\text{dyn}}, 0, D, R)$, and the expected time to checkpoint now is $T(0, C, D, R + w_{\text{dyn}})$: this is because if a failure strikes during the checkpoint, we have to reexecute w_{dyn}. Finally, the useful execution time is w_{dyn}, hence

$$\begin{aligned}H_{\text{ckpt}}(w_{\text{dyn}}) &= \frac{T(w_{\text{dyn}}, 0, D, R) + T(0, C, D, R + w_{\text{dyn}})}{w_{\text{dyn}}}\\ &= e^{\lambda R}\left(\frac{1}{\lambda} + D\right)\frac{(e^{\lambda w_{\text{dyn}}} - 1) + e^{\lambda w_{\text{dyn}}}\left(e^{\lambda C} - 1\right)}{w_{\text{dyn}}}\\ &= e^{\lambda R}\left(\frac{1}{\lambda} + D\right)\frac{e^{\lambda(w_{\text{dyn}}+C)} - 1}{w_{\text{dyn}}}. \end{aligned}\tag{17}$$

Computing H_{no}. If we do not checkpoint now but only at the end of the next iteration of length $X = w$ (drawn from distribution $\mathcal{D}$), the actual execution time will be $T(w_{\text{dyn}}, 0, D, R) + T(w, C, D, R + w_{\text{dyn}})$ and the useful time will be $w_{\text{dyn}} + w$. Hence we need to take the expectation of the ratio and obtain

$$\begin{aligned}H_{\text{no}}(w_{\text{dyn}}) &= \mathbb{E}\left(\frac{T(w_{\text{dyn}}, 0, D, R) + T(X, C, D, R + w_{\text{dyn}})}{w_{\text{dyn}} + X}\right)\\ &= \int_D \frac{T(w_{\text{dyn}}, 0, D, R) + T(w, C, D, R + w_{\text{dyn}})}{w_{\text{dyn}} + w} f(w)dw, \end{aligned}\tag{18}$$

where $f(x)$ is the PDF of $\mathcal{D}$ and $\mathcal{D}$ is its domain. Computing the expectation of this ratio is too difficult, and we approximate it by taking the ratio of the expectations (actual time over useful time),

so we redefine H_{no} by

$$H_{\text{no}}(w_{\text{dyn}}) = \frac{\mathbb{E}\left[T(w_{\text{dyn}},0,D,R) + T(X,C,D,R+w_{\text{dyn}})\right]}{\mathbb{E}\left[w_{\text{dyn}}+X\right]}$$
$$= \frac{T(w_{\text{dyn}},0,D,R) + \mathbb{E}\left[T(X,C,D,R+w_{\text{dyn}})\right]}{w_{\text{dyn}}+\mathbb{E}[X]}$$
$$= e^{\lambda R}\left(\frac{1}{\lambda}+D\right)\frac{(e^{\lambda w_{\text{dyn}}}-1) + e^{\lambda w_{\text{dyn}}}\left(e^{\lambda C}\mathbb{E}[e^{\lambda X}]-1\right)}{w_{\text{dyn}}+\mathbb{E}[X]}$$
$$= e^{\lambda R}\left(\frac{1}{\lambda}+D\right)\frac{e^{\lambda(w_{\text{dyn}}+C)}\mathbb{E}[e^{\lambda X}]-1}{w_{\text{dyn}}+\mathbb{E}[X]} \quad (19)$$

The last line of Equation (19) is obtained using Equation (7).

Computing W_{th} . By defintion, W_{th} is the threshold value where

$$H_{\text{ckpt}}(W_{th}) = H_{\text{no}}(W_{th})$$

Using Equations (17) and (19), we obtain

$$W_{th}\left(e^{\lambda(W_{th}+C)}\mathbb{E}[e^{\lambda X}]-1\right) = (W_{th}+\mathbb{E}[X])\left(e^{\lambda(W_{th}+C)}-1\right). \quad (20)$$

After simplification, we have

$$\left(\left(\mathbb{E}[e^{\lambda X}]-1\right)W_{th} - \mathbb{E}[X]\right)e^{\lambda(W_{th}+C)} = -\mathbb{E}[X],$$

by multipling $\frac{\lambda}{\mathbb{E}[e^{\lambda X}]-1}e^{-\lambda\left(C+\frac{\mathbb{E}[X]}{\mathbb{E}[e^{\lambda X}]-1}\right)}$ on both sides of the equation, we have

$$te^{t} = -\frac{\lambda\mathbb{E}[X]}{\mathbb{E}[e^{\lambda X}]-1}e^{-\lambda\left(C+\frac{\mathbb{E}[X]}{\mathbb{E}[e^{\lambda X}]-1}\right)},$$

where $t = \lambda W_{th} - \frac{\lambda\mathbb{E}[X]}{\mathbb{E}[e^{\lambda X}]-1}$. Finally, we derive the threshold value:

$$W_{th} = \frac{1}{\lambda}\mathcal{W}_0\left(-\frac{\lambda\mathbb{E}[X]}{\mathbb{E}[e^{\lambda X}]-1}e^{-\lambda\left(C+\frac{\mathbb{E}[X]}{\mathbb{E}[e^{\lambda X}]-1}\right)}\right) + \frac{\mathbb{E}[X]}{\mathbb{E}[e^{\lambda X}]-1}. \quad (21)$$

5.2 First-order approximation

In this section, we show that the first-order approximation of W_{th} leads to the Young/Daly formula. This result holds for all distributions with finite expectation $\mathbb{E}[e^{\lambda X}]$, hence for all classic distributions. More precisely, we have:

Proposition 3. *The first-order approximation W_{FO} of W_{th} obeys the equation*

$$W_{FO} = \sqrt{\frac{2C}{\lambda}}. \quad (22)$$

Equation (22) shows that the first order approximation of the threshold value W_{th} , namely W_{FO}, is equal to the Young/Daly period.

Proof. We use Taylor expansions to solve Equation(20). After simplification, we have

$$we^{\lambda(w+C)}\left(\mathbb{E}\left[e^{\lambda X}\right]-1\right) = \mathbb{E}[X]\left(e^{\lambda(w+C)}-1\right), \quad (23)$$

by plugging Equations (13) in Equation (23), we have

$$w\left(1+\lambda(w+C)+\frac{\lambda^2(w+C)^2}{2}\right)\left(\lambda E[X]+\frac{\lambda^2 E\left[X^2\right]}{2}\right)$$
$$= E[X]\left(\lambda(w+C)+\frac{\lambda^2(w+C)^2}{2}\right).$$

After simplification, we obtain $\frac{\lambda w^2}{2} - C\lambda = o(\lambda)$, hence $W_{\text{FO}} = \sqrt{\frac{2C}{\lambda}}$, which corresponds to the Young/Daly formula. □

Simulations. In the experiments in Section 6, we use Equation (22).

6 EXPERIMENTS

In this section, we describe the experiments conducted to assess the efficiency of static and dynamic solutions, as well as the accuracy of the Young/Daly formula. Propositions 2 and 3, show that when the number of failures is low, the Young/Daly formula is a good approximation. We aim at showing experimentally that this remains the case with higher failure rates, when the first-order approximation is no longer valid. In Subsection 6.1, we detail the experimental methodology with all simulation parameters. Results are presented in Subsection 6.2.

6.1 Experimental methodology

For each experiment, the evaluations are performed on 10,000 randomly generated instances $\{\mathcal{I}_1,\ldots,\mathcal{I}_{10000}\}$. For all i, an instance $\mathcal{I}_i$ is a pair $(\mathcal{S}_i,\mathcal{F}_i)$, where $\mathcal{S}_i$ (resp. $\mathcal{F}_i$) is the application (resp. failure) scenario associated to the instance.

The algorithms are implemented in MATLAB and R. The corresponding code is available at [11]. This simulator computes the makespan for our static strategy, the Young/Daly-static strategy, our dynamic strategy, and the Young/Daly-dynamic strategy.

Application scenarios. We consider an iterative application composed of $n = 1,000$ consecutive iterations[2]. We assume that the execution time of each iteration follows a probability distribution $\mathcal{D}$, where $\mathcal{D}$ is either Uniform(a,b), Gamma(α,β) or truncated Normal$(\mu,\sigma^2,[0,\infty))$ (see Section 4.2 for the corresponding PDFs). The default instantiations for these distributions are $\mu_{\mathcal{D}} = 50$ with Uniform$[20,80]$, Gamma$(25,0.5)$ and Normal$(50,2.5^2)$ (recall that we sample the latter one until a positive value is found). We also study the impact of the standard deviation σ.

Failure scenarios. We consider different failure rates. To allow for consistent comparisons of results across different iterative processes with different probability distributions, we fix the probability that failure occurs during each iteration, which we denote at p_{fail}, and then simulate the corresponding failure rate. Formally, for a given p_{fail} value, we compute the failure rate λ such that $p_{\text{fail}} = 1 - e^{-\lambda(\mu_{\mathcal{D}}+C)}$, where $\mu_{\mathcal{D}}+C$ is the average length of an iteration followed by a checkpoint. We conduct experiments for seven p_{fail} values: 10^{-3}, $10^{-2.5}$, 10^{-2}, $10^{-1.5}$, 10^{-1}, $10^{-0.5}$ and $10^{-0.1}$. For example, $p_{\text{fail}} = 10^{-2}$ means one failure may occur every 100 iterations.

[2]The experiments show very good stability already with $n = 1,000$. We compared the results with $n = 10,000$ before keeping $n = 1,000$.

Checkpointing costs. Important factors that influence the performance of checkpointing strategies are the checkpointing and recovery costs. We set checkpoint time as $C = \eta\mu_{\mathcal{D}}$, where η is the proportion of checkpoint time to the expectation of iteration time. And we set recovery time[3] as $R = C$, and fixed downtime as $D = 1$. We conducted the experiments with $\eta = 0.1$.

Static strategies. For an instance $\mathcal{I}$, we define $MS_{sim\_sta}(k)(\mathcal{I})$ to be the makespan when checkpointing every k iterations. In addition we define the average value

$$\overline{MS}_{sim\_sta}(k) = \frac{1}{10000}\sum_{i=1}^{10000} MS_{sim\_sta}(k)(\mathcal{I}_i)$$

and the minimal average makespan over k:

$$MS^{\min}_{sim\_sta} = \min_k \overline{MS}_{sim\_sta}(k)$$

This minimum is reached for $k = k_{\text{sim}}$.

We also compare the simulations with the theoretical model. We use $\mathbb{E}[MS_{\mathcal{D}}](k) = n \cdot e^{\lambda R}\left(\frac{1}{\lambda} + D\right) C_{ind}(k)$, where C_{ind} depends on $\mathcal{D}$, and $n = 1,000$. We define $MS^{OPT}_{the\_sta} = \mathbb{E}[MS_{\mathcal{D}}](k_{\text{static}})$. Finally, we define the Young/Daly static as $MS_{YD\_sta} = MS_{sim\_sta}(k_{\text{FO}})$ and $\overline{MS}_{YD\_sta}$ as its average value over all instances.

Dynamic strategies. We simulate the dynamic strategy with different threshold values $W = \gamma \cdot W_{th}$ with $\gamma \in \{0.1, 0.2, \ldots, 2\}$. For an instance $\mathcal{I}$, we define $MS_{sim\_dyn}(W)(\mathcal{I})$ as the makespan with threshold W, and $\overline{MS}_{sim\_dyn}(W)$ as its average value over all instances. Then we let $MS^{OPT}_{sim\_dyn} = \min_W \overline{MS}_{sim\_dyn}(W)$. It is reached for $W = W_{\text{sim}}$. Finally, we define the Young/Daly dynamic as $MS_{YD\_dyn} = MS_{sim\_dyn}(W_{\text{FO}})$ and $\overline{MS}_{YD\_dyn}$ as its average value over all instances.

6.2 Results

Due to space limitations, we only report here a subset of our simulation results (see the extended version [12]). For instance, while we report synthetic results for all distributions, we only comment on the Gamma distribution because results are similar for the Normal distribution and the Uniform distribution.

Table 1: Simulation for static case.

$p_{\text{fail}} = 10^{-2}$	Gamma	Normal	Uniform
k_{sim}	5	5	5
x_{static}	4.6114	4.6122	4.6097
k_{static}	5	5	5
$\frac{1}{\mu_{\mathcal{D}}}\sqrt{\frac{2C}{\lambda}}$	4.6787	4.6787	4.6787
k_{FO}	5	5	5

[3]We let $C = R$ for all experiments because the value of R has no impact on the optimal checkpointing strategy (see Equations (11), (17) and (19)), This is not surprising: the values of D and R are costs that happen if and only if there is a failure. Of course these values impact expected execution time, and the larger they are, the more difference between a suboptimal strategy and our optimal strategy.

Static strategies. The results from the static case are reported in Figure 1. Specifically, the box plots represent the evolution of the function $\mathcal{I} \mapsto \frac{MS_{sim\_sta}(k)}{MS_{YD\_sta}}(\mathcal{I})$ (for different values of k), and the black lines correspond to its mean. The diamonds represent the average: $\frac{1}{10000}\sum_{i=1}^{10000}\frac{\mathbb{E}[MS_{\mathcal{D}}](k)}{MS_{YD\_sta}(\mathcal{I}_i)}$.

The first important result from this plot is the experimental validation of our model. Indeed, the blacklines and diamonds are almost identical for all k. The closer we get to the optimal value k_{static}, the closer the theoretical makespan gets to the simulation makespan. In particular, for $k = 5$, which corresponds to k_{FO} (and k_{static}), the makespan obtained is exactly the same for MS_{sim_sta} and MS_{YD_sta}, leading to a ratio of 1 in all cases: the boxplot contains a single value.

A consequence is that the solution k_{static} (as well as Young/Daly's solution) always provides the optimal expected makespan, in coherence with the theoretical results. Because it is a stochastic process, it can not always give the optimal makespan, but we see from these figures that it is always within 3% of the makespan obtained by other strategies, which shows the robustness of this choice.

As expected the ratio $MS^{OPT}_{the\_sta}/MS_{YD\_sta}$ is equal to 1 since in those cases $k = 5$ for k_{static} and k_{FO}. We have tried a large range of values to check if there are cases when they are not and have found that they almost always are (see Figures 3 and 4 and comments).

In order to compare static strategies with dynamic strategies, we plot blue and red lines corresponding to the ratios $\overline{MS}_{sim\_dyn}(W_{th})/\overline{MS}_{YD\_sta}$, and $\overline{MS}_{YD\_dyn}/\overline{MS}_{YD\_sta}$, respectively. Both lines are very close to 1, meaning that these two dynamic strategies have the same performance as the optimal static strategy.

Overall the conclusions of this section is that the simple strategy based on the Young/Daly setting remains a good and robust solution for stochastic applications, and can safely be used in this context.

Table 2: Simulation for dynamic case.

$p_{\text{fail}} = 10^{-2}$	Gamma	Normal	Uniform
γ	1.0	1.0	1.0
W_{sim}	206.0492	206.8876	204.2743
$MS^{OPT}_{sim\_dyn}$	52267	52264	52267
W_{th}	206.0492	206.8876	204.2743
$\overline{MS}_{sim\_dyn}(W_{th})$	52267	52264	52267
W_{FO}	233.9328	233.9328	233.9328
$\overline{MS}_{YD\_dyn}$	52284	52271	52288

Dynamic strategies. We compare our dynamic strategy with the threshold obtained with the Young/Daly formula. For each γ, we report the makespan of 10,000 random simulations using boxplots. From Table 2, it can be observed that $W_{\text{sim}} = W_{th}$. Of course, giving more precision to γ may give slightly better performance, but the gain remains negligible. Contrarily to the static case, W_{th} and W_{FO} are different (up to 15%). However the performance obtained (Figure 2) are similar, and again the Young/Daly formula seems a safe bet given its simplicity of use.

In Figure 2, we plot orange and purple lines corresponding to the ratios $\overline{MS}_{sim\_sta}(k_{\text{static}})/\overline{MS}_{YD\_dyn}$ and $\overline{MS}_{YD\_sta}/\overline{MS}_{YD\_dyn}$, respectively. As in Figure 1, these ratios are very close to 1, meaning

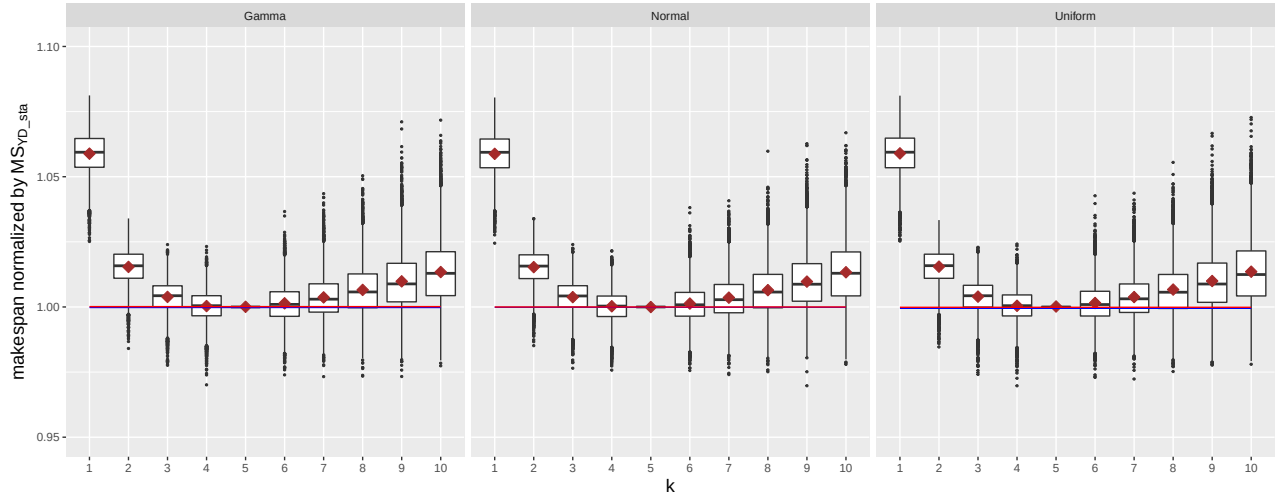

Figure 1: Performance (with boxplots) of the static strategy that chooses the value of k. Brown-red diamonds plot $\mathbb{E}[MS_{\mathcal{D}}](k)$ (theoretical makespan). The blue (resp. red) line represents the makespan obtained by the optimal dynamic strategy $\overline{MS}_{sim\_dyn}(W_{th})$ (resp. the YD-dynamic strategy $\overline{MS}_{YD\_dyn}$).

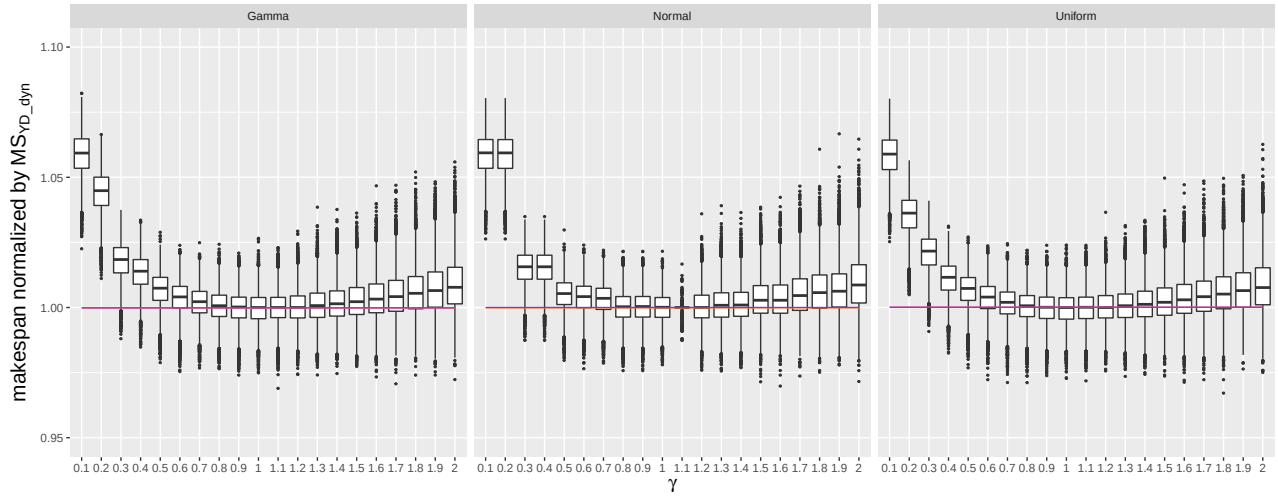

Figure 2: Performance (with boxplots) of the dynamic strategy that chooses a threshold of $\gamma \cdot W_{th}$. The orange (resp. purple) line represents the makespan obtained by the optimal static strategy $\overline{MS}_{sim\_sta}(k_{\textbf{static}})$ (resp. the YD-static strategy $\overline{MS}_{YD\_sta}$).

that the static and the dynamic strategies give similar results both when using optimal parameters or the one approximated using the Young/Daly formulas.

Both strategies for varying p_{fail}. In order to study the sensibility of our results to the failure probability, we compare in Figure 3 the makespan obtained by the static Young/Daly approximation ($\overline{MS}_{YD\_sta}$) to the makespan obtained by the simulation when using the optimal k_{static} ($\overline{MS}_{sim\_sta}(k_{\text{static}})$), and the one of the optimal dynamic strategy ($\overline{MS}_{sim\_dyn}(W_{th})$). We observe that the first two makespans are always equal, because in all cases $k_{\text{FO}} = k_{\text{static}}$. The optimal dynamic strategy is sometimes slightly better that the static ones, but with a gap smaller than 0.5% for all failure probabilities.

Both strategies for varying σ. We vary the standard deviation σ of each distribution of execution times in Figure 4. Again, there is no difference between k_{static} and k_{FO} in all tested cases, leading to a ratio $\overline{MS}_{sim\_sta}(k_{\text{static}})/\overline{MS}_{YD\_sta}$ constant and equal to 1. The optimal dynamic strategy is again very close, with a gap smaller than 0.05% even for very large deviations.

Both strategies for varying η. Finally, we vary the proportion η of checkpoint time to expected iteration time in Figure 5 (for Gamma

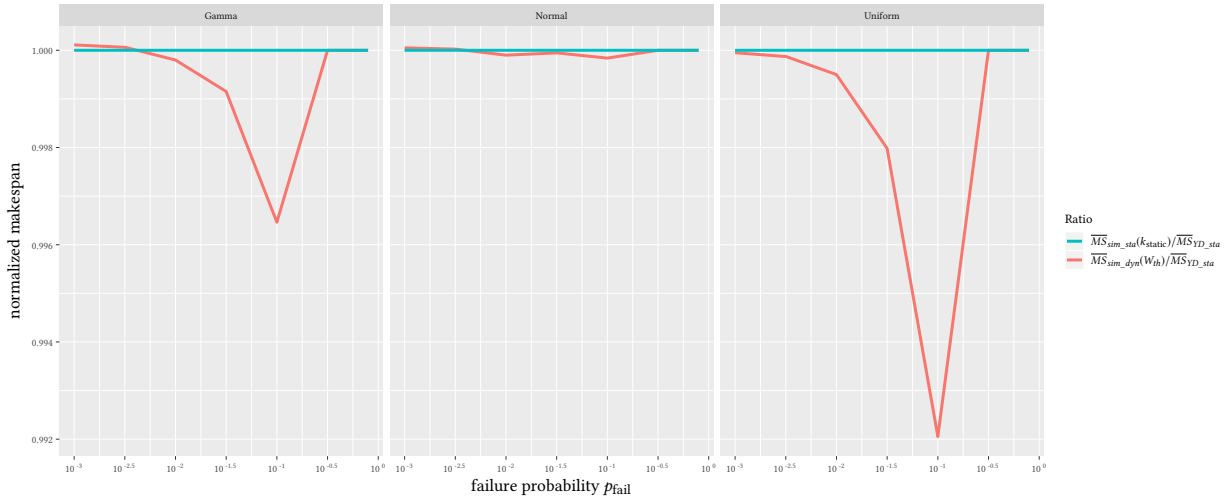

Figure 3: Simulation with varying failure probability.

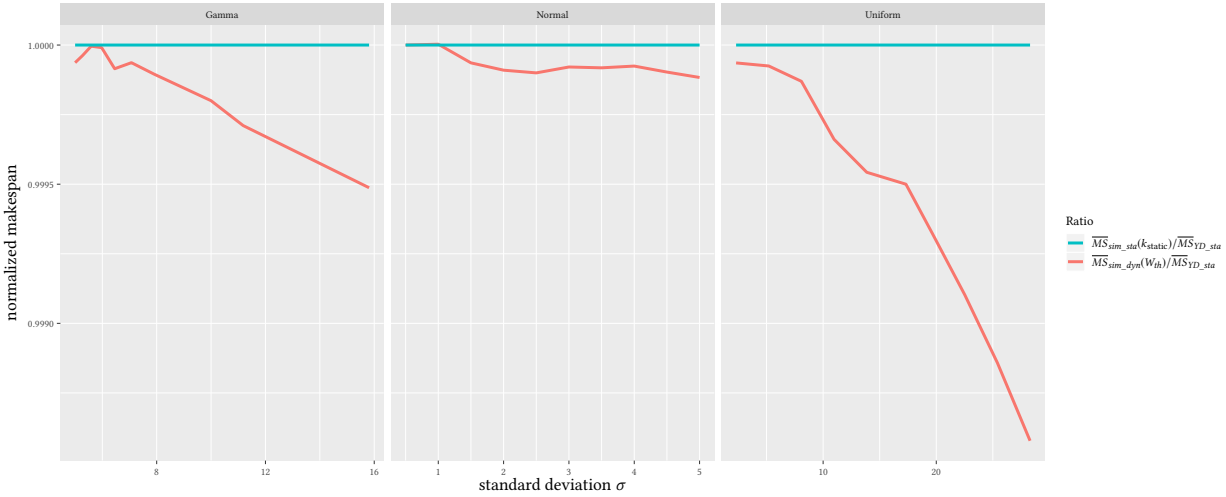

Figure 4: Simulation with varying standard deviation.

distributions, see [12] for the other distributions). As expected, both the optimal k and W increase together with checkpoint time. The optimal static and dynamic strategies are still very close, with a gap larger or smaller than 0.05%, even for very large checkpoint times.

7 CONCLUSION

We have introduced and analyzed checkpointing strategies for iterative applications. The key novelty is that this work does not assume deterministic iterations, but instead models execution times with probabilistic distributions. Our first main contribution is to provide a closed-form formula, valid for any distribution, to compute the optimal period at which one should checkpoint as a function of the failure rate. Then, we provide efficient solutions for non periodic, online solutions, where one decides on the fly whether to perform a checkpoint or to perform an additional iteration. In addition to these solutions, we study the behavior of the Young/Daly solution. We then show the following: as a first-order approximation, both periodic and non periodic solutions converge to the Young/Daly formula. All these derivations are quite technical, and constitute a major extension of the deterministic case.

In addition, we are able to show via extensive simulations that the Young/Daly formula is in general an excellent solution for non-deterministic execution times. This is done in two steps: (i) we show that our mathematical model is extremely accurate, since the mathematical formula fits almost perfectly the evaluated execution time; and (ii) the performance of the Young/Daly formula is always within one percent that of the optimal strategy that we obtained.

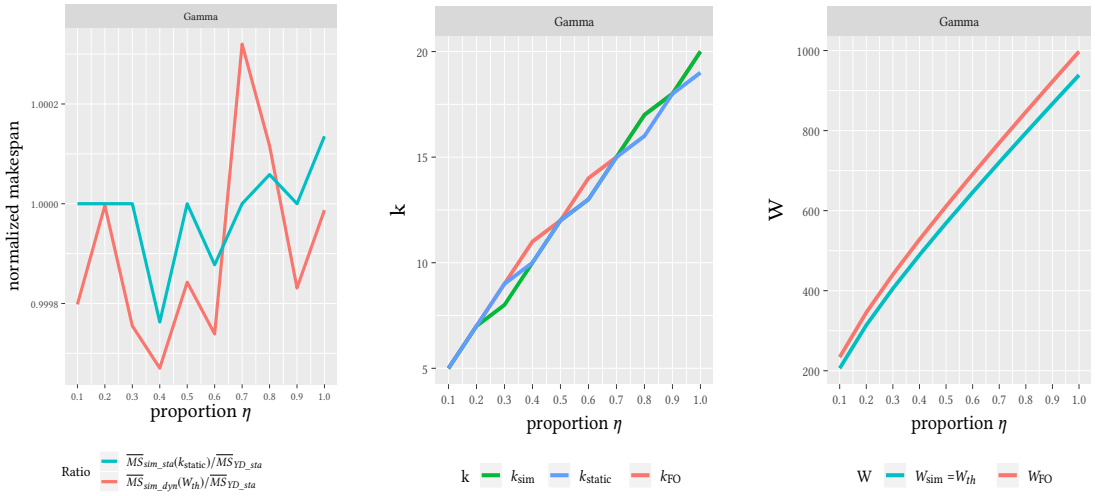

Figure 5: Simulation with varying the proportion of checkpoint time η to the expected iteration time.

Further work will be devoted to extending this study to multi-level checkpointing protocols, which correspond to state-of-the-art approaches but are already quite difficult to model and optimize analytically in a deterministic setting. Extending know results to a stochastic framework is a challenging problem.

REFERENCES

[1] Z.-Z. Bai and W.-T. Wu. On greedy randomized kaczmarz method for solving large sparse linear systems. *SIAM J. Sci. Comput.*, 40(1):A592–A606, 2018.

[2] Z.-Z. Bai and W.-T. Wu. On greedy randomized coordinate descent methods for solving large linear least-squares problems. *Numer. Linear Algebr. Appl.*, 26(4):1–15, 2019.

[3] A. Benoit, A. Cavelan, V. Le Fèvre, Y. Robert, and H. Sun. Towards optimal multi-level checkpointing. *IEEE Trans. Computers*, 66(7):1212–1226, 2017.

[4] V. Bharadwaj, D. Ghose, and T. Robertazzi. Divisible load theory: a new paradigm for load scheduling in distributed systems. *Cluster Computing*, 6(1):7–17, 2003.

[5] R. H. Bisseling and A.-J. N. Yzelman. Thinking in sync: The bulk-synchronous parallel approach to large-scale computing. *ACM Computing reviews*, 57(6):322–327, 2016.

[6] D. Blackston and T. Suel. Highly portable and efficient implementations of parallel adaptive n-body methods. In *Proc. ACM Supercomputing*, pages 1–20, 1997.

[7] F. Cappello, A. Geist, B. Gropp, L. Kale, B. Kramer, and M. Snir. Toward Exascale Resilience. *Int. J. High Performance Computing Applications*, 23(4):374–388, 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] J. T. Daly. A higher order estimate of the optimum checkpoint interval for restart dumps. *Future Generation Comp. Syst.*, 22(3):303–312, 2006.

[10] S. Di, M. S. Bouguerra, L. Bautista-Gomez, and F. Cappello. Optimization of multi-level checkpoint model for large scale HPC applications. In *IPDPS*. IEEE, 2014.

[11] Y. Du. Code for simulations. https://figshare.com/s/0077c4bd17925d412f97.

[12] Y. Du, L. Marchal, G. Pallez (Aupy), and Y. Robert. Robustness of the Young/Daly formula for stochastic iterative applications. Research report RR-9332, INRIA, Mar. 2020.

[13] A. Gainaru and G. Pallez. Making speculative scheduling robust to incomplete data. In *IEEE/ACM 10th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA)*, pages 62–71, 2019.

[14] A. V. Gerbessiotis and L. G. Valiant. Direct bulk-synchronous parallel algorithms. *J. Parallel Distributed Computing*, 22(2):251–267, 1994.

[15] R. M. Gower and P. Richtárik. Randomized iterative methods for linear systems. *SIAM Journal on Matrix Analysis and Applications*, 36(4):1660–1690, 2015.

[16] L. Han, L.-C. Canon, H. Casanova, Y. Robert, and F. Vivien. Checkpointing workflows for fail-stop errors. *IEEE Trans. Computers*, 67(8):1105–1120, 2018.

[17] L. Han, V. Le Fèvre, L.-C. Canon, Y. Robert, and F. Vivien. A Generic Approach to Scheduling and Checkpointing Workflows. *Int. Journal of High Performance Computing Applications*, 33(6):1255–1274, 2019.

[18] P. Hanrahan, D. Salzman, and L. Aupperle. A rapid hierarchical radiosity algorithm. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 197–206, 1991.

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

[20] J. M. Hill, B. McColl, D. C. Stefanescu, M. W. Goudreau, K. Lang, S. B. Rao, T. Suel, T. Tsantilas, and R. H. Bisseling. BSPlib: The BSP programming library. *Parallel Computing*, 24(14):1947–1980, 1998.

[21] D. Leventhal and A. S. Lewis. Randomized methods for linear constraints: convergence rates and conditioning. *Math. Operations Research*, 35(3):641–654, 2010.

[22] A. Ma, D. Needell, and A. Ramdas. Convergence properties of the randomized extended gauss–seidel and kaczmarz methods. *SIAM Journal on Matrix Analysis and Applications*, 36(4):1590–1604, 2015.

[23] I. Masliah, A. Abdelfattah, A. Haidar, S. Tomov, M. Baboulin, J. Falcou, and J. Dongarra. Algorithms and optimization techniques for high-performance matrix-matrix multiplications of very small matrices. *Parallel Computing*, 81:1–21, 2019.

[24] A. Moody, G. Bronevetsky, K. Mohror, and B. R. d. Supinski. Design, modeling, and evaluation of a scalable multi-level checkpointing system. In *SC*. ACM, 2010.

[25] D. Petcu. The performance of parallel iterative solvers. *Computers and Mathematics with Applications*, 50(7):1179–1189, 2005.

[26] T. Robertazzi. Ten reasons to use divisible load theory. *IEEE Computer*, 36(5):63–68, 2003.

[27] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2nd edition, 2003.

[28] T. Strohmer and R. Vershynin. A randomized kaczmarz algorithm with exponential convergence. *J. Fourier Analysis and Applications*, 15(2):262, 2009.

[29] Top500. Top 500 Supercomputer Sites, November 2018. https://www.top500.org/lists/2018/11/.

[30] S. Toueg and O. Babaoğlu. On the optimum checkpoint selection problem. *SIAM J. Comput.*, 13(3), 1984.

[31] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 8(3):410–421, 1979.

[32] J. W. Young. A first order approximation to the optimum checkpoint interval. *Comm. of the ACM*, 17(9):530–531, 1974.