When to checkpoint at the end of a fixed-length reservation?

Quentin Barbut, Anne Benoit, Thomas Herault, Yves Robert, Frédéric Vivien

When to checkpoint at the end of a fixed-length reservation?

Quentin Barbut
Laboratoire LIP, ENS Lyon & Inria
Lyon, France
quentin.barbut@ens-lyon.fr

Anne Benoit
Laboratoire LIP, ENS Lyon & Inria
Lyon, France
anne.benoit@ens-lyon.fr

Thomas Herault
University of Tennessee
Knoxville, TN, USA
thomas.herault@icl.utk.edu

Yves Robert*
Laboratoire LIP, ENS Lyon & Inria
Lyon, France
yves.robert@ens-lyon.fr

Frédéric Vivien
Laboratoire LIP, ENS Lyon & Inria
Lyon, France
frederic.vivien@inria.fr

FTXS'23, November 2023, Denver, CO, USA

ABSTRACT

This work considers an application executing for a fixed duration, namely the length of the reservation that it has been granted. The checkpoint duration is a stochastic random variable that obeys some well-known probability distribution law. The question is when to take a checkpoint towards the end of the execution, so that the expectation of the work done is maximized. We address two scenarios. In the first scenario, a checkpoint can be taken at any time; despite its simplicity, this natural problem has not been considered yet (to the best of our knowledge). We provide the optimal solution for a variety of probability distribution laws modeling checkpoint duration. The second scenario is more involved: the application is a linear workflow consisting of a chain of tasks with IID stochastic execution times, and a checkpoint can be taken only at the end of a task. First, we introduce a static strategy where we compute the optimal number of tasks before the application checkpoints at the beginning of the execution. Then, we design a dynamic strategy that decides whether to checkpoint or to continue executing at the end of each task. We instantiate this second scenario with several examples of probability distribution laws for task durations.

KEYWORDS

Fixed-length reservation, checkpoint, preemption, iterative application, linear workflow, stochastic durations.

ACM Reference Format:
Quentin Barbut, Anne Benoit, Thomas Herault, Yves Robert, and Frédéric Vivien. 2023. When to checkpoint at the end of a fixed-length reservation?. In *Proceedings of ACM Conference (FTXS'23).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Scheduling a job onto a computing platform typically involves making a series of reservations of the required resources. Long running applications, or applications whose total run-time are hard to predit, usually split their reservation in multiple smaller reservations and use checkpoint-restart [12, 23] to save intermediate steps of computation. There are multiple advantages to this approach, but the main one is that it lowers the wait-time of the application, as the job scheduler can easily place a smaller reservation. On some platforms, a maximum reservation time is imposed on applications, forcing applications that run longer than this maximum time to split their reservation and rely on a form of checkpoint-restart. These scenarios occur in large scale High Performance Computing (HPC) platforms as well as on the Cloud.

For each actual reservation, the job needs to be checkpointed before the reservation time has elapsed, otherwise the progress of the execution during the reservation will be lost.

This work focuses on an application executing for a fixed duration. The size of the application (sequential or parallel) is irrelevant; what matters is the volume of the data that needs to be saved before the end of the execution, or equivalently, the time needed to checkpoint that data. The (very natural) objective is to squeeze the most out of the reservation by executing as much work as possible before checkpointing. Obviously, in a perfect world, with a reservation of duration R and a checkpoint of duration C, one should checkpoint exactly C seconds before the end of the reservation, i.e., at time $R - C$ if the execution started at time 0.

While assuming a perfect knowledge of the value of R is quite reasonable (you know what you paid for), assuming a perfect knowledge of the value of C is more questionable. If the actual value of C exceeds the one planned by, say, a few seconds, all the work executed during the reservation will be lost. In fact, it is very likely that the value of C would vary from one execution to another; the range of variation of the possible values of C would typically depend upon the application. What is the best strategy then? If we know a worst-case value C_{max} for C, should we always use it and checkpoint at time $R - C_{max}$? Using C_{max} means taking no risk at all, but this pessimistic approach leads to wasting execution time whenever the actual value of C is significantly smaller than C_{max}.

A natural approach is to assume that a probability distribution law $\mathcal{D}_C$ for the values of C is known (instead of just an upper bound). The question becomes to determine the instant to checkpoint that maximizes the expected amount of work that will be saved before the end of the reservation. The probability distribution can be learned from traces of previous checkpoints. A main contribution of this work is to give the solution to this problem for an arbitrary distribution $\mathcal{D}_C$, and to determine the solution for a variety of widely-used distributions whose support lies in an interval $[a, b]$, where $a = C_{min}$ and $b = C_{max}$ represent the extreme values that C can take. Such distributions include Uniform($[a, b]$), the uniform law in the interval $[a, b]$, and Exponential or Normal laws truncated to $[a, b]$.

So far, we have considered that the execution of the application can be interrupted at any instant to take a checkpoint. This is a

*Also with: University of Tennessee, Knoxville, TN, USA.

very strong hypothesis too. For instance, numerical iterative applications are composed of a set of iterations that are repeated until convergence is reached: checkpoints should be taken only at the end of an iteration, because the data footprint to be saved has a much smaller volume than when the checkpoint is taken in the middle of an iteration. Another example is that of linear workflows, which are composed of a linear chain of tasks. These tasks are black boxes that operate on inputs and deliver outputs; checkpoints can only be taken at the end of a task. A main contribution of this work is to deal with such applications. To complicate matters, the duration of the tasks themselves is likely to vary from one execution to the next, just as the duration of the checkpoint. Assuming that all task durations are Independent and Identically Distributed (IID) and obey the same probability distribution law $\mathcal{D}_X$, and still using another probability distribution law $\mathcal{D}_C$ for the duration of the checkpoint, we provide the optimal strategy to maximize the expectation of the amount of work executed during the reservation. This optimal strategy comes in two flavors: either we compute the best time to checkpoint statically, at the beginning of the execution, or we dynamically decide either to checkpoint or to continue at the end of each task, accounting for the actual duration of all previously executed tasks.

We point out that this work deals with checkpointing on failure-free platforms! On HPC platforms, checkpoint/restart is the de facto standard to mitigate the impact of fail-stop errors [12]. By nature, fail-stop errors strike at random instants. Here we take checkpoints to save the work at the end of the reservation, which we can interpret as a fail-stop error that will strike at a well-known and fully deterministic instant. Another difference is that reservations are used for all kind of jobs, sequential or parallel, while checkpoint/restart is used only for very large jobs executing on very large platforms. Hence this work has a wider potential impact than large-scale HPC platforms

Altogether, the major contributions of this work are the following:

- When the (preemptible) application allows for checkpointing at any time-step: assuming that checkpoint times obey a probability distribution law $\mathcal{D}_C$, we compute the optimal time to checkpoint, in order to maximize the expected work done during the reservation.
- When the application consists of a linear chain of tasks and a checkpoint can be taken only at the end of a task: assuming IID stochastic task execution times that obey a probability distribution law $\mathcal{D}_X$, and still assuming that checkpoint times obey a probability distribution law $\mathcal{D}_C$, we compute the optimal number of tasks after which to checkpoint, in order to maximize the expected work done during the reservation. This optimal number of tasks is computed either statically at the beginning of the execution, or dynamically at the end of each task.
- For both scenarios, we provide several examples with a variety of probability distribution laws for checkpoint and task durations.

The rest of the paper is organized as follows. Section 2 reviews related work. We discuss applications that can checkpoint at any instant in Section 3 and stochastic linear workflows where checkpoints can only be taken at the end of a task in Section 4. Finally, Section 5 provides concluding remarks and directions for future work.

2 RELATED WORK

We survey related work in this section. First, we point out that most of the literature uses checkpoints to mitigate the impact of fail-stop errors that can strike during the execution of a large-scale parallel application. In such a context, the natural strategy is to checkpoint periodically, and the optimal checkpointing period is given by the Young/Daly formula [4, 26]. In our framework, checkpointing is used only to save the application data at the end of the reservation. The application may well be sequential or moderately parallel. The execution is assumed to be safe while progressing during the reservation. In other words, the only *catastrophic* event is the end of the reservation, but this one is fully known in advance. What is not known is the duration of the final checkpoint at the end of the reservation.

The first part of this work deals with a fully preemptible application executing for R seconds and where checkpoints can be taken at any instant. We assume that checkpoint time obeys a probability distribution law $\mathcal{D}_C$ and investigate what is the optimal instant to checkpoint in order to maximize the expectation of the amount of work executed during the reservation. This is a very natural and important problem because checkpointing at the end of a reservation is routinely used in many scientific fields as a way to *save state* [22, 23]. However, to the best of our knowledge, this work is the first to investigate this problem.

The second part of this work deals with linear workflows made of identical tasks that are repeated until some criterion is met. This framework corresponds to iterative methods that 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 [19], the Gauss-Seidel method [19] and the Successive Overrelaxation method (SOR) [7, 25], and non-stationary iterative methods like Krylov subspace methods, including Generalized Minimal Residual method (GMRES) [20], Biconjugate Gradient Stabilized method (BiCGSTAB) [10], Generalized Conjugate Residual method (GCR) [6], together with their ABFT (algorithm-based fault-tolerance) variants [1, 15].

The class of iterative applications goes well beyond sparse linear solvers. Uncertainty Quantification (UQ) workflows explore a parameter space in an iterative fashion [16, 18]. This class also encompasses many image and video processing software which operate a chain of computations kernels (each being a task) on a sequence of data sets (each corresponding to an iteration). Examples include image analysis [21], video processing [9], motion detection [14], signal processing [3, 11], databases [2], molecular biology [17], medical imaging [8], and various scientific data analyses, including particle physics [5], earthquake [13], weather and environmental data analyses [17].

Iterative applications are the primary motivation for the second scenario of this work: we have an unknown number of tasks, whose number depends on the convergence rate. The total execution time is unknown, which calls for a series of fixed-length reservations of duration R, where R depends upon many parameters provided

both by the user (estimating the order of magnitude of the total execution time) and the resource provider (availability and cost of each reservation). Within each reservation, the execution progresses from one iteration to the next until a checkpoint is taken in the end. if the execution starts with a recovery of length r, this amounts to working with a reservation of length $R - r$ instead of R. Each iteration is a task whose length obeys the same probability distribution law. Another probability distribution law is used for the duration of the final checkpoint. To the best of our knowledge, this work is the first to investigate this important but challenging problem.

3 CHECKPOINTING AT ANY INSTANT

In this section, we assume that a checkpoint can be taken at any instant during the reservation. Section 3.1 details the framework and provides a general formulation for the expectation $\mathbb{E}(W(X))$ of the work done when checkpointing X seconds before the end of the reservation, when assuming that checkpoint times obey a probability distribution law $\mathcal{D}_C$. Section 3.2 shows how to optimize this expectation for several widely used probability laws $\mathcal{D}_C$.

3.1 Framework

Starting the execution at time 0, we take a checkpoint at time $R-X$, where $0 \leq X \leq R$. The time to checkpoint C is a random variable that obeys a probability distribution law $\mathcal{D}_C$ with support $[a, b]$, where $0 < a < b \leq R$. In particular, we always have $a \leq C \leq b$. In fact, the lower bound a of C leads to refine the range of X as $a \leq X \leq R$: if $X < a$, there is simply not enough time left to checkpoint! We use this range $a \leq X \leq R$ throughout the paper.

How to choose X to maximize the expectation $\mathbb{E}(W(X))$ of the amount of work $W(X)$ that is saved when checkpointing at time $R - X$? The work saved by a checkpoint at time $R - X$ is

$$W(X) = \begin{cases} (R-X)\mathbb{1}_{[a,X]}(C) & \text{if } X \leq b \\ R - X & \text{if } X > b \end{cases}$$

Here $\mathbb{1}_{[a,X]}(C)$ is the indicator function whose value is 1 is $C \in [a, X]$ and 0 otherwise. Indeed, we save $R-X$ if $C \leq X$ and nothing otherwise. This confirms that choosing $X > b$ will never be optimal, but it may well be the case that the optimal is reached for $X < b$.

Let Z be a random variable with cumulative distribution function (CDF) F and probability density function (PDF) f with possibly an infinite support. The law $\mathcal{D}_C$ of C is defined as the law of Z truncated within $[a, b]$. Then we have:

$$\begin{aligned} \mathbb{P}(C \leq x) &= \mathbb{P}(Z \leq x | a \leq Z \leq b) \\ &= \frac{\mathbb{P}(Z \leq x \cap a \leq Z \leq b)}{\mathbb{P}(a \leq Z \leq b)} \\ &= \begin{cases} 0 & \text{if } x \notin [a, b] \\ \frac{\int_a^x f(t)dt}{\int_a^b f(t)dt} = \frac{F(x)-F(a)}{F(b)-F(a)} & \text{otherwise} \end{cases} \end{aligned}$$

This gives the CDF F_C of C. Rewriting it as

$$F_C(X) = \mathbb{P}(C \leq x) = \frac{\int_a^x f(t)dt}{\int_a^b f(t)dt} = \int_a^x \frac{f(t)}{\int_a^b f(u)du}dt$$

we obtain that the PDF f_C of C is $t \longmapsto \frac{f(t)}{F(b)-F(a)}$.

We can now derive the expectation of the work saved when checkpointing at time X:

$$\begin{aligned} \mathbb{E}(W(X)) &= \int_a^b W(X)\frac{f(c)}{F(b)-F(a)}dc \\ &= \begin{cases} \int_a^b (R-X)\mathbb{1}_{[0,X]}(c)\frac{f(c)}{F(b)-F(a)}dc & \\ \quad = \frac{F(X)-F(a)}{F(b)-F(a)}(R-X) & \text{if } X \leq b \\ R - X & \text{otherwise} \end{cases} \end{aligned} \tag{1}$$

In Section 3.2, we use Equation (1) to find the optimal value of X for various probability distribution laws.

3.2 Solution for several probability distribution laws

3.2.1 Uniform law. For a uniform law in $[a, b]$, there is no need for truncating, and we directly have the PDF and CDF as

$$f_C(t) = \frac{1}{b-a}$$
$$F_C(X) = \int_a^X f(t)dt = \frac{X-a}{b-a}$$

The expectation of the work saved when checkpointing at time X is:

$$\mathbb{E}(W(X)) = \begin{cases} \frac{X-a}{b-a}(R-X) & \text{if } X \leq b \\ R - X & \text{otherwise} \end{cases} \tag{2}$$

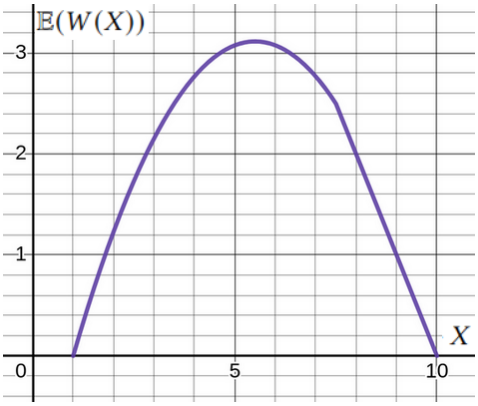
The trinomial $x \longmapsto (x-a)(R-x)$ is maximum for $x = \frac{R+a}{2}$. This is the optimal value X_{opt} of X if $\frac{R+a}{2} < b$, otherwise the maximum is obtained for some x larger than b, and then b is optimal in the interval $[a, b]$. Altogether

$$X_{opt} = \begin{cases} \frac{R+a}{2} & \text{if } R \leq 2b - a \\ b & \text{otherwise} \end{cases} = min\left(\frac{R+a}{2}, b\right)$$

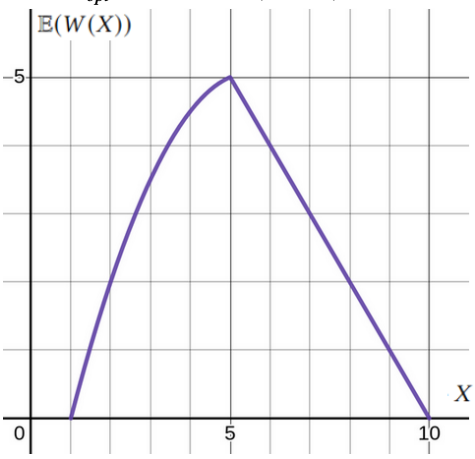
Figure 1 provides an example of each case (optimal reached before b or at b). Recall that the range of X is $[a, R]$. When there remains $X = a$ seconds before the end of the reservation, the checkpoint will fail almost surely, and the expectation of the work saved is $\mathbb{E}(W(a)) = 0$. Similarly, if we checkpoint at the very beginning of the reservation, i.e., $X = R$, no work is executed and $\mathbb{E}(W(R)) = 0$. In between, the expectation $\mathbb{E}(W(X))$ of the work done obeys Equation (2). In particular, it decreases linearly from $X = b$ to $X = R$. In Figure 1(a), the maximum of $\mathbb{E}(W(X))$ is reached for $X_{opt} = \frac{R+a}{2} = 5.5$, with $\mathbb{E}(W(X_{opt})) \approx 3.1$; the pessimistic approach would use $X = C_{max} = b$ and get $\mathbb{E}(W(b)) = 2.5$, reaching only 80% of the optimal work amount in average. On the contrary, in Figure 1(b), the pessimistic approach is optimal since $X_{opt} = b$. The main take-away is that deciding to checkpoint with $X = b$, hence preparing for the worst-case of checkpoint duration, is not always a good strategy.

3.2.2 Exponential law. Let F and f be the CDF and PDF of an Exponential distribution law of parameter $\lambda = \frac{1}{\mu}$ with $\mu \in [a, b]$. We have $f(t) = \lambda e^{-\lambda t}$ and $F(x) = 1 - e^{-\lambda x}$. The distribution law of C is this Exponential law truncated to $[a, b]$. From Section 3.1, we have

$$\mathbb{E}(W(X)) = \begin{cases} \frac{F(X)-F(a)}{F(b)-F(a)}(R-X) = \frac{e^{-\lambda a}-e^{-\lambda X}}{e^{-\lambda a}-e^{-\lambda b}}(R-X) & \text{if } X \leq b \\ R - X & \text{otherwise} \end{cases}$$



(a) Graph of $\mathbb{E}(W(X))$. The maximum is $X_{opt} = 5.5$ with $a = 1$, $b = 7.5$, $R = 10$.



(b) Graph of $\mathbb{E}(W(X))$. The maximum is $X_{opt} = b$ with $a = 1$, $b = 5$, $R = 10$.

Figure 1: Both cases for X_{opt} with a Uniform law.

Differentiating for $X \leq b$, we obtain

$$\frac{d\mathbb{E}(W(X))}{dX} = \frac{-e^{-\lambda a} + (\lambda R + 1)e^{-\lambda X} - \lambda X e^{-\lambda X}}{e^{-\lambda a} - e^{-\lambda b}}$$

Using Wolfram Alpha [24], this derivative has a unique zero for

$$X = \frac{-\mathcal{W}(e^{-\lambda a + \lambda R + 1}) + \lambda R + 1}{\lambda}$$

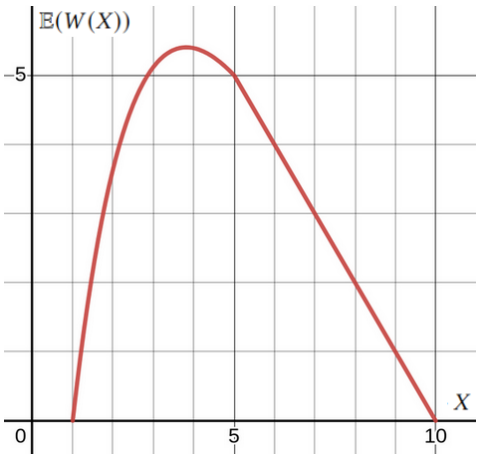
where $\mathcal{W}$ is the main branch of Lambert's $\mathcal{W}$ function (defined as $\mathcal{W}(z) = x \Leftrightarrow xe^x = z$). Differentiating again, we get

$$\frac{d^2\mathbb{E}(W(X))}{dX^2} = \frac{-\lambda(2 + \lambda R - \lambda X)e^{-\lambda X}}{e^{-\lambda a} - e^{-\lambda b}}$$

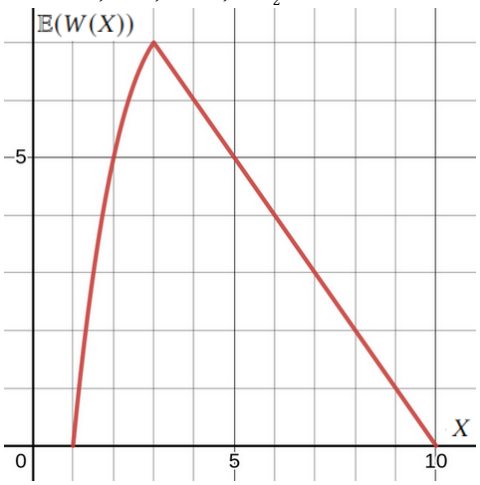
Since $X < R + \frac{2}{\lambda}$, we have $2 + \lambda R - \lambda X > 0$. Moreover, $e^{-\lambda a} - e^{-\lambda b} > 0$ (since $t \longmapsto e^{-\lambda t}$ is decreasing) and $-\lambda e^{-\lambda X} < 0$. Therefore the second derivative is strictly negative on $[a, b]$. The expectation $\mathbb{E}(W(X))$ of the work saved when checkpointing at time X is a concave function on $[a, b]$. The zero of the first derivative is thus a maximum. Altogether, the optimal value X_{opt} is given by

$$X_{opt} = \min\left(\frac{-\mathcal{W}(e^{-\lambda a + \lambda R + 1}) + \lambda R + 1}{\lambda}, b\right)$$

Figure 2 provides an example of each case (optimal reached before b or at b). Again, the main take-away is that preparing for the worst-case of checkpoint duration and choosing $X = b$ is not always a good strategy. Contrarily to the Uniform law, the Exponential law requires to compute a complicated value for X_{opt}, but this can be done easily with available tools like [24].



(a) Graph of $\mathbb{E}(W(X))$. The maximum is $X_{opt} \approx 3.9$ with $a = 1$, $b = 5$, $R = 10$, $\lambda = \frac{1}{2}$.



(b) Graph of $\mathbb{E}(W(X))$. The maximum is $X_{opt} = b$ with $a = 1$, $b = 3$, $R = 10$, $\lambda = \frac{1}{2}$.

Figure 2: Both cases for X_{opt} with an Exponential law.

3.2.3 Normal law. Let Φ and φ the CDF and PDF of the standard Normal law: $\varphi(t) = \frac{e^{-\frac{t^2}{2}}}{\sqrt{2\pi}}$ and $\Phi(x) = \int_{-\infty}^{x} \varphi(t)dt$. The Normal law with mean μ and standard deviation σ has CDF $\Phi(\frac{x-\mu}{\sigma})$. We assume that C obeys the Normal law with mean $\mu \in [a, b]$ and standard

deviation $\sigma > 0$ truncated to $[a, b]$. From Section 3.1, we have

$$\mathbb{E}(W(X)) = \begin{cases} \frac{\Phi(\frac{X-\mu}{\sigma}) - \Phi(\frac{a-\mu}{\sigma})}{\Phi(\frac{b-\mu}{\sigma}) - \Phi(\frac{a-\mu}{\sigma})}(R - X) & \text{if } X \leq b \\ R - X & \text{otherwise} \end{cases}$$

We get rid of the positive constant $K = \frac{1}{\Phi(\frac{b-\mu}{\sigma}) - \Phi(\frac{a-\mu}{\sigma})}$ by letting $g(X) = K\mathbb{E}(W(X))$. Differentiating for $X < b$, we get

$$g'(X) = \frac{d\mathbb{E}(W(X))}{dX} = \frac{1}{\sigma}\varphi\left(\frac{X-\mu}{\sigma}\right)(R-X) - \left[\Phi\left(\frac{X-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)\right]$$

We have $g'(a) = \frac{1}{\sigma}\varphi(\frac{a-\mu}{\sigma})(R-a) > 0$ since $a < R$. We also have $g'(R) = -[\Phi(\frac{R-\mu}{\sigma}) - \Phi(\frac{a-\mu}{\sigma})]$. We see that $g'(R) < 0$ since Φ is an increasing function and $a < R$. Since g' is continuous, the intermediate value theorem shows that there exists $c \in [a, R]$ such that $g'(c) = 0$. Differentiating again:

$$g''(X) = \frac{d^2\mathbb{E}(W(X))}{dX^2} = -\frac{\varphi(\frac{X-\mu}{\sigma})}{\sigma}\left[2 + \left(\frac{X-\mu}{\sigma}\right)(R-X)\right]$$

g'' has two zeros:

$$X_1 = \frac{R+\mu-\sqrt{(R+\mu)^2+8\sigma^2-4\mu R\sigma}}{2}$$
$$X_2 = \frac{R+\mu+\sqrt{(R+\mu)^2+8\sigma^2-4\mu R\sigma}}{2}$$

We see that $X_1 < \mu < R < X_2$. Furthermore, we have

$$\begin{cases} g''(X) > 0 \text{ si } X < X_1 \text{ ou } X > X_2 \\ g''(X) < 0 \text{ si } X_1 < X < X_2 \end{cases}$$

g is thus a concave function on $[X_1, X_2]$ and a convex function elsewhere. There are two possible cases:

(1) either $a \geq X_1$, and then g is concave on $[a, R]$; hence the zero c of g' is a maximum of g.
(2) or $X_1 > a$, and then g' is increasing $[a, X_1]$; since $g'(a) > 0$, g' is positive on $[a, X_1]$. Hence $c \in]X_1, R[$ and g is concave in a neighborhood of c, and c is again a maximum of g.

Altogether, we have shown the existence of a maximum $c \in]a, R]$ for the expectation $\mathbb{E}(W(X))$, namely

$$X_{opt} = \min(c, b)$$

We do not have an explicit formula for X_{opt} but we can evaluate it numerically. Figure 3 provides an example of each case (optimal reached before b or at b). The main take-away for the Normal law is the same as for the Exponential law.

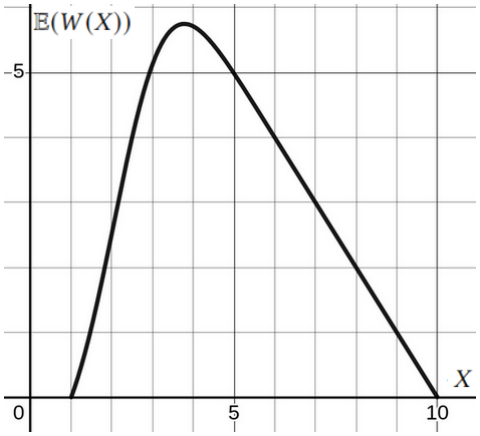
3.2.4 LogNormal law. Let F and f be the CDF and PDF of a Log-Normal law of parameters μ and σ: we have

$$f(t) = \frac{1}{t\sigma\sqrt{2\pi}} exp\left(-\frac{(ln(t)-\mu)^2}{2\sigma^2}\right) = \frac{1}{t\sigma}\varphi\left(\frac{ln(t)-\mu}{\sigma}\right)$$

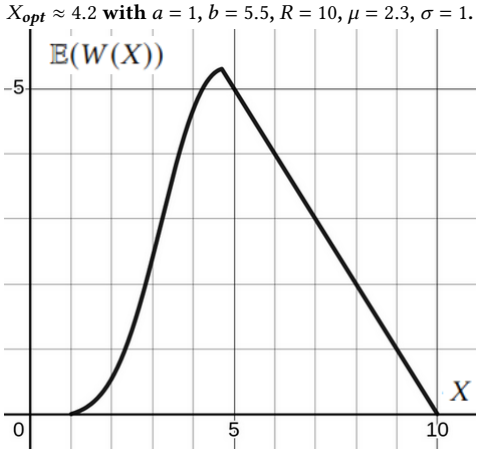
and $F(x) = \int_0^x f(t)dt = \Phi\left(\frac{ln(x)-\mu}{\sigma}\right)$. Recall that the mean μ^* and standard deviation σ^* of this law are such that

$$\mu^* = \exp(\mu + \frac{\sigma^2}{2}) \text{ and } \sigma^* = \sqrt{\left(\exp(\sigma^2) - 1\right)\exp(2\mu + \sigma^2)}$$

We assume that C obeys the LogNormal law with parameters μ and σ truncated to $[a, b]$, and we choose these parameters μ and σ



(a) Graph of $\mathbb{E}(W(X))$. The maximum is $X_{opt} \approx 4.2$ with $a = 1$, $b = 5.5$, $R = 10$, $\mu = 2.3$, $\sigma = 1$.



(b) Graph of $\mathbb{E}(W(X))$. The maximum is $X_{opt} = b$ with $a = 1$, $b = 4.7$, $R = 10$, $\mu = 3.5$, $\sigma = 1$.

Figure 3: Both cases for X_{opt} with a Normal law.

so that $\mu^* \in [a, b]$. From Section 3.1, we have

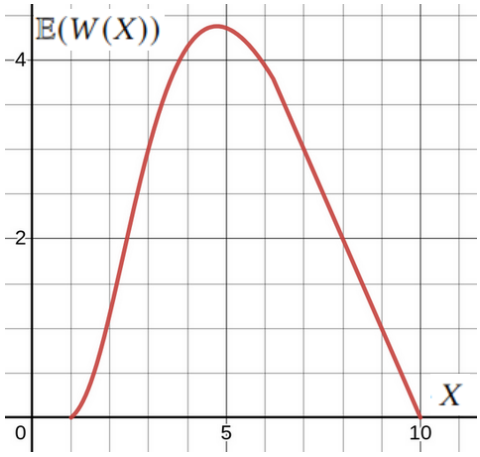
$$\mathbb{E}(W(X)) = \begin{cases} \frac{\Phi(\frac{ln(X)-\mu}{\sigma}) - \Phi(\frac{ln(a)-\mu}{\sigma})}{\Phi(\frac{ln(b)-\mu}{\sigma}) - \Phi(\frac{ln(a)-\mu}{\sigma})}(R - X) & \text{if } X \leq b \\ R - X & \text{otherwise} \end{cases}$$

The determination of the maximum of the expectation of the work done is similar to what we have done for a truncated Normal law, therefore we do not detail the derivations. Just as for the truncated Normal law, the maximum can be obtained either for $X_{opt} < b$ or for $X_{opt} = b$. Figure 4 provides an example of each case (optimal reached before b or at b). The main take-away for the Log Normal law is the same as for the Exponential and Normal laws.

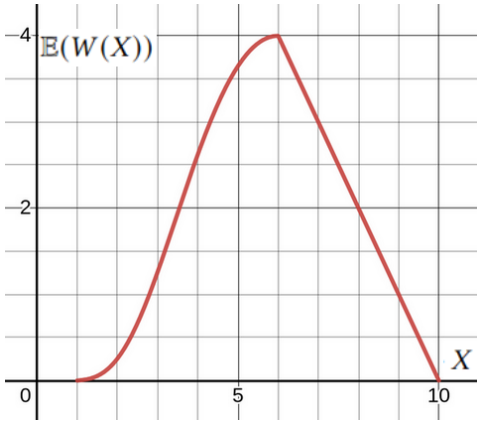
4 STOCHASTIC LINEAR WORKFLOWS

4.1 Framework

This section addresses a much more challenging problem than the one of Section 3. Now the application consists of a linear chain of tasks. Checkpoints cannot be taken at any instant during the



(a) Graph of $\mathbb{E}(W(X))$. The maximum is $X_{opt} \approx 4.6$ with $a = 1$, $b = 6.2$, $R = 10$, $\mu = 1.25$, $\sigma = 0.5$.



(b) Graph of $\mathbb{E}(W(X))$. The maximum is $X_{opt} = b$ with $a = 1$, $b = 6$, $R = 10$, $\mu = 1.75$, $\sigma = 0.5$.

Figure 4: Both cases for X_{opt} with a LogNormal law.

execution but instead must be taken at the end of a task. The objective is to execute as many tasks as possible and to checkpoint successfully before the end of the reservation. We further assume that task execution times are not fully deterministic and can vary from one execution to the next. In the most general setting, we would have a chain

$$T_1 \to T_2 \to \cdots \to T_n \to T_{n+1} \to \dots$$

where each task T_i is characterized by two probability distributions:

- a first probability distribution $\mathcal{D}_X^{(i)}$ to model the execution time of the task
- a second independent probability distribution $\mathcal{D}_C^{(i)}$ to model the time to checkpoint at the end of the task

All the $\mathcal{D}_X^{(i)}$ and $\mathcal{D}_C^{(i)}$ distributions are supposed to be independent.

We note that if task execution times are deterministic instead of stochastic (in other words, if $\mathcal{D}_X^{(i)}$ is constant for all i), the problem can be solved using the same approach as in Section 3. Obviously, it is much more realistic to assume that task execution times can change, even moderately, from one execution to another; but this assumption dramatically complicates the problem.

In this work, we restrict to a simpler yet challenging instance of the problem: we assume that the probability distributions are the same for all tasks. More precisely, we assume that the $\mathcal{D}_X^{(i)}$ are independent and identically distributed (IID) and obey the same distribution $\mathcal{D}_X$; similarly, the $\mathcal{D}_C^{(i)}$ are independent and identically distributed (IID) and obey the same distribution $\mathcal{D}_C$. As mentioned in Section 2, this problem instance with IID stochastic tasks perfectly models the behavior of large-scale numerical iterative solvers for sparse linear systems of equations.

Because of the difficulty of the problem, we make further technical assumptions. The key argument in the solution of the static strategy described in Section 4.1 is to restrict to distributions $\mathcal{D}_X$ such that the sum of n IID random variables X_i obeying $\mathcal{D}_X$ will obey a well-known probability distribution, typically the same type as $\mathcal{D}_X$ but with scaled parameters. Recall that each X_i represents the execution time of task T_i, so that $S_n = \sum_{i=1}^{n} X_i$ represents the execution time of the first n tasks: this is why we need $X^{(n)}$ to obey some well-known probability distribution. We investigate three cases below that match this restriction: when $\mathcal{D}_X$ is a Normal law, a Gamma law, or a Poisson law (this last one requires discretization of time). Finally, for simplicity of the derivations, we assume that each distribution $\mathcal{D}_C^{(i)}$ obeys the same Normal law $\mathcal{D}_C \sim \mathcal{N}(\mu_C, \sigma_C^2)$ truncated to positive values (support $[0, \infty)$). It is easy to extend the approach to different distributions $\mathcal{D}_C^{(i)}$ of arbitrary types: simply compute the expectation of the work done after n iterations for each value of n, using the approach below, and select the best value.

To summarize, task execution times are IID distributions $\mathcal{D}_X$ with positive support $[0, +\infty[$, checkpoint times are IID truncated Normal distributions $\mathcal{D}_C \sim \mathcal{N}_{[0,+\infty[}(\mu_C, \sigma_C^2)$, and all these distributions are independent. We investigate two strategies. First we use a static approach: at the beginning of the execution, we compute the value n_{opt} of the number of iterations that should be executed before taking a checkpoint in order to maximize the expectation of the work done. Then, we provide a dynamic strategy that accounts for the work actually done so far and decides at the end of each iteration whether it is better (in expectation) to checkpoint now or to perform another iteration and checkpoint only then.

4.2 Static strategy

The static strategy is applied before the beginning of the execution, and takes a checkpoint at the end of the same iteration number n_{opt}. The goal is to determine the value of n_{opt} which maximizes the expectation $\mathbb{E}(n)$ of the work done when checkpointing after n iterations. Assuming that $\mathcal{D}_X$ has positive support $[0, \infty)$, $\mathbb{E}(n)$ can be expressed as follows:

$$\mathbb{E}(n) = \int_0^R x \left(\int_0^{R-x} f_C(c)dc \right) f_{S_n}(x)dx \tag{3}$$

In Equation (3), f_C is the PDF of $\mathcal{D}_C$ and f_{S_n} is the PDF of $S_n = \sum_{i=1}^{n} X_i$. As stated before, this expression is useful only when each random variable S_n obeys some well-known distribution.

4.2.1 Normal law. In this section, we assume that task execution times obey a Normal law: $X_i \sim \mathcal{N}(\mu, \sigma^2)$. Then S_n is a Normal law too: $S_n \sim \mathcal{N}(n\mu, n\sigma^2)$. A non-truncated Normal law to model task execution times is meaningful only if its mean is a large positive number and its standard deviation relatively small, so that the probability for X_i to be negative remains very low, thereby ensuring the coherence of the model. But for correctness, we need to update Equation (3) to account for possible negative values. We derive:

$$\mathbb{E}(n) = \int_{-\infty}^{R} x \left(\int_{0}^{R-x} f_C(c)dc \right) f_{S_n}(x)dx$$
$$= \int_{-\infty}^{R} x \left[\frac{\Phi\left(\frac{R-x-\mu_C}{\sigma_C}\right) - \Phi\left(-\frac{\mu_C}{\sigma_C}\right)}{1-\Phi\left(-\frac{\mu_C}{\sigma_C}\right)} \right] \varphi\left(\frac{x-n\mu}{\sqrt{n}\sigma}\right) \frac{1}{\sqrt{n}\sigma} dx$$

We replace n by a real variable $y \in]0, +\infty[$ to get the continuous function

$$f(y) = \int_{-\infty}^{R} x \left[\frac{\Phi\left(\frac{R-x-\mu_C}{\sigma_C}\right) - \Phi\left(-\frac{\mu_C}{\sigma_C}\right)}{1-\Phi\left(-\frac{\mu_C}{\sigma_C}\right)} \right] \varphi\left(\frac{x-y\mu}{\sqrt{y}\sigma}\right) \frac{1}{\sqrt{y}\sigma} dx$$

If f has a maximum y_{opt}, then the optimal value n_{opt} will be either $n_{opt} = \lfloor y_{opt} \rfloor$ or $n_{opt} = \lceil y_{opt} \rceil$, whichever gives the larger value for f.

We provide a numerical example in Figure 5. In this example, f has a maximum $y_{opt} \approx 7.4$. We have $f(7) \approx 20.9$ and $f(8) \approx 17.6$, hence $n_{opt} = 7$.

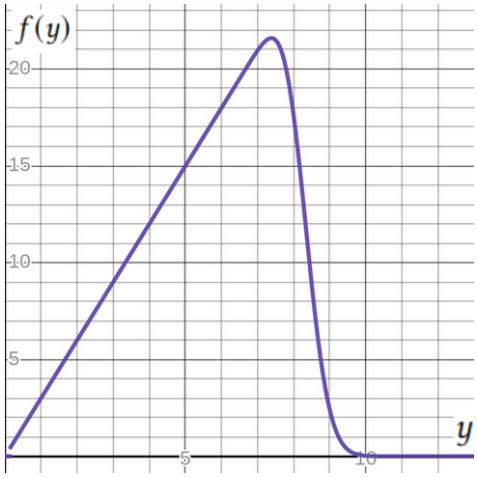


Figure 5: Graph of function f (continuous function for $\mathbb{E}(n)$) with $\mu = 3$, $\sigma = 0.5$, $\mu_C = 5$, $\sigma_C = 0.4$ and R=30.

4.2.2 Gamma law. In this section, we assume that task execution times obey a Gamma law: $X_i \sim \text{Gamma}(k, \theta)$. Recall that its support is $[0, \infty($ and its PDF is $f(x, k, \theta) = \frac{x^{k-1} e^{\frac{x}{\theta}}}{\Gamma(k)\theta^k}$ where Γ denotes the Euler function $z \longmapsto \int_0^{+\infty} t^{z-1} e^{-t} dt$. Because the support of this distribution only has positive values, we can use Equation (3) as such. Indeed, the sum S_n of n independent $X_i \sim \text{Gamma}(k, \theta)$ is $\S_n \sim \text{Gamma}(nk, \theta)$. We derive:

$$\mathbb{E}(n) = \int_{0}^{R} x \left(\int_{0}^{R-x} f_C(c)dc \right) f_{S_n}(x)dx$$
$$= \int_{0}^{R} x \left[\frac{\Phi\left(\frac{R-x-\mu_C}{\sigma_C}\right) - \Phi\left(-\frac{\mu_C}{\sigma_C}\right)}{1-\Phi\left(-\frac{\mu_C}{\sigma_C}\right)} \right] \frac{x^{nk-1} e^{-\frac{x}{\theta}}}{\Gamma(nk)\theta^{nk}} dx$$

We replace n by a real variable $y \in]0, +\infty[$ to get the continuous function

$$g(y) = \int_{0}^{R} x \left[\frac{\Phi\left(\frac{R-x-\mu_C}{\sigma_C}\right) - \Phi\left(-\frac{\mu_C}{\sigma_C}\right)}{1-\Phi\left(-\frac{\mu_C}{\sigma_C}\right)} \right] \frac{x^{yk-1} e^{-\frac{x}{\theta}}}{\Gamma(yk)\theta^{yk}} dx$$

If g has a maximum y_{opt}, then the optimal value n_{opt} will be either $n_{opt} = \lfloor y_{opt} \rfloor$ or $n_{opt} = \lceil y_{opt} \rceil$, whichever gives the larger value for g.

We provide a numerical example in Figure 6. In this example, g has a maximum $y_{opt} \approx 11.8$. We have $g(11) \approx 4.77$ and $g(12) \approx 4.82$, hence $n_{opt} = 12$.

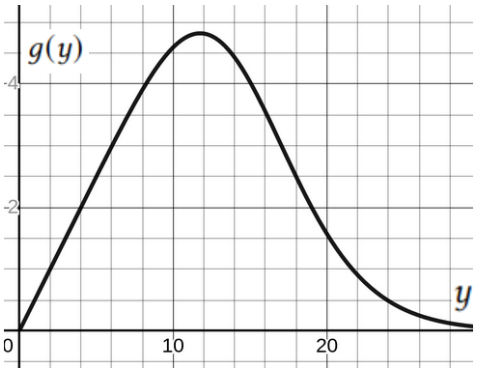


Figure 6: Graph of function g (continuous function for $\mathbb{E}(n)$) with k=1, $\theta = 0.5$, $\mu_C = 2$, $\sigma_C = 0.4$, R=10

4.2.3 Poisson law. In this section, we consider a Poisson law. As $\text{Poisson}(\lambda)$ has for support the set $\mathbb{N}$ of nonnegative integers, we assume that task execution times are expressed in some discrete unit (e.g., seconds) and take only integer values. We assume that R and the mean of the X_i random variables are large in front of the time unit. We also assume w.l.o.g. that R is an integer. Task execution times obey a Poisson law: $X_i \sim \text{Poisson}(\lambda)$. Recall that $\text{Poisson}(\lambda)$ has for PDF $f(k) = e^{-\lambda} \frac{\lambda^k}{k!}$. The sum S_n of n independent $X_i \sim \text{Poisson}(\lambda)$ is $\S_n \sim \text{Poisson}(n\lambda)$. We derive:

$$\mathbb{E}(n) = \sum_{j=0}^{R} j \left[\frac{\Phi\left(\frac{R-j-\mu_C}{\sigma_C}\right) - \Phi\left(-\frac{\mu_C}{\sigma_C}\right)}{1-\Phi\left(-\frac{\mu_C}{\sigma_C}\right)} \right] e^{-n\lambda} \frac{(n\lambda)^j}{j!}$$

We replace n by a real variable $y \in]0, +\infty[$ to get the continuous function

$$h(y) = \sum_{j=0}^{\lfloor R \rfloor} j \left[\frac{\Phi\left(\frac{R-j-\mu_C}{\sigma_C}\right) - \Phi\left(-\frac{\mu_C}{\sigma_C}\right)}{1-\Phi\left(-\frac{\mu_C}{\sigma_C}\right)} \right] e^{-y\lambda} \frac{(y\lambda)^j}{j!}$$

If h has a maximum y_{opt}, then the optimal value n_{opt} will be either $n_{opt} = \lfloor y_{opt} \rfloor$ or $n_{opt} = \lceil y_{opt} \rceil$, whichever gives the larger value for h.

We provide a numerical example in Figure 7. In this example, h has a maximum $y_{opt} \approx 5.98$. We have $h(5) \approx 14.6$ and $h(6) \approx 15.8$, hence $n_{opt} = 6$.

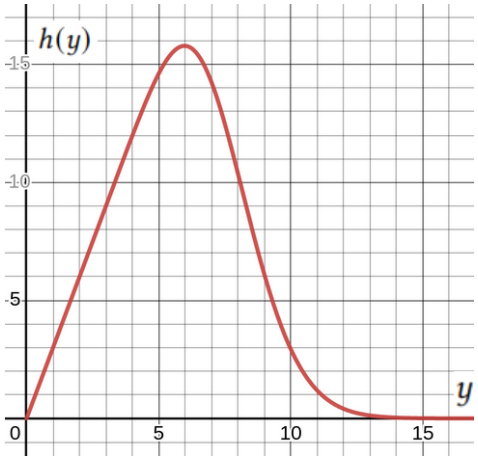


Figure 7: Graph of function h (continuous function for $\mathbb{E}(n)$) with $\lambda = 3$, $\mu_C = 5$, $\sigma_C = 0.4$, $R = 29$

4.3 Dynamic strategy

The static strategy does not account for the actual duration of the tasks during the beginning of the execution and always takes a checkpoint after the same (optimal) number of iterations. This approach is best suited to the scenario where the random variables X_i follow a distribution $\mathcal{D}_X$ with a small standard deviation. On the contrary, if $\mathcal{D}_X$ has a large standard deviation, there is a risk to checkpoint much too early or much too late, depending upon what values the X_i have effectively been taking. In this section, we introduce a dynamic strategy: given the values of previous X_I's, we decide at the end of each task whether it is better to checkpoint now or to continue with (at least) one additional task. To this purpose, at the end of each task, we compare the expectation $\mathbb{E}(W_C)$ of the work done if we checkpoint now, and the expectation $\mathbb{E}(W_{+1})$ of the work done if we execute one more task before checkpointing. If $\mathbb{E}(W_C) \geq \mathbb{E}(W_{+1})$ we stop the execution and checkpoint; otherwise, we execute one more task and re-apply the algorithm at the end of that new task.

For $n \in \mathbb{N}$, let W_n be the work done after the first n tasks. We have:

- If we checkpoint:

$$\mathbb{E}(W_C) = W_n \mathbb{P}(C \leq R - W(n))$$

- If we continue execution:

$$\mathbb{E}(W_{+1}) = \int_0^{R-W_n} (x + W_n) \left(\int_0^{R-W_n-x} f_C(c)dc \right) f_{X_{n+1}}(x)dx$$

where $f_{X_{n+1}}$ is the PDF of the random variable X_{n+1}. Since $C \sim \mathcal{N}_{[0,+\infty[}(\mu_C, \sigma_C^2)$, we derive:

$$\mathbb{E}(W_C) = W_n \mathbb{P}(C \leq R - W_n) = W_n \left[\frac{\Phi\left(\frac{R-W_n-\mu_C}{\sigma_C}\right) - \Phi\left(-\frac{\mu_C}{\sigma_C}\right)}{1 - \Phi\left(-\frac{\mu_C}{\sigma_C}\right)} \right]$$

and

$$\begin{aligned}\mathbb{E}(W_{+1}) &= \int_0^{R-W_n} (x + W_n) \left(\int_0^{R-W_n-x} f_C(c)dc \right) f_{X_{n+1}}(x)dx \\ &= \int_0^{R-W_n} (x + W_n) \left[\frac{\Phi\left(\frac{R-W_n-x-\mu_C}{\sigma_C}\right) - \Phi\left(-\frac{\mu_C}{\sigma_C}\right)}{1 - \Phi\left(-\frac{\mu_C}{\sigma_C}\right)} \right] f_{X_{n+1}}(x)dx\end{aligned}$$

Technically, the dynamic strategy provides more flexibility than the static one. because we know the actual value W_n of the work executed after n tasks, we no longer need that $S_n = \sum_{i=1}^{n} X_i$ obeys some well-known probability distribution. In what follows, we instantiate the problem with $\mathcal{D}_X$ being a truncated Normal law, a Gamma law or a Poisson law.

4.3.1 Truncated Normal law. We assume here that $X_i \sim \mathcal{N}_{[0,+\infty[}(\mu, \sigma^2)$ for all i: $\mathcal{D}_X$ is a Normal law truncated to $0, +\infty$. We derive that

$$\mathbb{E}(W_{+1}) = \int_0^{R-W_n} (x + W_n) \left[\frac{\Phi\left(\frac{R-W_n-x-\mu_C}{\sigma_C}\right) - \Phi\left(-\frac{\mu_C}{\sigma_C}\right)}{1 - \Phi\left(-\frac{\mu_C}{\sigma_C}\right)} \right] \frac{\varphi\left(\frac{x-\mu}{\sigma}\right)}{\sigma(1 - \Phi(-\frac{\mu}{\sigma}))} dx$$

We can now directly compare $\mathbb{E}(W_C)$ and $\mathbb{E}(W_{+1})$. We provide a numerical example in Figure 8. In this example, the two graphs intersect at $W_{int} \approx 20.3$. When $W_n > W_{int}$, it is better to checkpoint right now than executing another task, while it is the opposite for $W_n < W_{int}$.

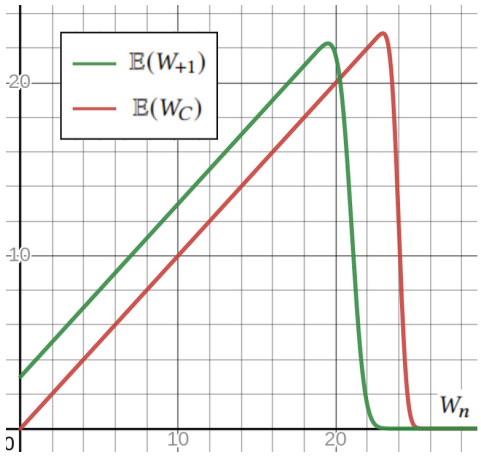


Figure 8: Graph of $\mathbb{E}(W_{+1})$ (in green) and of $\mathbb{E}(W_C)$ (in red) as a function of W_n with $\mu = 3$, $\sigma = 0.5$, $\mu_C = 5$, $\sigma_C = 0.4$, $R = 29$.

4.3.2 Gamma law. We assume here that $X_i \sim \text{GAMMA}(k, \theta)$:$\mathcal{D}_X$ is a Gamma law. We derive that

$$\mathbb{E}(W_{+1}) = \int_0^{R-W_n} (x+W_n) \left[\frac{\Phi\left(\frac{R-W_n-x-\mu_C}{\sigma_C}\right) - \Phi\left(-\frac{\mu_C}{\sigma_C}\right)}{1-\Phi\left(-\frac{\mu_C}{\sigma_C}\right)} \right] \frac{x^{k-1}e^{-\frac{x}{\theta}}}{\Gamma(k)\theta^k} dx$$

We can now directly compare $\mathbb{E}(W_C)$ and $\mathbb{E}(W_{+1})$. We provide a numerical example in Figure 9. In this example, the two graphs intersect at $W_{int} \approx 6.4$. When $W_n > W_{int}$, it is better to checkpoint right now than executing another task, while it is the opposite for $W_n < W_{int}$.

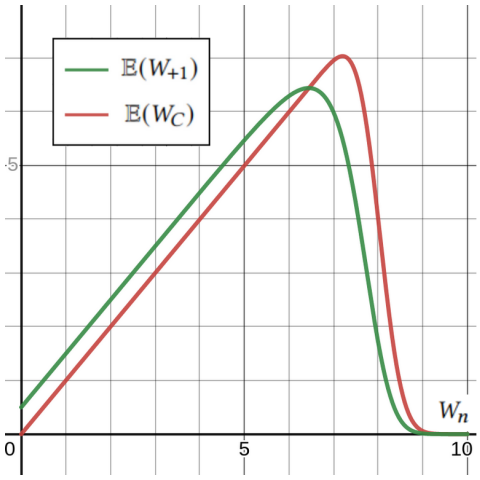


Figure 9: Graph of $\mathbb{E}(W_{+1})$ (in green) and of $\mathbb{E}(W_C)$ (in red) as a function of W_n with $k = 1$, $\sigma = 0.5$, $\theta = 0.5$, $\mu_C = 2$, $\sigma_C = 0.4$, $R = 10$.

4.3.3 Poisson law. In this section, similarly to Section 4.2.3, we consider that task execution times are expressed in some discrete unit (e.g., seconds) and take only integer values. We assume that R and the mean of the X_i random variables are large in front of the time unit. We also assume w.l.o.g. that R is an integer. Task execution times obey a Poisson law: $X_i \sim \text{POISSON}(\lambda)$. We derive:

$$\mathbb{E}(W_{+1}) = \sum_{j=0}^{R-W_n} (j + W_n) \left[\frac{\Phi\left(\frac{R-W_n-j-\mu_C}{\sigma_C}\right) - \Phi\left(-\frac{\mu_C}{\sigma_C}\right)}{1-\Phi\left(-\frac{\mu_C}{\sigma_C}\right)} \right] e^{-\lambda} \frac{\lambda^j}{j!}$$

We can now directly compare $\mathbb{E}(W_C)$ and $\mathbb{E}(W_{+1})$. We provide a numerical example in Figure 10. In this example, the two graphs intersect at $W_{int} \approx 18.9$. When $W_n > W_{int}$, it is better to checkpoint right now than executing another task, while it is the opposite for $W_n < W_{int}$.

4.4 And after the checkpoint?

We conclude this section with a short discussion about using the time left in the reservation, if any, after a checkpoint has been successfully taken. Should we attempt to execute one or several new tasks and take a new checkpoint after these new tasks? or should we drop the reservation?

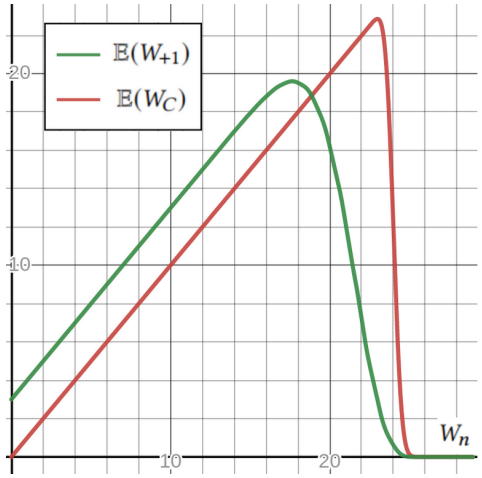


Figure 10: Graph of $\mathbb{E}(W_{+1})$ (in green) and of $\mathbb{E}(W_C)$ (in red) as a function of W_n with $\lambda = 3$, $\mu_C = 5$, $\sigma_C = 0.4$, $R = 29$.

This question can be raised when there is enough time left in the reservation after a successful checkpoint. Of course there must remain at least $a = C_{min}$ seconds, the minimum time to checkpoint. Such a scenario is indeed possible; it is more likely with the static approach which determines when to checkpoint at the beginning of the execution, hence which can overestimate actual task execution times; but it can also happen with the dynamic strategy.

If we decide to continue the execution, we can always re-use both approaches, either static or dynamic, for the time left in the reservation. However, some HPC or cloud systems charge by time actually spent rather by time reserved. In that case, it may be worth to drop the reservation and save money on our account. Obviously, the decision involves many parameters, including the urgency of getting application results and the budget of the user!

5 CONCLUSION

This work has dealt with the problem of maximizing the expectation of the work that can be done during a fixed-length reservation. The key question is when to take a checkpoint at the end of the reservation. We have started with applications where a checkpoint can be taken at any time. For such applications, we have provided the optimal solution when checkpoint time can be modeled as a random variable obeying a probability distribution law $\mathcal{D}_C$ with bounded support $[a, b]$. An important result was to assess the gain that can be achieved over the pessimistic (but risk-free) approach, which assumes the highest value $C_{max} = b$ for the checkpoint duration C, using a variety of well-known probability distribution laws $\mathcal{D}_C$.

Then, we have focused on the more involved problem where the application is a linear workflow consisting of a chain of tasks with IID stochastic execution times, and a checkpoint can be taken only at the end of a task. We have introduced a static strategy where we compute the optimal number of tasks before the checkpoint at the beginning of the execution. We have also designed a dynamic strategy, which decides whether to checkpoint or to continue execution at the end of each task. We have instantiated this second

scenario with several examples of probability distribution laws for task durations. Obviously, the dynamic strategy is to be preferred whenever its use is possible, because it accounts for the actual execution times of all tasks that have been executed so far. But not all applications can be modified on the fly to insert a checkpoint, and the static strategy has a wider potential of applicability.

However, the static strategy requires all task execution times to be IID, while the dynamic strategy does not have this restriction. In fact, it would be easy to extend the dynamic strategy to deal with the general instance of the problem, as described in Section 4.1: in the general instance, each task T_i is characterized by two probability distributions, $\mathcal{D}_X^{(i)}$ to model the execution time of the task, and $\mathcal{D}_C^{(i)}$ to model the time to checkpoint at the end of the task. The only requirement is that all the $\mathcal{D}_X^{(i)}$ and $\mathcal{D}_C^{(i)}$ distributions are independent. However, extending the static strategy to find the optimal solution for the general case seems out or reach. Future work will be devoted to the design of efficient heuristics to solve this challenging problem.

This work has laid the foundations for the design of checkpoint strategies within a fixed-length reservation. Further work is needed to experimentally assess the gain provided by such strategies for real-life scientific applications. We expect this gain to be much higher for stochastic linear workflows than for fully preemptible applications: indeed, in the former case (workflows), the pessimistic, risk free, approach needs to account for (and add-up) two worst-case durations, namely that of a task and that of a checkpoint; while in the latter case (preemptible applications), only the maximum duration of a checkpoint is required. An experimental campaign, either via simulations using traces or through actual application runs, is needed to quantify the effective gain for both application types.

Finally, as mentioned in the introduction, this work is not related to checkpointing on failure-prone platforms. Dealing with the occurrence of fail-stop errors within fixed-size reservations would be an interesting direction for future work.

REFERENCES

[1] E. Agullo, L. Giraud, A. Guermouche, J. Roman, and M. Zounon. Numerical recovery strategies for parallel resilient krylov linear solvers. *Numerical Linear Algebra with Applications*, 23(5):888–905, 2016.

[2] C. Chekuri, W. Hasan, and R. Motwani. Scheduling problems in parallel query optimization. In *PODS'1995, the 14th ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 255–265, New York, NY, USA, 1995. ACM Press.

[3] A. Choudhary, W.-k. Liao, D. Weiner, P. Varshney, R. Linderman, M. Linderman, and R. Brown. Design, implementation and evaluation of parallel pipelined STAP on parallel computers. *IEEE Transactions on Aerospace and Electronic Systems*, 36(2):655 – 662, Apr. 2000.

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

[5] E. Deelman, J. Blythe, Y. Gil, and C. Kesselman. Workflow management in griphyn. In *Grid Resource Management*. Springer, 2003.

[6] S. C. Eisenstat, H. C. Elman, and M. H. Schultz. Variational iterative methods for nonsymmetric systems of linear equations. *SIAM Journal on Numerical Analysis*, 20(2):345–357, 1983.

[7] S. P. Frankel. Convergence rates of iterative treatments of partial differential equations. *Mathematical Tables and Other Aids to Computation*, 4(30):65–75, 1950.

[8] F. Guirado, A. Ripoll, C. Roig, A. Hernandez, and E. Luque. Exploiting throughput for pipeline execution in streaming image processing applications. In *Euro-Par 2006, Parallel Processing*, LNCS 4128, pages 1095–1105. Springer, 2006.

[9] F. Guirado, A. Ripoll, C. Roig, and E. Luque. Optimizing latency under throughput requirements for streaming applications on cluster execution. In *Cluster Computing, 2005. IEEE International*, pages 1–10, Sept. 2005.

[10] M. H. Gutknecht. Variants of bicgstab for matrices with complex spectrum. *SIAM journal on scientific computing*, 14(5):1020–1033, 1993.

[11] T. D. R. Hartley, A. R. Fasih, C. A. Berdanier, F. Ozguner, and Ü. V. Çatalyürek. Investigating the Use of GPU-Accelerated Nodes for SAR Image Formation. In *Proceedings of the IEEE International Conference on Cluster Computing, Workshop on Parallel Programming on Accelerator Clusters (PPAC)*, 2009.

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

[13] J. Kim, Y. Gil, and M. Spraragen. A knowledge-based approach to interactive workflow composition. In *14th International Conference on Automatic Planning and Scheduling (ICAPS 04)*, 2004.

[14] K. Knobe, J. M. Rehg, A. Chauhan, R. S. Nikhil, and U. Ramachandran. Scheduling constrained dynamic applications on clusters. In *Supercomputing'1999, the 1999 ACM/IEEE conference on Supercomputing*, page 46, New York, NY, USA, 1999. ACM.

[15] J. Langou, Z. Chen, G. Bosilca, and J. Dongarra. Recovery patterns for iterative methods in a parallel unstable environment. *SIAM Journal on Scientific Computing*, 30(1):102–116, 2008.

[16] W. L. Oberkampf and C. J. Roy. *Verification and validation in scientific computing*. Cambridge University Press, 2010.

[17] A. Rowe, D. Kalaitzopoulos, M. Osmond, M. Ghanem, and Y. Guo. The discovery net system for high throughput bioinformatics. *Bioinformatics*, 19(Suppl 1):i225–31, 2003.

[18] C. J. Roy and W. L. Oberkampf. A comprehensive framework for verification, validation, and uncertainty quantification in scientific computing. *Computer methods in applied mechanics and engineering*, 200(25-28):2131–2144, 2011.

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

[20] Y. Saad and M. H. Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.

[21] O. Sertel, J. Kong, H. Shimada, Ü. V. Çatalyürek, J. H. Saltz, and M. N. Gurcan. Computer-aided prognosis of neuroblastoma on whole-slide images: Classification of stromal development. *Pattern Recognition*, 42(6):1093–1103, 2009.

[22] D. Sharma. Application checkpointing, 2022. https://hevodata.com/learn/application-checkpointing/.

[23] Wikipedia. Application checkpointing, 2023. https://en.wikipedia.org/wiki/Application_checkpointing.

[24] Wolfram Alpha. Mathematics, 2023. https://www.wolframalpha.com.

[25] D. Young. Iterative methods for solving partial difference equations of elliptic type. *Transactions of the American Mathematical Society*, 76(1):92–111, 1954.

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