Reservation and Checkpointing Strategies for Stochastic Jobs (Extended Version)

Ana Gainaru, Brice Goglin, Valentin Honoré, Guillaume Pallez, Padma Raghavan, Yves Robert, Hongyang Sun

Reservation and Checkpointing Strategies for Stochastic Jobs (Extended Version)

Ana Gainaru, Brice Goglin, Valentin Honoré, Guillaume Pallez, Padma Raghavan, Yves Robert, Hongyang Sun

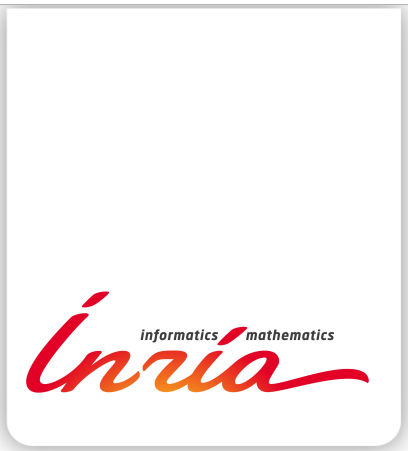
RESEARCH REPORT

N° 9294

October 2019

Project-Teams TADaaM and Roma

ISSN 0249-6399 ISRN INRIA/RR--9294--FR+ENG



Reservation and Checkpointing Strategies for Stochastic Jobs (Extended Version)

Ana Gainaru*, Brice Goglin†, Valentin Honoré†, Guillaume Pallez†, Padma Raghavan*, Yves Robert‡, Hongyang Sun *

Project-Teams TADaaM and Roma

Research Report n° 9294 — October 2019 — 36 pages

Abstract: In this work, we are interested in scheduling and checkpointing stochastic jobs on a reservation-based platform. We assume that jobs can be interrupted at any time to take a checkpoint, and that job execution times follow a known probability distribution. The user has to determine a sequence of fixed-length reservation requests, and to decide whether to checkpoint the state of the execution, or not, at the end of each request. The execution of the job is successful only when it terminates within a request, otherwise it must be resubmitted, using the next request in the reservation sequence, and restarting execution from the last checkpointed state. The cost of each reservation depends on both its duration and on the actual utilization of the platform during that request, which includes a restart if some previous reservation was terminated with a checkpoint, and possibly a checkpoint at the end of the current request. The cost of a job is then the cumulated cost of all the reservations that were needed until its completion. Overall, the objective is to find a reservation sequence that minimizes the total expected cost to execute a job. We provide an optimal strategy for discrete probability distributions of job execution times, and we design fully polynomial-time approximation strategies for continuous distributions with bounded support. We experimentally evaluate these strategies for jobs following a wide range of usual probability distributions, as well as one distribution obtained from traces of a neuroscience application. We compare our strategies with standard approaches that use periodic-length reservations (the next reservation is longer than the previous one by a constant amount of time) and simple checkpointing strategies (either checkpoint all reservations, or none).

Key-words: scheduling, checkpointing, stochastic cost, computing platform, sequence of requests, neuroscience applications

* Department of EECS, Vanderbilt University, Nashville, TN, USA
† Inria, LaBRI, Univ. Bordeaux
‡ Laboratoire LIP, ENS Lyon & University of Tennessee Knoxville, Lyon, France

RESEARCH CENTRE
BORDEAUX – SUD-OUEST

200 avenue de la Vieille Tour
33405 Talence Cedex

Stratégies de réservation avec points de sauvegarde pour l'ordonnancement de tâches stochastiques

Résumé :

Dans ce rapport, nous nous intéressons à l'ordonnancement de tâches stochastiques exécutées sur une plateforme à réservations, où l'utilisateur réalise des requêtes successives de temps de calcul. Le temps d'exécution des tâches considérées n'est pas connu à l'avance. Ce temps est représenté par une loi de probabilité, décrite sous la forme d'une densité de probabilité. Nous nous intéressons à ordonnancer une instance d'une telle tâche, c'est à dire que nous ne connaissons pas son temps d'exécution qui reste constant tout au long de l'ordonnancement. Dans ce cas, le coût de l'ordonnancement dépend à la fois de la durée des requêtes et du temps d'exécution de la tâche considérée.

Nous supposons de plus que la tâche peut être interrompue à tout instant (tâche divisible) pour un point de sauvegarde. Nous avons donc la possibilité de prendre un point de sauvegarde à la fin de certaines réservations bien choisies. L'avantage est de pouvoir repartir de l'état sauvegardé à la prochaine réservation au lieu de repartir du début, si la tâche ne termine pas pendant la réservation courante. Le prix à payer est le temps de sauvegarde, et de redémarrage.

L'objectif de ce travail est de déterminer une stratégie de réservation optimale qui minimise le coût total de l'ordonnancement. Une stratégie de réservations est une séquence de requêtes croissantes, qui sont payées les unes à la suite des autres en ordre croissant jusqu'à complétion de la tâche. Pour chaque réservation, nous indiquons si un point de sauvegarde doit être pris ou non.

Nous décrivons une telle solution optimale pour des distributions de probabilité discrètes, et nous donnons un schéma d'approximation polynomial pour les distributions continues à support compact. Nous comparons expérimentalement ces stratégies aux stratégies usuelles qui incrémentent la longueur de chaque réservation par une valeur constante, et qui décident de sauvegarder soit toutes les réservations, soit aucune. Nous utilisons un grand nombre de lois de probabilité usuelles (i.e. Uniforme, Exponentielle, Log-Normale, Weibull, Beta etc), ainsi qu'une distribution basée sur l'interpolation de traces d'applications de neurosciences exécutées sur une plateforme HPC.

Mots-clés : ordonnancement, coût stochastique, plateformes de calcul, séquence de réservations, point de sauvegarde, applications de neurosciences

Contents

1 Introduction

In this report, we revisit our recent work on reservation strategies for stochastic jobs [3]. Stochastic jobs originate from Big Data or Machine Learning workloads, whose performance is widely dependent on characteristics of input data. Figure 1 shows an example of a Neuroscience job. Reservation strategies provide a sequence of fixed length reservations to execute a stochastic job. If the reservation is too short for the job, it is restarted in a longer reservation. We extend the approach to include the possibility of checkpointing at the end of some (well-chosen) reservations. The idea of checkpointing is very natural and widely used in practice, in particular for long jobs lasting several hours, but it dramatically complicates the design of scheduling strategies. To the best of our knowledge, existing approaches either checkpoint at the end of all reservations, or never. For large-scale applications, checkpointing to save intermediate results at the end of each reservation is the de facto standard approach.

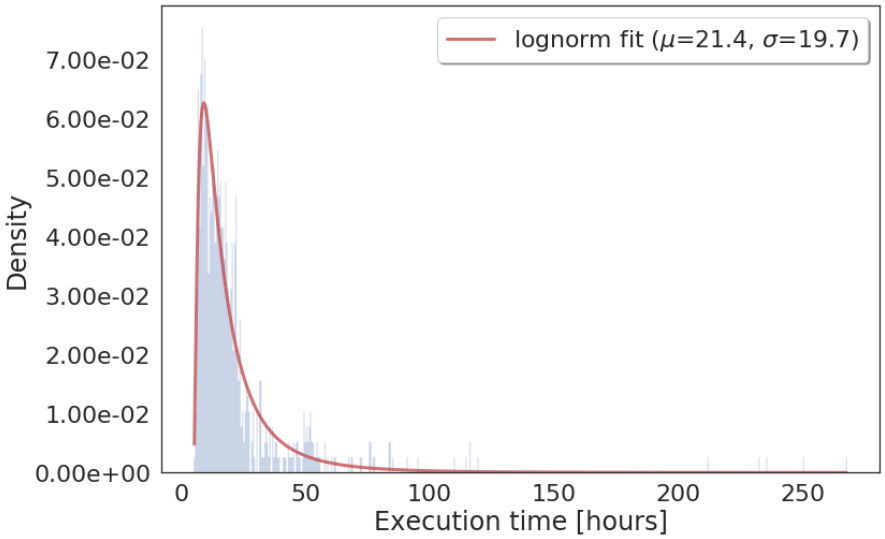


Figure 1: Execution times from 2017 for a *Structural identification of orbital anatomy* application, and its fitted distribution (in red).

We use an example to help understand the challenges of the problem under study. Consider the jobs depicted in Figure 1. We model their execution time with $\mathcal{D}$, a truncated LogNormal probability distribution on the domain $[a, b] = [1\text{mn}, 80\text{h}]$ (mean $\mu = 21\text{h}$, standard deviation $\sigma = 20\text{h}$). The exact execution time X of the next job to be scheduled is not known until that job has successfully completed, but instead is randomly and uniformly sampled from the target probability distribution $\mathcal{D}$. We want to minimize the expected cost of scheduling this job. To do so, we have to derive a sequence of reservations. Then we compute the cost of the job given that sequence, and aim at minimizing the expected value. To determine the cost of a reservation, we use the generic model from our previous work [3]. This model has been shown to encompass a variety of scenarios, ranging from the *Reserved Instances* of Cloud Computing where one pays (for a cheaper cost) only the reserved time [2], to High-Performance Computing (HPC) platforms where one pays the total execution time (wait time and runtime).

Specifically, for a reservation of length W_1 and an actual execution duration of length X, the cost is expressed as:

$$\alpha W_1 + \beta \min(W_1, X) + \gamma \tag{1}$$

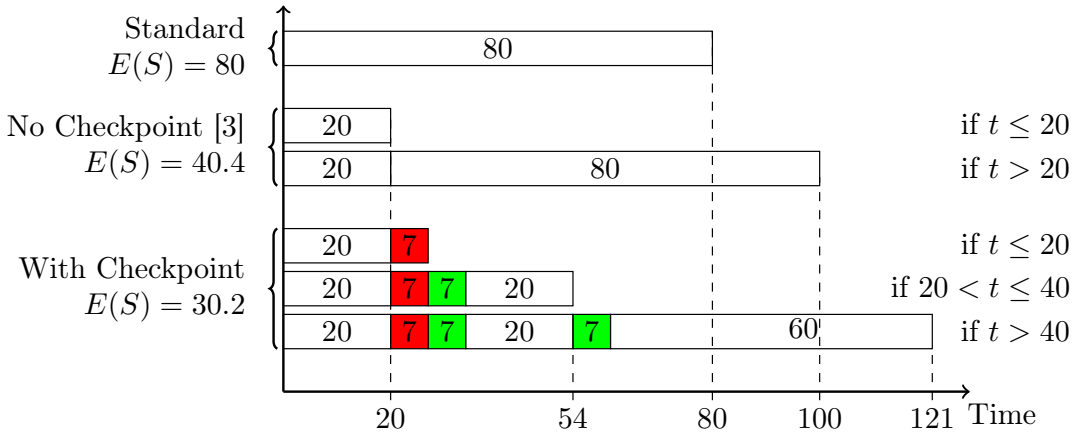


Figure 2: Illustration of different reservation strategies. The checkpoint (red) and restart (green) costs are equal to 7.

where α, β and γ are constant parameters that depend on the platform and the cost model. The first component αW_1 is proportional to the reservation length (pay for what you ask). The second component $\beta \min(W_1, X)$ is proportional to the actual execution time (pay for what you use). Finally, the third and last component is a start-up time possibly associated with the first and/or second components.

To illustrate the contribution of this work, we use $\alpha = 1, \beta = \gamma = 0$ in the example. In Figure 2 , we depict three strategies, and their expected costs in hours: (i) S_1 (Standard), which reserves the upper bound of $\mathcal{D}$, $W_1 = b = 80$; (ii) S_2 (No Checkpoint), which introduces a first reservation of size $W_1 = 20$ before the second reservation $W_2 = 80$; (iii) S_3 (With Checkpoint), which introduces a first checkpointed reservation of size $W_1 = 20 + 7$ (20 to cover jobs shorter than 20, and 7 (red box) is the cost to checkpoint), then a second non-checkpointed reservation of size $W_2 = 7 + 20$ (7 (green box) is the cost to restart, 20 to cover jobs larger than 20 and smaller than 40), and a third reservation of size $W_3 = 7 + 60$ (7 is the cost to restart, 60 to cover jobs of size up to b). Here is how to compute the expected cost of the last strategy S_3:

$$\begin{aligned}\mathbb{E}(S_3) &= 27 \cdot \mathbb{P}(X \leq 20) + 27 \cdot \mathbb{P}(20 < X \leq 40) + 67 \cdot \mathbb{P}(40 < X) \\ &= 27 \times 0.66 + 27 \times 0.26 + 67 \times 0.08 = 30.2.\end{aligned}$$

Note that $\tilde{S}_3$, the variant of S_3 where the second reservation is also checkpointed, would have a larger expected cost due to this second checkpoint: $\mathbb{E}(\tilde{S}_3) = 27 \times 0.66 + 34 \times 0.26 + 44 \times 0.08 = 30.42$. Similarly one can verify that not performing the second reservation at all would also have increased the expected cost. This example shows that checkpointing does help for some scenarios but has too much overhead for others, and suggests that finding the best trade-off is difficult.

Indeed, in the general case, one has to decide which reservations should be checkpointed, depending on application profile and platform parameters. Moreover, determining the expected cost of a given reservation sequence together with scheduling decisions gets quite complicated. Section 2 gives a detailed formula for the expected cost, and Theorem 1 in Section 3.1 provides a simplified version. In our previous work without checkpoints [3], we have been able to analytically characterize the optimal sequence of reservations for any smooth probability distribution (except the length of the first reservation which had to be found numerically). The problem with checkpoints is dramatically more difficult, but we provide a holistic approach: we show how to

compute the optimal solution for any discrete probability distribution, using a sophisticated dynamic programming algorithm. Then we show how to approximate the optimal solution for any continuous probability distribution with bounded support, by providing a reservation sequence (and its checkpointing decisions) whose expected cost is arbitrarily close to the optimal one. In practice, the restriction to bounded support is not a limitation. Given, say, a Lognormal or Weibull probability distribution defined on $[0, \infty)$, it is very natural to truncate it on a bounded interval $[a, b]$ where a corresponds to the quantile $Q(\varepsilon)$ and b to the quantile $Q(1-\varepsilon)$ for a small value of ε. This amount to discarding job execution times that are unreasonably too short or too long, and never encountered in practice.

The main contributions of this work are the following:

- The characterization of an optimal reservation sequence, together with its checkpointing decisions, for any discrete probability distribution, using a sophisticated dynamic programming algorithm.
- An approximation of the optimal solution for any continuous probability distribution with bounded support, by providing an algorithm to compute a reservation sequence (and its checkpointing decisions) whose expected cost is arbitrarily close to the optimal one.
- An extensive set of simulation results based on execution times from nine probability distributions and from neuroscience application traces.
- Experimentation with neuroscience applications on a manycore platform, showing the efficiency of our strategies in a HPC environment.

The rest of the paper is organized as follows. Section 2 introduces the framework and main notations, and provides a detailed formula for the expected cost of a reservation sequence ant its checkpointing decisions. Section 3 describes our key algorithmic contributions. Section 4 is devoted to experimental evaluation and comparison with existing approaches through simulation process. Section 5 establishes performance evaluation of real applications in an HPC framework. Section 6 presents the related work. Finally, we provide concluding remarks and hints for future work in Section 7.

2 Framework

In this section, we introduce some notations and formally define the optimization problem under study.

2.1 Stochastic jobs

We consider stochastic jobs whose execution times are unknown but (i) deterministic with respect to input data, so that two successive executions of the same job will have the same duration; and (ii) randomly and uniformly sampled from a given probability distribution law $\mathcal{D}$, whose density function (PDF) is f and cumulative distribution function (CDF) is F. The probability distribution is assumed to be nonnegative, since we model execution times, and it is defined either on a finite support $[a, b]$, where $0 \leq a < b$, or on an infinite support $[a, \infty)$ where $a \geq 0$.

Hence, the execution time of a job is a random variable X, and

$$\mathbb{P}(X \leq T) = F(T) = \int_a^T f(t)dt$$

For notational convenience, we sometimes extend the domain of f outside the support of $\mathcal{D}$ by letting $f(t) = 0$ for $t \in [0, a] \cup [b, \infty)$.

In addition, we assume that we can interrupt the jobs at any time (divisible load application) to take a checkpoint: this will save the current progress of the execution, and enable to restart

from that point on. We assume that the cost of checkpoint and of recovery is constant throughout the execution: let C be the cost to checkpoint the data at the end of an execution, and R the cost to read the data to restart a computation.

Remark 1. *By setting either C or R to ∞, then it is never useful to checkpoint, hence this problem can be reduced to the problem without checkpointing.*

2.2 Cost model

We use the cost model motivated in our previous work [3]. For a reservation of length W and an actual execution duration w for the job, the cost is $\alpha W + \beta \min(W, w) + \gamma$, where $\alpha > 0$, $\beta \geq 0$ and $\gamma \geq 0$. If the job does not complete within W seconds, then another reservation should be paid for.

However, we take checkpoints into account in this work. If the job did not complete its execution during last reservation, but was checkpointed during the last C seconds of that reservation, then in the current reservation, the job can restart from that checkpoint during the first R seconds, and then continue execution from its saved state. On the contrary, if no checkpoint was taken during the last reservation, the work done during that reservation is lost, and the execution must restart from the last checkpoint (or from the very beginning if no checkpoint was taken yet).

Altogether, the user needs to schedule a (possibly infinite) sequence of reservations $\mathcal{W} = (W_1, W_2, \dots, W_i, W_{i+1}, \dots)$ to execute any job whose execution time follows the distribution $\mathcal{D}$, and to launch these reservations one after the other, until the job successfully terminates within the duration of some reservation. In addition, the user should decide whether to take a checkpoint or not at the end of each reservation.

Reservation-based strategy

Definition 1 (Reservation sequence for $\mathcal{D}$)**.** Given a probability distribution $\mathcal{D}$, a *reservation sequence* $\mathcal{S} = \{(W_1, \delta_1), (W_2, \delta_2), \dots\}$, is defined as a sequence of reservation lengths W_k and a sequence of checkpointing decisions $\delta_k \in \{0, 1\}$: $\delta_k = 1$ means the k^{th} reservation ends with a checkpoint, and $\delta_k = 0$ means it does not.

Then, the k^{th} reservation can be decomposed into:

$$W_k = R_k + T_k + C_k \tag{2}$$

where R_k is the time spent for restart, T_k for actual job execution, and C_k for checkpoint. We have $C_k = \delta_k C$ by definition. There is a restart if and only if there has been a checkpoint at some point before, hence

$$R_k = (1 - \prod_{i=1}^{k-1}(1 - \delta_i))R$$

(assuming $R_1 = 0$ for the first reservation).

It is hard to keep track of actual job progress when using only the (W_k, δ_k) values. Consider for instance the following sequence $\mathcal{S} = \{(W_1, 1), (W_2, 0), (W_3, 1), (W_4, 0)\}$, which is depicted in Figure 3. If the actual job duration is $X = t$, during which reservation will the job complete its execution? We introduce another view of the reservation sequence $\mathcal{S}$ by introducing the milestones (t_k's) as shown in Figure 4. A milestone t_k represents the amount of work that has been actually executed at the end of the k^{th} reservation. Then, the last reservation for the job of length t is W_k, where $t_{k-1} \leq t \leq t_k$. Of course, we need that $t \leq t_4$ for all values of $\mathcal{D}$

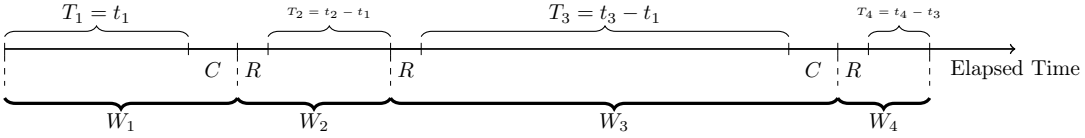


Figure 3: Graphical representation of elapsed time for the reservation sequence $\mathcal{S} = \{(W_1, 1), (W_2, 0), (W_3, 1), (W_4, 0)\}$.

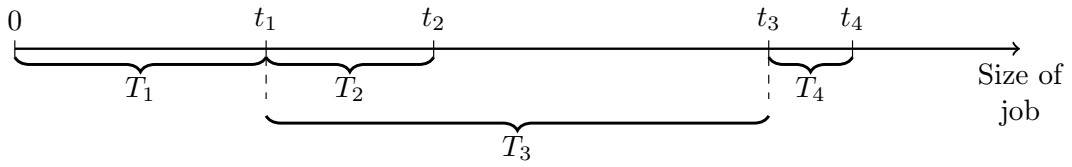


Figure 4: Graphical representation of job progress (and showing t_k versus T_k) for the reservation sequence $\mathcal{S} = \{(W_1, 1), (W_2, 0), (W_3, 1), (W_4, 0)\}$.

(equivalently, the upper bound of the support of $\mathcal{D}$ is $b \leq t_4$) for all jobs to complete successfully with the four reservations of $\mathcal{S}$.

The relationship between the milestone t_k (actual work progress) and the value of T_k (time spent computing during reservation W_k; see Equation (2)) is the following:

$$t_k = T_k + \sum_{i=1}^{k-1} \delta_i T_i \tag{3}$$

Indeed, the work actually progresses only from the last checkpoint, while the work executed during the previous non-checkpointed reservations is lost whenever these non-checkpointed reservations do not allow for the full completion of the job. Another way to express the relationship between t_k and T_k is the following:

$$t_k = T_k + \max\{t_i \;\Big|\; 1 \leq i \leq k-1 \text{ and } \delta_i = 1\} \tag{4}$$

Indeed, Equation (4) gives a recursive way to compute t_k from its definition. We recapitulate the relations between all notations introduced in Figures 3 and 4:

$$W_k = R_k + T_k + C_k \tag{5}$$

$$R_k = (1 - \Pi_{i<k}(1 - \delta_i))R \tag{6}$$

$$\begin{aligned} T_k &= t_k - \sum_{i<k} \delta_i T_i \\ &= t_k - \max\{t_i \;\Big|\; 1 \leq i \leq k-1 \text{ and } \delta_i = 1\} \end{aligned} \tag{7}$$

$$C_k = \delta_k C \tag{8}$$

In the following, we use milestones t_k rather than reservation lengths W_k to characterize a reservation sequence, and we write

$$\mathcal{S} = \{(t_1, \delta_1), (t_2, \delta_2), \dots\}$$

instead of

$$\mathcal{S} = \{(W_1, \delta_1), (W_2, \delta_2), \dots\}$$

because it is easier to use milestones when computing the expected cost of a sequence, as shown below. For notational convenience, we define $t_0 = 0$ as the first milestone of each sequence $\mathcal{S}$. Note also that we can restrict to sequences where $t_{k-1} < t_k$, because otherwise (if $t_{k-1} = t_k$), the execution does not progress during the k^{th} reservation.

2.3 Expected cost

Given a reservation sequence $\mathcal{S} = ((t_i, \delta_i))_i$ and a job with execution time t such that $t_{k-1} < t \leq t_k$, the cost of the sequence for that job is given by:

$$C_{\mathcal{S}}(k,t) = \sum_{i=1}^{k-1} (\alpha W_i + \beta W_i + \gamma) + \alpha W_k + \beta(R_k + t - (t_k - T_k)) + \gamma \tag{9}$$

where the first part is the total cost from the $k-1$ first reservations that did not allow the job to complete, and the second part is the cost of the k^{th} reservation. The actual execution time during the k^{th} reservation is $t - (t_k - T_k)$, because $t_k - T_k$ is the amount of work done up to the beginning of that reservation; we add the restart time (R_k) but do not need to checkpoint (if $\delta_k = 1$) because the job successfully completes before it is taken.

We let $k(t) = k$ for a job of length t such that $t_{k-1} < t \leq t_k$. Now, the expected cost of the reservation sequence $\mathcal{S}$ over all jobs following distribution $\mathcal{D}$ is

$$\mathbb{E}(\mathcal{S}) = \int_0^\infty C_{\mathcal{S}}(k(t), t) f(t) dt = \sum_{k=1}^{\infty} \int_{t_{k-1}}^{t_k} C_{\mathcal{S}}(k, t) f(t) dt \tag{10}$$

2.4 Objective

We are now ready to state the optimization problem:

Definition 2 (Stochastic)**.** Given a probability distribution $\mathcal{D}$ (with PDF f and CDF F) for the execution times of stochastic jobs, and given a cost function given by Equation (9) (with parameters $\alpha > 0$, $\beta \geq 0$ and $\gamma \geq 0$), find a reservation strategy $\mathcal{S}$ with minimal expected cost $\mathbb{E}(\mathcal{S})$ as given in Equation (10).

We further define ReservationOnly to be the instance of Stochastic where the cost is a linear function of the reservation length only, i.e., when $\beta = \gamma = 0$. For ReservationOnly, we can further consider $\alpha = 1$ without loss of generality. For instance, such costs are incurred when making reservations of resources to schedule jobs on some cloud platforms, with hourly or daily rates. Throughout the paper, we focus on the usual probability distributions, hence we assume that the density function f and the CDF F of $\mathcal{D}$ are smooth (infinitely differentiable), and that $\mathcal{D}$ has finite expectation.

3 Algorithms

In this section, we establish some key properties of an optimal solution in the general setting.

3.1 Expected cost

We start by establishing a simpler expression for the expected cost function of STOCHASTIC.

Theorem 1. *Given a reservation sequence $\mathcal{S} = ((t_1, \delta_1), (t_2, \delta_2), \dots)$, the expected cost $\mathbb{E}(\mathcal{S})$ of a strategy $\mathcal{S}$ given by Equation* (10)*, with parameters α, β and γ, can be rewritten as*

$$\begin{aligned}\mathbb{E}(\mathcal{S}) = \beta \cdot \mathbb{E}[X] + \alpha\big(t_1 + \delta_1 C\big) + \gamma \\ + \sum_{i=2}^{\infty} \Big(\alpha W_i + \beta\big(R_i + (1-\delta_{i-1})T_{i-1} + C_{i-1}\big) + \gamma\Big) \cdot \mathbb{P}\left(X > t_{i-1}\right)\end{aligned} \tag{11}$$

Proof. Firstly we rewrite Equation (11) as follows:

$$\mathbb{E}(\mathcal{S}) = \beta \cdot \mathbb{E}[X] + \sum_{i=1}^{\infty} \Big(\alpha W_i + \beta\big(R_i + (1-\delta_{i-1})T_{i-1} + C_{i-1}\big) + \gamma\Big) \cdot \mathbb{P}\left(X > t_{i-1}\right) \tag{12}$$

with initialization $\delta_0 = W_0 = R_1 = 0$ and $t_0 = 0$.

From Equations (9) and (10), we have

$$\mathbb{E}(\mathcal{S}) = \mathbb{E}_1 + \mathbb{E}_2 + \mathbb{E}_3 \tag{13}$$

where

$$\begin{aligned}\mathbb{E}_1 &= \sum_{k=1}^{\infty} \int_{t_{k-1}}^{t_k} \Big(\sum_{i=1}^{k} (\alpha W_i + \gamma)\Big) f(t)dt \\ \mathbb{E}_2 &= \sum_{k=1}^{\infty} \int_{t_{k-1}}^{t_k} \Big(\sum_{i=1}^{k-1} \beta W_i\Big) f(t)dt \\ \mathbb{E}_3 &= \sum_{k=1}^{\infty} \int_{t_{k-1}}^{t_k} \beta\Big(t + R_k + T_k - t_k\Big) f(t)dt\end{aligned}$$

Using $t_0 = 0$, we compute the first term as

$$\begin{aligned}\mathbb{E}_1 &= \sum_{k=1}^{\infty} \sum_{i=1}^{k} (\alpha W_i + \gamma) \int_{t_{k-1}}^{t_k} f(t)dt \\ &= \sum_{k=1}^{\infty} \sum_{i=1}^{k} (\alpha W_i + \gamma) \cdot \mathbb{P}\left(t_{k-1} < X \leq t_k\right) \\ &= \sum_{i=1}^{\infty} (\alpha W_i + \gamma) \sum_{k=i}^{\infty} \mathbb{P}\left(t_{k-1} < X \leq t_k\right) \\ &= \sum_{i=1}^{\infty} (\alpha W_i + \gamma) \cdot \mathbb{P}\left(X > t_{i-1}\right)\end{aligned}$$

Similarly, using $W_0 = 0$, we express the second term as:

$$\mathbb{E}_2 = \sum_{i=1}^{\infty} \beta W_i \cdot \mathbb{P}\left(X > t_i\right)$$

Finally, we derive the third term as:

$$\begin{aligned}\mathbb{E}_3 &= \int_{t_0}^{\infty} \beta t f(t)dt + \sum_{k=1}^{\infty} \int_{t_{k-1}}^{t_k} \beta\Big(R_k + T_k - t_k\Big) f(t)dt \\ &= \beta \cdot \mathbb{E}[X] + \sum_{i=1}^{\infty} \beta\Big(R_i + T_i - t_i\Big) \cdot \mathbb{P}\left(t_{i-1} < X \leq t_i\right)\end{aligned}$$

Plugging these three terms back into Equation (13), we get:

$$\begin{aligned}\mathbb{E}(\mathcal{S}) &= \beta \cdot \mathbb{E}[X] + \sum_{i=1}^{\infty} \left(\alpha W_i + \beta W_{i-1} + \gamma\right) \cdot \mathbb{P}\left(X > t_{i-1}\right) + \sum_{i=1}^{\infty} \beta\left(R_i + T_i - t_i\right) \cdot \mathbb{P}\left(t_{i-1} < X \leq t_i\right) \\ &= \beta \cdot \mathbb{E}[X] + \sum_{i=1}^{\infty} \left(\alpha W_i + \beta(R_i + T_{i-1} + C_{i-1}) + \gamma\right) \cdot \mathbb{P}\left(X > t_{i-1}\right) - \sum_{i=1}^{\infty} \beta\left(t_i - T_i\right) \cdot \mathbb{P}\left(t_{i-1} < X \leq t_i\right)\end{aligned} \tag{14}$$

For the last derivation, we used

$$\sum_{i=1}^{\infty} (R_{i-1} \cdot \mathbb{P}\left(X > t_{i-1}\right) + R_i \cdot \mathbb{P}\left(t_{i-1} < X \leq t_i\right)) = \sum_{i=1}^{\infty} (R_i \cdot \mathbb{P}\left(X > t_{i-1}\right)$$

Now, we study the second part of Equation (14) above. For all $j \leq 1$, let $\phi^o(j)$ denote the index of the j^{th} checkpointed reservation of $\mathcal{S}$. For instance in the example of Figures 3 and 4, $\phi^o(1) = 1$ and $\phi^o(2) = 3$. Then, using Equation (7), we have

$$\begin{aligned}\sum_{i=1}^{\infty} \left(t_i - T_i\right) \cdot \mathbb{P}\left(t_{i-1} < X \leq t_i\right) &= \sum_{j=1}^{\infty} t_{\phi(j)} \cdot \mathbb{P}\left(t_{\phi(j)} < X \leq t_{\phi(j+1)}\right) \\ &= \sum_{j=1}^{\infty} T_{\phi(j)} \cdot \mathbb{P}\left(X > t_{\phi(j)}\right) \\ &= \sum_{i=1}^{\infty} \delta_i T_i \cdot \mathbb{P}\left(X > t_i\right)\end{aligned}$$

Plugging the above back into Equation (14), we get the desired result shown in Equation (12). □

3.2 Dynamic programming for discrete distributions

We study the problem for a finite discrete distribution: $Y \sim (v_i, f_i)_{1 \leq i \leq n}$, where $v_i < v_{i+1}$ for all $1 \leq i \leq n-1$ and $f_i = \mathbb{P}\left(Y = v_i\right)$. We assume that $f_n \neq 0$ and $\sum_{i=1}^{n} f_i = 1$. Consider a strategy $\mathcal{S} = \{(t_1, \delta_1), (t_2, \delta_2), \ldots, (t_{|\mathcal{S}|}, \delta_{|\mathcal{S}|})\}$, where $t_i = v_{\pi(i)}$ and $t_i < t_{i+1}$ for all $1 \leq i \leq |\mathcal{S}| - 1$. Also, the last reservation is necessarily $t_{|\mathcal{S}|} = v_n$ to ensure that the expected cost of the strategy is finite. By convention, we let $t_0 = v_0 = a$, hence $\mathbb{P}\left(Y > t_0\right) = 1$. Note that we can safely restrict to strategies where each milestone t_i is equal to some threshold v_j of the discrete distribution: otherwise, replacing t_i by the largest v_j such that $v_j \leq t_i$ leads to a smaller cost.

Rewriting Equation (11) with $W_i = R_i + T_i + C_i$, and since $W_0 = 0$, the expected cost of

strategy $\mathcal{S}$ can be expressed as:

$$\begin{aligned}\mathbb{E}(\mathcal{S}) = \beta \cdot \mathbb{E}[Y] \\ + \sum_{i=1}^{|\mathcal{S}|} \big(\alpha\left(R_i + T_i + C_i\right) + \beta R_i + \gamma\big) \cdot \mathbb{P}\left(Y > t_{i-1}\right) \\ + \sum_{i=1}^{|\mathcal{S}|-1} \beta\big((1-\delta_i)T_i + C_i\big) \cdot \mathbb{P}\left(Y > t_i\right)\end{aligned} \tag{15}$$

Based on Equation (15), and using Equations (6) to (8), we construct a dynamic programming algorithm to compute the optimal reservation sequence:

Theorem 2. *For a discrete distribution $Y \sim (v_i, f_i)_{1\le i\le n}$, the optimal expected cost is returned by $\mathbb{E}_{\text{ckpt}}(0,0)$, where, for $0 \le i_c \le i_l \le n$, $\mathbb{E}_{\text{ckpt}}(i_c, i_l)$ is:*

$$\begin{aligned}&= \beta \cdot \mathbb{E}[Y], && \text{if } i_l = n \\ &= \min_{\substack{i_l+1\le j\le n,\\ \Delta_j\in\{0,1\}}} \left(\mathbb{E}_{\text{ckpt}}\big(\Delta_j j, j\big) + \Big(\alpha\big(v_j + \Delta_j C\big) + \gamma\Big) \cdot \sum_{k=i_l+1}^{n} f_k + \beta\big((1-\Delta_j)v_j + \Delta_j C\big) \cdot \sum_{k=j+1}^{n} f_k \right) \\ & && \text{if } i_c = 0 \\ &= \min_{\substack{i_l+1\le j\le n,\\ \Delta_j\in\{0,1\}}} \left(\mathbb{E}_{\text{ckpt}}\big((1-\Delta_j)i_c + \Delta_j j, j\big) + \Big(\alpha\big(R + (v_j - v_{i_c}) + \Delta_j C\big) + \beta R + \gamma\Big) \cdot \sum_{k=i_l+1}^{n} f_k \right. \\ &\quad \left. + \beta\big((1-\Delta_j)(v_j - v_{i_c}) + \Delta_j C\big) \cdot \sum_{k=j+1}^{n} f_k \right), && \text{otherwise}\end{aligned}$$

The optimal solution can be computed in $O(n^3)$ time.

Intuitively, i_c denotes the index of the last checkpointed value, while i_l denotes the index of the last value that was tried before we try the next one with index j. Here, Δ_j indicates whether the value v_j will be checkpointed or not.

Proof. To prove the optimality, consider $\mathbb{E}(\mathcal{S})$ given in Equation (15) for any reservation sequence:

$$\begin{aligned}\mathcal{S} &= \{(t_1, \delta_1), \ldots, (t_{|\mathcal{S}|}, \delta_{|\mathcal{S}|})\} \\ &= \{(v_{\pi(1)}, \Delta_{\pi(1)}), \ldots, (v_{\pi(|\mathcal{S}|)}, \Delta_{\pi(|\mathcal{S}|)})\}\end{aligned}$$

and define $\mathbb{E}_\ell$ as the following partial sum:

$$\begin{aligned}\mathbb{E}_\ell = \beta \cdot \mathbb{E}[Y] \\ + \sum_{i=\ell+1}^{|\mathcal{S}|} \big(\alpha\left(R_i + T_i + C_i\right) + \beta R_i + \gamma\big) \cdot \mathbb{P}\left(Y > t_{i-1}\right) \\ + \sum_{i=\ell+1}^{|\mathcal{S}|-1} \beta\big((1-\delta_i)T_i + C_i\big) \cdot \mathbb{P}\left(Y > t_i\right)\end{aligned} \tag{16}$$

Note that $\mathbb{E}_0 = \mathbb{E}(\mathcal{S})$. We show by induction that the following invariant is true for all $\ell = |\mathcal{S}|, |\mathcal{S}|-1, \ldots, 0$:

$$\mathbb{E}_\ell \ge \mathbb{E}_{\text{ckpt}}(\pi(\bar{\ell}), \pi(\ell)) \tag{17}$$

where $\pi(\overline{\ell})$ is the index of the last reservation not larger than $\pi(\ell)$ and such that $\Delta_{\pi(\overline{\ell})} = 1$. We denote the corresponding reservation by $t_{\overline{\ell}} = v_{\pi(\overline{\ell})}$.

For the base case with $\ell = |\mathcal{S}|$, we have $\pi(|\mathcal{S}|) = n$, and $\mathbb{E}_{|\mathcal{S}|} = \beta \cdot \mathbb{E}[Y] = \mathbb{E}_{\text{ckpt}}(\pi(\overline{|\mathcal{S}|}), n)$. Now, suppose $\mathbb{E}_{\ell+1} \geq \mathbb{E}_{\text{ckpt}}(\pi(\overline{\ell+1}), \pi(\ell+1))$ for $\ell + 1 \leq |\mathcal{S}|$. Here, we note that $\pi(\overline{\ell+1}) = \pi(\ell+1)$ if $\Delta_{\pi(\ell+1)} = 1$ (i.e., if $v_{\pi(\ell+1)}$ is checkpointed). Otherwise, we have $\pi(\overline{\ell+1}) = \pi(\overline{\ell})$. Then, from Equation (16), we derive:

$$
\begin{aligned}
\mathbb{E}_\ell &= \mathbb{E}_{\ell+1} + \big(\alpha\left(R_{\ell+1} + T_{\ell+1} + C_{\ell+1}\right) + \beta R_{\ell+1} + \gamma\big) \cdot \mathbb{P}\left(Y > t_\ell\right) \\
&\quad + \beta\big((1-\delta_{\ell+1})T_{\ell+1} + C_{\ell+1}\big) \cdot \mathbb{P}\left(Y > t_{\ell+1}\right) \\
&\geq \mathbb{E}_{\text{ckpt}}(\pi(\overline{\ell+1}), \pi(\ell+1)) + \Big(\alpha\left(R_{\ell+1} + (t_{\ell+1} - t_{\overline{\ell}}) + C_{\ell+1}\right) + \beta R_{\ell+1} + \gamma\Big) \cdot \mathbb{P}\left(Y > v_{\pi(\ell)}\right) \\
&\quad + \beta\big((1-\delta_{\ell+1})(t_{\ell+1} - t_{\overline{\ell}}) + C_{\ell+1}\big) \cdot \mathbb{P}\left(Y > v_{\pi(\ell+1)}\right) \\
&= \mathbb{E}_{\text{ckpt}}\left((1-\Delta_{\pi(\ell+1)})\pi(\overline{\ell}) + \Delta_{\pi(\ell+1)}\pi(\ell+1), \pi(\ell+1)\right) \\
&\quad + \Big(\alpha\big(\mathbb{1}_{\pi(\overline{\ell})\neq 0}R + (v_{\pi(\ell+1)} - v_{\pi(\overline{\ell})}) + \Delta_{\pi(\ell+1)}C\big) + \beta\mathbb{1}_{\pi(\overline{\ell})\neq 0}R + \gamma\Big) \cdot \sum_{k=\pi(\ell)+1}^{n} f_k \\
&\quad + \beta\left((1-\Delta_{\pi(\ell+1)})(v_{\pi(\ell+1)} - v_{\pi(\overline{\ell})}) + \Delta_{\pi(\ell+1)}C\right) \cdot \sum_{k=\pi(\ell+1)+1}^{n} f_k \\
&\geq \mathbb{E}_{\text{ckpt}}\left(\pi(\overline{\ell}), \pi(\ell)\right)
\end{aligned}
$$

In the derivation above, the first inequality is due to the inductive hypothesis, and using $T_{\ell+1} = t_{\ell+1} - t_{\overline{\ell}}$ from Equation (7). The last inequality is due to the definition of $\mathbb{E}_{\text{ckpt}}$. s Thus, by induction, we get $\mathbb{E}_{\text{ckpt}}(0,0) = \mathbb{E}_{\text{ckpt}}(\pi(\overline{0}), \pi(0)) \leq \mathbb{E}_0 = \mathbb{E}(\mathcal{S})$. This shows that $\mathbb{E}_{\text{ckpt}}(0,0)$ is not greater than the expected cost of any reservation sequence $\mathcal{S}$, thus returning the optimal solution.

Finally, one can pre-compute values of $\sum_{k=\ell+1}^{n} f_k$ for all $0 \leq \ell < n$ (in linear time) and store them. Then, computing $\mathbb{E}_{\text{ckpt}}(i_c, i_l)$ depends on $2(n - i_l)$ other $\mathbb{E}_{\text{ckpt}}$ values, thus takes $O(n - i_l)$ time. The overall complexity is therefore $O(n^3)$. □

3.3 Approximation algorithm for continuous distributions

In this section, we provide an approximation algorithm of the optimal strategy for continuous distributions with bounded support $[a, b]$, where $a > 0$ and b is finite. Because we model job execution times, it is very natural to truncate continuous distributions whose support is $[0, \infty[$ such as an Exponential or Lognormal distribution, say, to a bounded support $[a, b]$ with (very) small a and (very) large b. This underlies the significance of our approximation result, which returns an arbitrarily good quality solution with low complexity. More precisely, let X be a continuous random variable defined on $[a, b]$ modeling the probability distribution $\mathcal{D}$, where $0 < a < b$, with CDF F and PDF f. Theorem 3 shows that Algorithm 1 computes a close-to-optimal strategy for STOCHASTIC:

Theorem 3. *Given a continuous random variable X on the domain $[a, b]$, where $0 < a < b$, and given a constant $\varepsilon > 0$,* DYN-PROG-COUNT(X, ε) *is a $1 + \varepsilon$-approximation algorithm for* STOCHASTIC *and executes in time $\mathcal{O}\left(\frac{1}{\varepsilon^3}\right)$.*

Proof. Given the continuous random variable X of support $[a, b]$, we define the discrete random

Algorithm 1 DYN-PROG-COUNT(X, ε)

Let $[a, b]$ be the domain of X, with $0 < a < b$
$c_0 = (b-a)\min\left(\frac{1}{\min(a,R,C)}, \frac{\alpha+\beta}{\gamma}\right)$
$n \leftarrow \lceil c_0/\varepsilon \rceil$
Define the discrete distribution $Y_n \sim (v_i, f_i)_{i=1\ldots n}$ s.t.

$$v_i = a + i \cdot \frac{b-a}{n};$$
$$f_i = \begin{cases} \mathbb{P}(Y_n = v_1) = \mathbb{P}(X \leq v_1) & \text{if } i = 1 \\ \mathbb{P}(Y_n = v_i) = \mathbb{P}(v_{i-1} < X \leq v_i) & \text{otherwise.} \end{cases}$$

$\mathcal{S}_n^{\mathrm{dp}} \leftarrow$ Optimal strategy for Y_n (Theorem 2)
return $\mathcal{S}_n^{\mathrm{dp}}$

variable $Y_n \sim (v_i, f_i)_{i=1\ldots n}$ as stated in Algorithm 1:

$$\begin{aligned} v_i &= a + i \cdot \frac{b-a}{n} \\ f_i &= \mathbb{P}(Y_n = v_i) = \mathbb{P}(v_{i-1} < X \leq v_i) \end{aligned} \tag{18}$$

For any random variable Z and a strategy $\mathcal{S}$, we denote by $\mathbb{E}(\mathcal{S}(Z))$ the expected cost given by Equation (11):

$$\mathbb{E}(\mathcal{S}(Z)) = \beta \cdot \mathbb{E}[Z] + \sum_{i=1}^{|\mathcal{S}|} \Big(\alpha W_i + \beta\big(R_i + (1-\delta_{i-1})T_{i-1} + C_{i-1}\big) + \gamma\Big) \cdot \mathbb{P}(Z > t_{i-1}) \tag{19}$$

with initialization $t_0 = \delta_0 = W_0 = R_1 = 0$.

Let $\mathcal{S}^{\mathrm{opt}} = ((t_i^o, \delta_i^o))_{1 \leq i \leq |\mathcal{S}^{\mathrm{opt}}|}$ denote the optimal solution for X, and let $\mathcal{S}_n^{\mathrm{dp}}$ denote the optimal solution for Y_n returned by Theorem 2. We want to show that

$$\mathbb{E}(\mathcal{S}_n^{\mathrm{dp}}(X)) \leq (1 + \frac{1}{c \cdot n})\mathbb{E}(\mathcal{S}^{\mathrm{opt}}(X))$$

In order to do that, we construct an intermediate strategy

$$\mathcal{S}^{\mathrm{algo}} = ((t_i^a, \delta_i^a))_{1 \leq i \leq |\mathcal{S}^{\mathrm{opt}}|}$$

(hence $|\mathcal{S}^{\mathrm{algo}}| = |\mathcal{S}^{\mathrm{opt}}|$), where for $1 \leq i \leq |\mathcal{S}^{\mathrm{opt}}|$, we let

$$(t_i^a, \delta_i^a) = (v_{\pi^o(i)}, \delta_i^o)$$

Here, we use the sequence $(v_i)_{i=0\ldots n}$ from Equation (18), and the function π^o defined below:

$$v_{\pi^o(i)-1} < t_i^o \leq v_{\pi^o(i)} \tag{20}$$

In other words, for each reservation, $\mathcal{S}^{\mathrm{algo}}$ chooses the first discrete value larger than or equal to the corresponding one chosen by $\mathcal{S}^{\mathrm{opt}}$, and makes the same checkpointing decision.

Lemma 1. $\mathbb{E}(\mathcal{S}^{\mathrm{algo}}(X)) \leq (1+\varepsilon)\mathbb{E}(\mathcal{S}^{\mathrm{opt}}(X))$

Proof. We use the notations T_i^o, R_i^o, C_i^o, W_i^o for the parameters of $\mathcal{S}^{\text{opt}}$, and T_i^a, R_i^a, C_i^a, W_i^a for the parameters of $\mathcal{S}^{\text{algo}}$. From Equations (5) to (8), we see that for $1 \leq i \leq |\mathcal{S}^{\text{opt}}|$, we have:

1. $\delta_i^o = \delta_i^a$;
2. $R_i^o = R_i^a$;
3. $C_i^o = C_i^a$; and
4. $W_i^a - W_i^o = T_i^a - T_i^o$.

In addition, if $\sigma^o(i)$ (resp. $\sigma^a(i)$) is the index of the last checkpoint before t_i^o (resp. t_i^a), then $\sigma^o(i) = \sigma^a(i)$, and,

$$\begin{aligned}
|T_i^a - T_i^o| &= \left|\left(t_i^a - t_{\sigma^a(i)}^a\right) - \left(t_i^o - t_{\sigma^o(i)}^o\right)\right| \\
&= \left|\left(v_{\pi^o(i)} - v_{\pi^o(\sigma^o(i))}\right) - \left(t_i^o - t_{\sigma^o(i)}^o\right)\right| \\
&= \left|\left(v_{\pi^o(i)} - t_i^o\right) - \left(v_{\pi(\sigma^o(i))} - t_{\sigma^o(i)}^o\right)\right| \\
&\leq \max\left(v_{\pi^o(i)} - t_i^o; v_{\pi^o(\sigma^o(i))} - t_{\sigma^o(i)}^o\right) \leq \frac{b-a}{n}
\end{aligned}$$

From Equation (19) we have:

$$\mathbb{E}(\mathcal{S}^{\text{opt}}(X)) = \beta\mathbb{E}[X] + \sum_{i=1}^{|\mathcal{S}^{\text{opt}}|} \left(\alpha W_i^o + \beta\big(R_i^o + (1-\delta_{i-1}^o)T_{i-1}^o + C_{i-1}^o\big) + \gamma\right) \cdot \mathbb{P}\left(X > t_{i-1}^o\right)$$

$$\mathbb{E}(\mathcal{S}^{\text{algo}}(X)) = \beta\mathbb{E}[X] + \sum_{i=1}^{|\mathcal{S}^{\text{opt}}|} \left(\alpha W_i^a + \beta\big(R_i^a + (1-\delta_{i-1}^a)T_{i-1}^a + C_{i-1}^a\big) + \gamma\right) \cdot \mathbb{P}\left(X > t_{i-1}^a\right)$$

We study the difference $\mathbb{E}(\mathcal{S}^{\text{algo}}(X)) - \mathbb{E}(\mathcal{S}^{\text{opt}}(X))$. We already know that $\mathbb{E}(\mathcal{S}^{\text{algo}}(X)) - \mathbb{E}(\mathcal{S}^{\text{opt}}(X)) \geq 0$ because $\mathcal{S}^{\text{opt}}$ is optimal. We observe that $\mathbb{P}\left(X > t_{i-1}^a\right) \leq \mathbb{P}\left(X > t_{i-1}^o\right)$ because $t_{i-1}^a \geq t_{i-1}^o$. We derive that:

$$\begin{aligned}
\mathbb{E}(\mathcal{S}^{\text{algo}}(X)) - \mathbb{E}(\mathcal{S}^{\text{opt}}(X)) &\leq \sum_{i=1}^{|\mathcal{S}^{\text{opt}}|} \left(\alpha(W_i^a + \beta\big(R_i^a + (1-\delta_{i-1}^a)T_{i-1}^a + C_{i-1}^a\big) + \gamma\right) \cdot \mathbb{P}\left(X > t_{i-1}^o\right) \\
&\quad - \sum_{i=1}^{|\mathcal{S}^{\text{opt}}|} \left(\alpha(W_i^o + \beta\big(R_i^o + (1-\delta_{i-1}^o)T_{i-1}^o + C_{i-1}^o\big) + \gamma\right) \cdot \mathbb{P}\left(X > t_{i-1}^o\right) \\
&\leq \sum_{i=1}^{|\mathcal{S}^{\text{opt}}|} \left(\alpha|T_i^a - T_i^o| + \beta(1-\delta_{i-1}^o)|T_{i-1}^a - T_{i-1}^o|\right) \cdot \mathbb{P}\left(X > t_{i-1}^o\right) \\
&\leq \alpha\frac{b-a}{n} + \sum_{i=1}^{|\mathcal{S}^{\text{opt}}|-1} \left((\alpha + \beta(1-\delta_i^o))\frac{b-a}{n}\right) \cdot \mathbb{P}\left(X > t_i^o\right) \\
&\leq \frac{b-a}{n}\left(\alpha + (\alpha+\beta)\sum_{i=1}^{|\mathcal{S}^{\text{opt}}|-1} \mathbb{P}\left(X > t_i^o\right)\right)
\end{aligned}$$

We observe that:

$$\mathbb{E}(\mathcal{S}^{\text{opt}}(X)) \geq \gamma + \sum_{i=1}^{|\mathcal{S}^{\text{opt}}|-1} \gamma \cdot \mathbb{P}\left(X > t_i^o\right)$$

and because for $1 \leq i \leq |\mathcal{S}^{\text{opt}}|$, $W_i^o \geq \min(a, R)$ (either $T_i^o \geq a$, or $R_i^o = R$), we derive that

$$\mathbb{E}(\mathcal{S}^{\text{opt}}(X)) \geq \min(a, R, C) \left(\alpha + (\alpha + \beta) \sum_{i=1}^{|\mathcal{S}^{\text{opt}}|-1} \mathbb{P}\left(X > t_i^o\right) \right).$$

Using the definition of $c_0 = (b-a)\min\left(\frac{1}{\min(a,R,C)}, \frac{\alpha+\beta}{\gamma}\right)$ in Algorithm 1, we obtain

$$\begin{aligned}\mathbb{E}(\mathcal{S}^{\text{algo}}(X)) - \mathbb{E}(\mathcal{S}^{\text{opt}}(X)) &\leq \frac{c_0}{n} \cdot \mathbb{E}(\mathcal{S}^{\text{opt}}(X)) \\ &\leq \varepsilon \cdot \mathbb{E}(\mathcal{S}^{\text{opt}}(X))\end{aligned}$$

which concludes the proof of Lemma 1. □

Lemma 2. $\mathbb{E}(\mathcal{S}_n^{\text{dp}}(X)) \leq \mathbb{E}(\mathcal{S}^{\text{algo}}(X))$

Proof. Given any reservation strategy $\mathcal{S} = ((t_i, \delta_i))_{1 \leq i \leq |\mathcal{S}|}$ such that $\forall i,\ t_i \in \{v_1, \ldots, v_n\}$, we show that:

$$\mathbb{E}(\mathcal{S}(Y_n)) - \mathbb{E}(\mathcal{S}(X)) = \beta\left(\mathbb{E}[Y_n] - \mathbb{E}[X]\right)$$

Indeed, for the two distributions Y_n and X, the only differences in the cost function are: (i) the expectations $\mathbb{E}[Y_n]$ and $\mathbb{E}[X]$; and (ii) the probability values $\mathbb{P}\left(Y_n > t_i\right)$ and $\mathbb{P}\left(X > t_i\right), \forall i$. But if $t_i \in \{v_1, \ldots, v_n\}$, we have:

$$\begin{aligned}\mathbb{P}\left(Y_n > t_i\right) &= \mathbb{P}\left(Y_n > v_k\right) \\ &= \mathbb{P}\left(Y_n \in \cup_{j=k+1}^{n} \{v_j\}\right) = \sum_{j=k+1}^{n} \mathbb{P}\left(Y_n = v_j\right) \\ &= \sum_{j=k+1}^{n} \mathbb{P}\left(X \in]v_{j-1}, v_j]\right) = \mathbb{P}\left(X \in]v_k, v_n]\right) \\ &= \mathbb{P}\left(X > v_k\right) = \mathbb{P}\left(X > t_i\right)\end{aligned}$$

We obtain that:

$$\mathbb{E}(\mathcal{S}(Y_n)) - \mathbb{E}(\mathcal{S}(X)) = \beta\left(\mathbb{E}[Y_n] - \mathbb{E}[X]\right)$$

We apply this result to both $\mathcal{S}_n^{\text{dp}}$ and $\mathcal{S}^{\text{algo}}$ and derive that:

$$\mathbb{E}(\mathcal{S}_n^{\text{dp}}(Y_n)) - \mathbb{E}(\mathcal{S}_n^{\text{dp}}(X)) = \mathbb{E}(\mathcal{S}^{\text{algo}}(Y_n)) - \mathbb{E}(\mathcal{S}^{\text{algo}}(X))$$

or equivalently,

$$\mathbb{E}(\mathcal{S}_n^{\text{dp}}(Y_n)) - \mathbb{E}(\mathcal{S}^{\text{algo}}(Y_n)) = \mathbb{E}(\mathcal{S}_n^{\text{dp}}(X)) - \mathbb{E}(\mathcal{S}^{\text{algo}}(X))$$

But $\mathcal{S}_n^{\text{dp}}$ is optimal for Y_n, hence

$$\mathbb{E}(\mathcal{S}_n^{\text{dp}}(Y_n)) - \mathbb{E}(\mathcal{S}^{\text{algo}}(Y_n)) \leq 0$$

Therefore,

$$\mathbb{E}(\mathcal{S}_n^{\text{dp}}(X)) - \mathbb{E}(\mathcal{S}^{\text{algo}}(X)) \leq 0$$

This concludes the proof of Lemma 2. □

Combining Lemma 1 and Lemma 2, we get:

$$\begin{aligned}\mathbb{E}(\mathcal{S}_n^{\mathrm{dp}}(X)) &\leq \mathbb{E}(\mathcal{S}^{\mathrm{algo}}(X)) \\ &\leq (1+\varepsilon)\mathbb{E}(\mathcal{S}^{\mathrm{opt}}(X))\end{aligned}$$

which concludes the proof of Theorem 3. □

3.4 Extensions

All the results presented in Sections 3.1 to 3.3, namely the cost model (Theorem 1), the optimal algorithm for discrete distributions (Theorem 2), and the approximation algorithm for continuous distributions with bounded support (Theorem 3), can be extended to some variants of the problem where the checkpoint strategy is determined a priori.

Indeed, there are two important and natural variants to consider: strategies where no reservation is checkpointed, and strategies where all reservations are checkpointed. The former variant (called No-Checkpoint) was studied in our previous work [3], where we derived an optimal algorithm for discrete distributions with reduced time complexity $O(n^2)$ instead of $O(n^3)$ as in Theorem 2. The latter variant (called All-Checkpoint) also admits an optimal dynamic programming algorithm of reduced time complexity $O(n^2)$:

Theorem 4. *For a discrete distribution $Y \sim (v_i, f_i)_{1\leq i\leq n}$, the optimal expected cost for* All-Checkpoint *(when all reservations are checkpointed) is returned by $\mathbb{E}_{\mathrm{AllCkpt}}(0)$, where $v_0 = 0$ and:*

$$\mathbb{E}_{\mathrm{AllCkpt}}(n) = \beta \cdot \mathbb{E}[Y]$$

$$\mathbb{E}_{\mathrm{AllCkpt}}(i) = \min_{i+1\leq j\leq n}\left(\mathbb{E}_{\mathrm{AllCkpt}}(j)+\beta C\cdot\sum_{k=j+1}^{n} f_k+\Big(\alpha\big(\mathbb{1}_{i\neq 0}R+(v_j-v_i)+\mathbb{1}_{j\neq n}C\big)+\beta\,\mathbb{1}_{i\neq 0}R+\gamma\Big)\cdot\sum_{k=i+1}^{n} f_k\right)$$

The optimal solution can be computed in $O(n^2)$ time.

3.5 Periodic Heuristics

In addition to the algorithms presented in Section 3, we propose a periodic heuristic for the case of bounded distributions. This strategy, described in Algorithm 2, is a natural policy, where successive reservations differ in length by a constant amount of time T, called the *period*. A checkpoint is performed at the end of each period. Hence, the value of W_i associated with each t_i is constant in this strategy. The algorithm specifies the number of chunks τ in the domain $[a, b]$ of the bounded distribution, thus the period can be computed as $T = \frac{b-a}{\tau}$.

Algorithm 2 All-Checkpoint-Periodic(X, τ)

1: Let $[a, b]$ be the domain of X, and let $T = \frac{b-a}{\tau}$
2: $(t_i, \delta_i) = \begin{cases} (a + i \cdot T, 1) & \text{for } i = 1, 2, \ldots, \tau - 1 \\ (b, 0) & \text{for } i = \tau \end{cases}$
3: **return** $\mathcal{S}_\tau^{\mathrm{period}} \leftarrow ((t_i, \delta_i))_{1\leq i\leq \tau}$

For this policy, one can derive optimal strategies for some distributions such as Uniform distributions. One can also prove that All-Checkpoint and its periodic counterpart are identical. The next subsections are dedicated to proving those assertions.

3.5.1 All-Checkpoint-Periodic for Exponential distributions

In this section, we are interested in proving that All-Checkpoint and All-Checkpoint-Periodic are similar for Exponential distribution.

Theorem 5. *When the execution time of a job follows a distribution $\mathcal{D} \sim Exponential(\lambda)$, the solution of* All-Checkpoint *and* All-Checkpoint-Periodic *are identical.*

Proof. For this specific result, we express the solutions under the form: $\{T_1, T_2, T_3, \cdots\}$.
We have the following properties:

- Given $(t_1, t_2, \dots,)$ a solution to All-Checkpoint, then the associated $\{T_1, T_2, T_3, \cdots\}$ satisfy: forall i, $t_i = \sum_{j \leq i} T_j$. This is a direct corollary of Equations (3) and (4).
- For $\mathcal{D} \sim$ Exponential (λ),

$$\mathbb{P}\left(X > V_1 + V_2\right) = \mathbb{P}\left(X > V_1\right) \cdot \mathbb{P}\left(X > V_2\right). \tag{21}$$

We define $\big(u_1, u_2, \cdots\big)$ the solution that minimizes

$$\sum_{i=1}^{\infty} \Big(\alpha\big(R + u_i + C\big) + \beta\big(R + C\big) + \gamma\Big) \cdot \mathbb{P}\left(X > \sum_{j<i} u_j\right) \tag{22}$$

To study $\mathcal{S}^{\text{opt}} = (T_1^o, T_2^o, \dots)$ the optimal solution given by All-Checkpoint for $\mathcal{D}$, we rewrite the expected cost of a solution $\mathcal{S} = (T_1, T_2, \dots)$ for All-Checkpoint:

$$\begin{aligned}
\mathbb{E}(\mathcal{S}) &= \beta \cdot \mathbb{E}[X] + \sum_{i=1}^{\infty} \Big(\alpha W_i + \beta\big(R_i + (1-\delta_{i-1})T_{i-1} + C_{i-1}\big) + \gamma\Big) \cdot \mathbb{P}\left(X > t_{i-1}\right) \\
&= \beta \cdot \mathbb{E}[X] - C + \sum_{i=1}^{\infty} \Big[\alpha\big(R + T_i + C\big) + \beta\big(R + C\big) + \gamma\Big] \cdot \mathbb{P}\left(X > \sum_{j<i} T_j\right)
\end{aligned}$$

From this formulation, we can see that $T_1^o = u_1$, $T_2^o = u_2$, $T_3^o = u_3$, … as they satisfy the same equations.

By rewriting and using Eq (21),

$$\begin{aligned}
\mathbb{E}(\mathcal{S}) &= \beta \cdot \mathbb{E}[X] + \big(\alpha W_1 + \beta R + \gamma\big) + \sum_{i=2}^{\infty} \Big(\alpha W_i + \beta\big(R + C\big) + \gamma\Big) \cdot \mathbb{P}\left(X > \sum_{j<i} T_j\right) \\
&= \beta \cdot \mathbb{E}[X] + \alpha\big(W_1 + \beta C + \gamma\big) \\
&\quad + \mathbb{P}\left(X > T_1\right) \cdot \sum_{i=2}^{\infty} \Big(\alpha\big(R + T_i + C\big) + \beta\big(R + C\big) + \gamma\Big) \cdot \mathbb{P}\left(X > \sum_{2 \leq j < i} T_j\right) \\
&= \beta \cdot \mathbb{E}[X] + \alpha\big(W_1 + \beta C + \gamma\big) \\
&\quad + \mathbb{P}\left(X > T_1\right) \cdot \sum_{i=1}^{\infty} \Big(\alpha\big(R + T_{i-1} + C\big) + \beta\big(R + C\big) + \gamma\Big) \cdot \mathbb{P}\left(X > \sum_{j<i-1} T_{j+1}\right)
\end{aligned} \tag{23}$$

From this last formulation, we can see that the sequence $(T_2^o, T_3^o, \dots)$ that minimizes Eq. (23) can be optimized independently of T_1^o, and that it is the solution that minimizes Eq. (22). Hence we obtain, $T_2^o = u_1$, $T_3^o = u_2$,... Iterating the process, we obtain the result: $T_1^o = u_1 = T_2^o = u_2 = T_3^o = \dots$, and the solution to ALL-CHECKPOINT is a periodic solution.

□

3.5.2 ALL-CHECKPOINT-PERIODIC for Uniform distributions

In this section, we consider the ALL-CHECKPOINT approach for Uniform distributions in the RESERVATIONONLY scenario. Specifically, we are able to characterize the best periodic approach: for a Uniform distribution $\mathcal{D}$ over $[a, b]$ (where $0 < a < b$), consider a reservation sequence $\mathcal{S}^{(n)} = (W_i, 1)_{1 \le i \le n}$ where $n \ge 2$, $W_1 = a + \frac{b-a}{n} + C$, $W_i = R + \frac{b-a}{n} + C$ for $1 < i < n$ and $W_n = R + \frac{b-a}{n}$. In other words, we have $n-1$ evenly distributed checkpoints in the interval $[a, b]$. The first reservation has a checkpoint but no restart, the last reservation (whenever used) has a restart but no checkpoint, and all intermediate reservations have both a restart and a checkpoint. Finally, let $\mathcal{S}^{(1)} = (b, 0)$ be the sequence with a unique reservation of length (and cost) b (no need to checkpoint in this particular case). The following proposition provides the optimal value of n:

Proposition 1. *With the above notations, $\mathbb{E}(\mathcal{S}^{(n)})$ is minimized either for $n = \max(1, \lfloor n^{\text{opt}} \rfloor)$ or $n = \lceil n^{\text{opt}} \rceil$ where $n^{\text{opt}} = \sqrt{\frac{b-a-2C}{C+R}}$ if $b - a \ge 2C$ and $n^{\text{opt}} = 1$ otherwise.*

Proof. Because the distribution is uniform, the probability that i reservations are need for a given job is always equal to $\frac{1}{n}$, hence the expected cost of $\mathcal{S}^{(n)}$ for $n \ge 2$ is

$$\mathbb{E}(\mathcal{S}^{(n)}) = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{i} W_j$$

where W_i is the cost of the i-th reservation. After several algebraic manipulations, we derive that

$$\mathbb{E}(\mathcal{S}^{(n)}) = \frac{n-1}{2n} a + \frac{n+1}{2n} b + \frac{n^2+n-2}{2n} C + \frac{n-1}{2} R$$

Differentiating, the derivative gets zeroed for $n = n^{\text{opt}}$ when $b - a \ge 2C$, and otherwise it stays positive, hence the result. □

If $\mathcal{D} \sim \text{Uniform}(a, b)$ with $[a, b] = [2, 20]$ and $C = R = 1$ we find $n^{\text{opt}} = \sqrt{8}$. One can compute that the cost for three reservations (ceil value for $\sqrt{8}$) is $\mathbb{E}(\mathcal{S}^{(3)}) = 97/6 \approx 16.2$. Let us now define a two-reservation strategies at milestones 11 and 20 [1], we can compute its associated cost by:

$$\begin{aligned}\mathbb{E}(\mathcal{S}^{(2)}) = &\int_2^{11} (\alpha 12 + \beta t + \gamma) \mathbb{P}(X = t | X \le 11) dt \\ &+ \int_{11}^{20} [(\alpha 12 + \beta 12 + \gamma) + (\alpha 10 + \beta(t-11) + \gamma)] \mathbb{P}(X = t | X \ge 11) dt\end{aligned}$$

This gives us $\mathbb{E}(\mathcal{S}^{(2)}) = 17$ for distribution $\mathcal{D}$. Hence, it is more efficient to use three reservations than two.

[1] The first reservation will be of length $11 + 1$ with the checkpointing overhead, and the second one of length $1 + 9$ with the restart cost.

Table 1: Probability distributions and parameter instantiations.

Distribution	**PDF** $f(t)$	**Instantiation**	**Support**
Distributions with infinite support			
Exponential (λ)	$\lambda e^{-\lambda t}$	$\lambda = 1.0h^{-1}$	$t \in [0, \infty)$
Weibull(λ, κ)	$\frac{\kappa}{\lambda}\left(\frac{t}{\lambda}\right)^{\kappa-1} e^{-\left(\frac{t}{\lambda}\right)^{\kappa}}$	$\lambda = 1.0h$ $\kappa = 0.5$	$t \in [0, \infty)$
Gamma(α, β)	$\frac{\beta^{\alpha}}{\Gamma(\alpha)} t^{\alpha-1} e^{-\beta t}$	$\alpha = 2.0$ $\beta = 2.0h^{-1}$	$t \in [0, \infty)$
Lognormal (ν, κ)	$\frac{1}{t\kappa\sqrt{2\pi}} e^{-\frac{(\ln t - \nu)^2}{2\kappa^2}}$	$\nu = 3.0h$ $\kappa = 0.5$	$t \in (0, \infty)$
Pareto(ν, α)	$\frac{\alpha \nu^{\alpha}}{t^{\alpha+1}}$	$\nu = 1.5h$ $\alpha = 3.0$	$t \in [\nu, \infty)$
Distributions with finite support			
Truncated Normal(ν, κ^2, a, b)	$\frac{1}{\kappa}\sqrt{\frac{2}{\pi}} \cdot \frac{e^{-\frac{1}{2}\left(\frac{t-\nu}{\kappa}\right)^2}}{1-\mathrm{erf}\left(\frac{a-\nu}{\kappa\sqrt{2}}\right)}$	$\nu = 8.0h$ $\kappa^2 = 2.0h^2$ $a = 1.0h$ $b = 20.0h$	$t \in [a, b]$
Uniform(a, b)	$\frac{1}{b-a}$	$a = 1.0h$ $b = 20.0h$	$t \in [a, b]$
Beta(α, β)	$\frac{t^{\alpha-1} \cdot (1-t)^{\beta-1}}{\mathrm{B}(\alpha, \beta)}$	$\alpha = 2.0$ $\beta = 2.0$	$t \in [0, 1]$
Bounded Pareto(L, H, α)	$\frac{\alpha L^{\alpha} t^{-\alpha-1}}{1-\left(\frac{L}{H}\right)^{\alpha}}$	$L = 1.0h$ $H = 20.0h$ $\alpha = 2.1$	$t \in [L, H]$

4 Performance evaluation

In this section, we evaluate the performance of the different algorithms in simulation. We use jobs that follow a wide range of usual probability distributions as well as a distribution obtained from traces of a real neuroscience application. The code for this section is publicly available on `https://gitlab.inria.fr/vhonore/ckpt-for-stochastic-scheduling`.

4.1 Evaluation methodology

In this section, we evaluate five different algorithms from the following two sets of strategies:
• DYN-PROG-COUNT: This set includes Algorithm 1, and its ALL-CHECKPOINT and NO-CHECKPOINT variants described in Section 3.4.
• ALL-CHECKPOINT-PERIODIC: This set includes Algorithm 2, and its NO-CHECKPOINT-PERIODIC counterpart where checkpointing is not allowed (i.e., $\delta_i = 0, \forall i$).

The above algorithms are evaluated using two scenarios and two different cost functions.

The first cost function is RESERVATIONONLY, presented in Section 2.4. This function is based on the *Reserved Instance* pricing scheme in AWS [2], where the user pays exactly what is requested. Hence, $\alpha = 1, \beta = 0, \gamma = 0$. The second cost function, called HPC, add an additional cost that is proportional to the actual execution time (pay for what you use). Thus, $\alpha = 1, \beta = 1, \gamma = 0$.

We now detail the two scenarios we use to evaluate the algorithms, using the two cost functions:

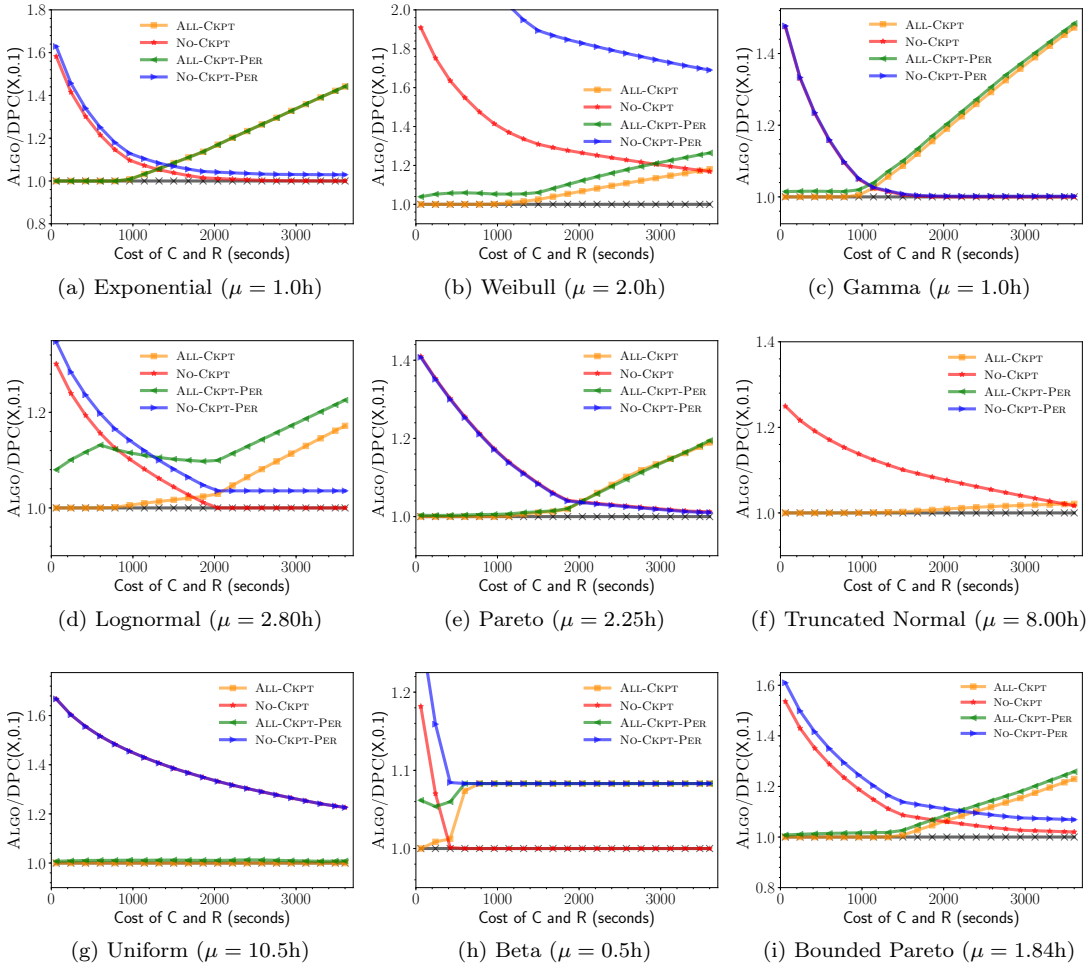
- *Scenario 1* (Section 4.2): We consider nine usual probability distributions, five of which have infinite support (Exponential, Weibull, Gamma, Lognormal, Pareto) and four have finite support (Truncated Normal, Uniform, Beta, Bounded Pareto). Table 1 lists all distributions used in simulation with the instantiations of their parameters for evaluation. The first five distributions are truncated and fed as input to Algorithm 1. To do so, we set the upper bound of the infinite support to $b = Q(1 - \upsilon)$, where $Q(x) = \inf\{t|F(t) \geq x\}$ is the quantile function and υ is a small constant. In our simulation, we set $\upsilon = 10^{-7}$. During the discretization procedure in Algorithm 1, we then normalize the probabilities of all discrete values so that they sum to 1. We set $C = R = 360$ seconds (0.1 hour). This checkpointing cost is extracted from [16] and corresponds to an average checkpointing duration, where an optimistic one is 60 seconds and a pessimistic one is 600 seconds. We further discuss the impact of the checkpointing cost on the performance.

- *Scenario 2* (Section 4.3): In this scenario, we consider the execution time traces of a real neuroscience application, and fit a Lognormal distribution to its execution times. To further evaluate the robustness of the algorithms, we perturb the parameters of the fitted distribution by varying its mean and standard deviation and show the impact on the performance.

4.2 Results for Scenario 1

We first evaluate the performance of Dyn-Prog-Count compared to the other strategies, when the values of R and C varies. Figure 5 presents the performance of these strategies normalized to that of Dyn-Prog-Count (black line for $y = 1.0$) for all distributions of Table 1 using ReservationOnly cost function. We use $\varepsilon = 0.1$ for Dyn-Prog-Count and its variants. Regarding periodic strategies, we choose the best value for the number of chunks τ in $[1, 1000]$. Not surprisingly, we can observe that when C and R are small, the best result is to use the All-Checkpoint strategy while when they are large, one should use the No-Checkpoint strategy. There exist threshold to the sizes of C and R where Dyn-Prog-Count uses a mix of checkpointed and not checkpointed reservations. In that case the gain can be up to 10% compared to the best strategy. An interesting future direction is to find properties on those threshold depending on the distribution. Finally, one should observe that the gain obtained with Dyn-Prog-Count compared to the best periodic solutions is in general more important (for Truncated Normal, the performance of periodic solutions are worse than a factor 2 of Dyn-Prog-Count). For Exponential distribution, All-Checkpoint and its periodic counterpart are identical (proof can be found in Section 3.5.1), due to the memoryless property of the exponential distribution. Figure 6 shows the results for a similar setup with HPC cost function. We see that results are consistent between the two cost functions, with small variations in results but same general trends.

We then study the impact of ε on the performance of Dyn-Prog-Count (DPC) for the two different cost functions, when $R = C =$ 6min, 30min and 60min. The idea is that when $\varepsilon = 1$, this theoretically guarantees that the performance is at most twice $(= 1 + \varepsilon)$ that of the optimal, but in practice it can be a lot better. We study in Figure 7 the performance of Dyn-Prog-Count for various values of ε for distributions of Table 1 with ReservationOnly cost function. All performance are normalized by Dyn-Prog-Count for $\varepsilon = 0.1$. We can see that in practice, the convergence to the lower bound in performance is fast. Indeed, for $\varepsilon = 1$ and $C = R =$ 6min (Figure 7a), almost all distributions already reach convergence, except for Weibull and Pareto (which have a much larger definition domain of definition). For those distributions, we see that they converged for $\varepsilon = 0.1$. We observe similar trends in Figures 7b and 7c when R and C increases. Interestingly, one can note that Pareto distribution converges

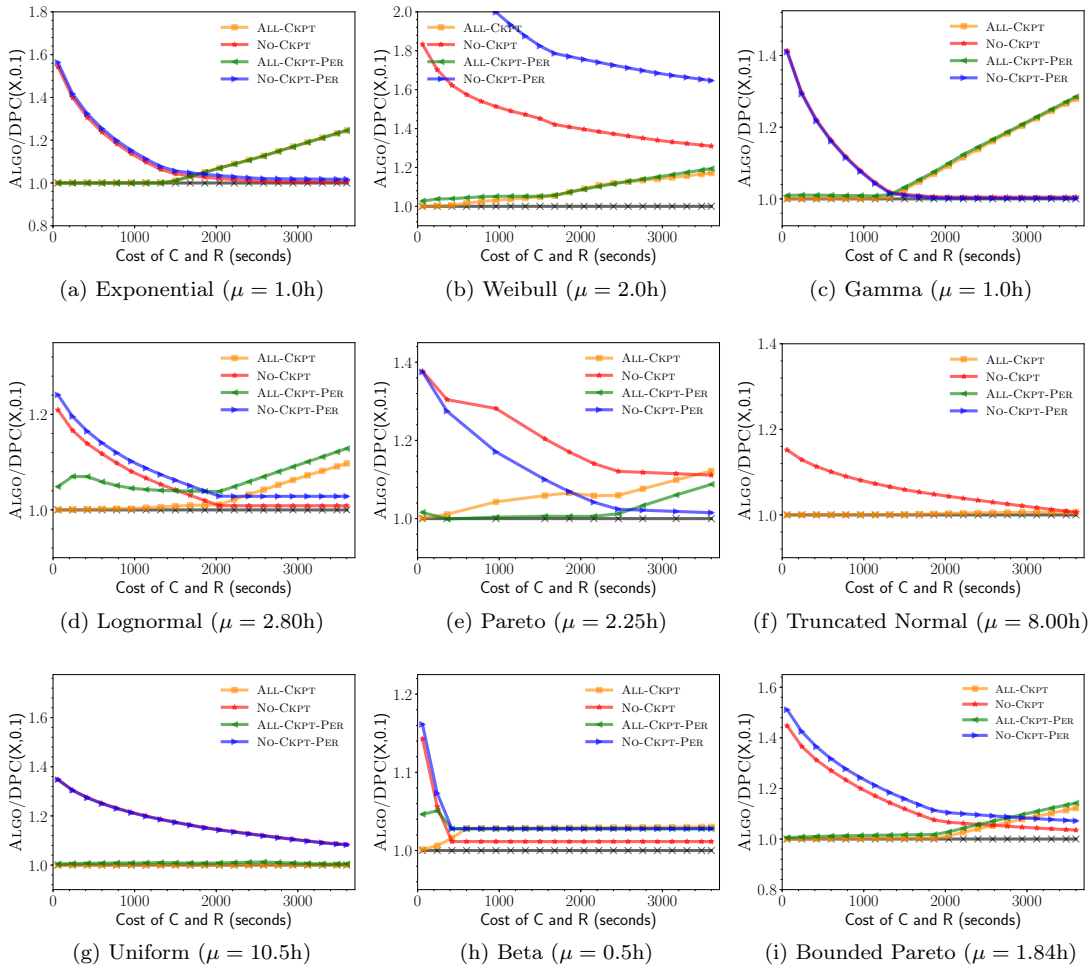


(a) Exponential ($\mu = 1.0$h) (b) Weibull ($\mu = 2.0$h) (c) Gamma ($\mu = 1.0$h)

(d) Lognormal ($\mu = 2.80$h) (e) Pareto ($\mu = 2.25$h) (f) Truncated Normal ($\mu = 8.00$h)

(g) Uniform ($\mu = 10.5$h) (h) Beta ($\mu = 0.5$h) (i) Bounded Pareto ($\mu = 1.84$h)

Figure 5: Expected costs of the different strategies normalized to that of Dyn-Prog-Count$(X, 0.1)$ when $C = R$ vary from 60 to 3600 seconds, for all distributions in Table 1 with support considered in hours with ReservationOnly cost function. We indicate in brackets the mean μ of each distribution.

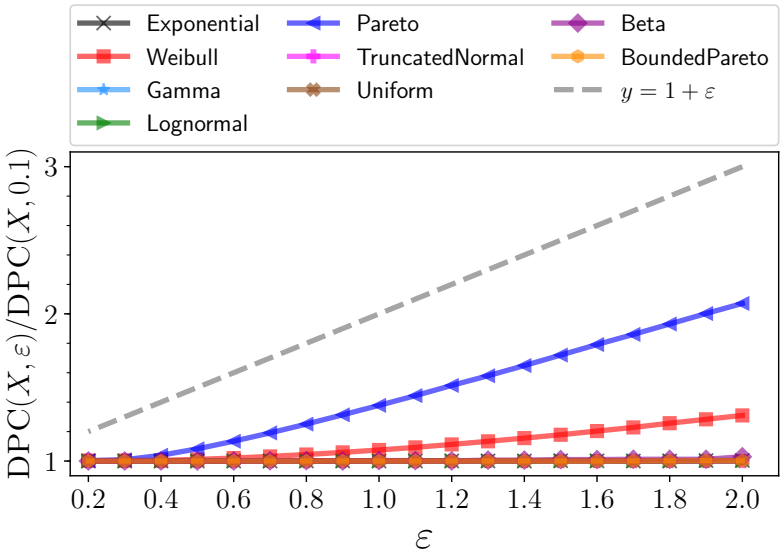


(a) Exponential ($\mu = 1.0$h) (b) Weibull ($\mu = 2.0$h) (c) Gamma ($\mu = 1.0$h)

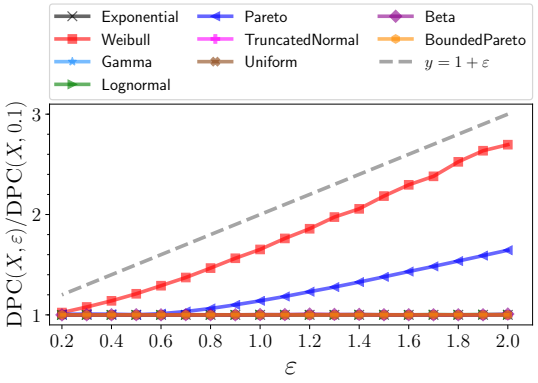
(d) Lognormal ($\mu = 2.80$h) (e) Pareto ($\mu = 2.25$h) (f) Truncated Normal ($\mu = 8.00$h)

(g) Uniform ($\mu = 10.5$h) (h) Beta ($\mu = 0.5$h) (i) Bounded Pareto ($\mu = 1.84$h)

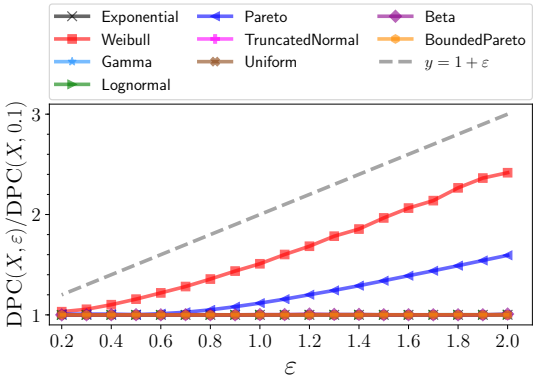
Figure 6: Expected costs of the different strategies normalized to that of Dyn-Prog-Count$(X, 0.1)$ when $C = R$ vary from 60 to 3600 seconds, for all distributions in Table 1 with support considered in hours with HPC cost function. We indicate in brackets the mean μ of each distribution.



(a) $C = R = 6$min

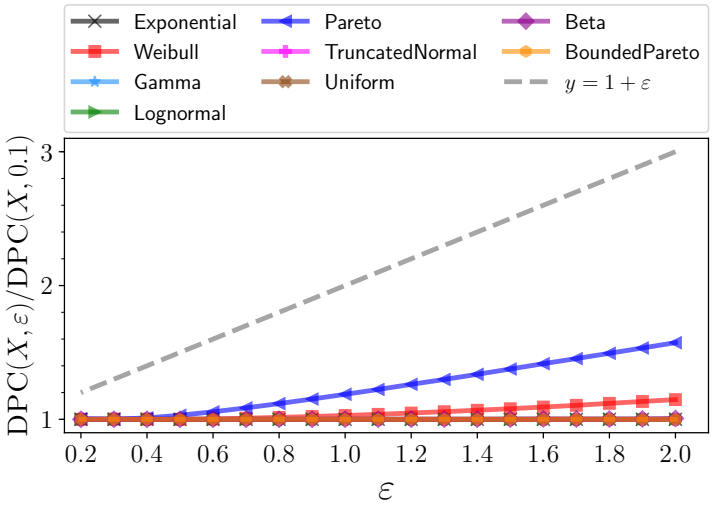


(b) $C = R = 30$min

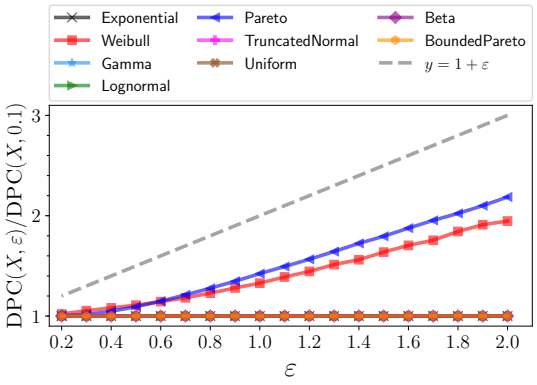


(c) $C = R = 60$min

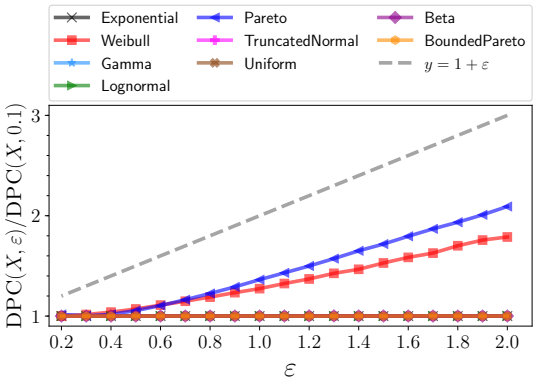
Figure 7: Expected cost of DYN-PROG-COUNT(X, ε) as a function of ε for different distributions for X with RESERVATIONONLY cost function. $C = R$ are set to $6, 30$ and 60min.



(a) $C = R = 6$min



(b) $C = R = 30$min



(c) $C = R = 60$min

Figure 8: Expected cost of Dyn-Prog-Count(X, ε) as a function of ε for different distributions for X with HPC cost function. $C = R$ are set to $6, 30$ and 60min.

faster for $C = R = 30$ or 60min than for $C = R = 6$min, while Weibull distribution shows contrary behavior. Convergence is still achieved for $\varepsilon = 0.1$. This shows the possible impact of application features on algorithm behavior.

Figure 8 shows the results of a similar setup for HPC cost function. One can note that all distributions converge even faster than for ReservationOnly cost function. Indeed, for $\varepsilon = 0.4$, all distribution converge. Overall, the results are consistent in between the two cost functions.

Our final evaluation for this scenario is a study of the impact of the size of the period. Until now we have always chosen the period that minimized the objective functions. Table 2 shows the performance of both variants of the periodic algorithms, All-Checkpoint-Periodic and No-Checkpoint-Periodic, normalized by that of Dyn-Prog-Count ($\varepsilon = 0.1$), when $C = R = 360$s using ReservationOnly cost function. For each distribution: the second columns shows the best period found when τ varies from 1 to 1000 (with its associated cost normalized by that of Dyn-Prog-Count), and the other columns present results for specific values of τ in that interval. As was observed before, All-Checkpoint-Periodic is in general not able to match Dyn-Prog-Count (except for some distributions). We can also clearly see that No-Checkpoint-Periodic performs even worse than All-Checkpoint-Periodic. The reason is that the checkpointing cost is relatively low in this setup, so it is preferable to checkpoint more often than never. Hence, when $C = R = 1$h, Table 3 shows that No-Checkpoint-Periodic performs slightly better than All-Checkpoint-Periodic. Finally another observation is that a wrong period size can significantly deteriorate the performance of the periodic algorithms. Tables 4 and 5 show similar trends using the HPC cost function.

Table 2: Performance of All-Checkpoint-Periodic and No-Checkpoint-Periodic, normalized by Dyn-Prog-Count$(X, 0.1)$ for $C = R = 360$s, with ReservationOnly cost function.

Distribution	All-Checkpoint-Periodic							No-Checkpoint-Periodic						
	Best τ	$\tau = 1$	$\tau = 200$	$\tau = 400$	$\tau = 600$	$\tau = 800$	$\tau = 1000$	Best τ	$\tau = 1$	$\tau = 200$	$\tau = 400$	$\tau = 600$	$\tau = 800$	$\tau = 1000$
Exponential	27 (1.00)	9.82	2.31	4.02	5.75	7.47	9.19	14 (1.38)	9.82	6.91	13.23	19.56	25.89	32.22
Weibull	380 (1.06)	106.50	1.15	1.06	1.10	1.18	1.27	89 (2.54)	106.50	3.26	5.32	7.52	9.74	11.98
Gamma	14 (1.02)	6.02	3.68	6.76	9.85	12.93	16.02	8 (1.26)	6.02	9.34	18.08	26.82	35.56	44.31
Lognormal	11 (1.11)	3.60	3.92	7.05	10.18	13.32	16.45	4 (1.25)	3.60	15.48	30.17	44.86	59.56	74.25
Pareto	1000 (1.01)	228.77	1.75	1.23	1.09	1.03	1.01	562 (1.33)	228.77	1.78	1.37	1.33	1.37	1.45
TruncatedNormal	2 (2.15)	2.18	8.17	14.31	20.45	26.59	32.73	1 (2.18)	2.18	197.54	394.01	590.47	786.93	983.39
Uniform	8 (1.01)	1.57	3.17	5.51	7.86	10.20	12.54	1 (1.57)	1.57	51.08	101.33	151.59	201.84	252.10
Beta	2 (1.06)	1.11	30.78	61.00	91.23	121.45	151.67	1 (1.11)	1.11	40.85	81.15	121.45	161.75	202.05
BoundedPareto	32 (1.01)	7.53	1.73	2.71	3.70	4.70	5.69	14 (1.44)	7.53	6.51	12.28	18.06	23.83	29.61

Table 3: Performance of All-Checkpoint-Periodic and No-Checkpoint-Periodic, normalized by Dyn-Prog-Count$(X, 0.1)$ for $C = R = 1$h, with ReservationOnly cost function.

Distribution	All-Checkpoint-Periodic							No-Checkpoint-Periodic						
	Best τ	$\tau = 1$	$\tau = 200$	$\tau = 400$	$\tau = 600$	$\tau = 800$	$\tau = 1000$	Best τ	$\tau = 1$	$\tau = 200$	$\tau = 400$	$\tau = 600$	$\tau = 800$	$\tau = 1000$
Exponential	12 (1.44)	7.35	9.49	18.53	27.57	36.61	45.66	14 (1.03)	7.35	5.17	9.90	14.64	19.38	24.12
Weibull	156 (1.26)	70.94	1.29	1.65	2.11	2.60	3.11	89 (1.69)	70.94	2.17	3.54	5.01	6.49	7.98
Gamma	7 (1.48)	4.77	17.59	34.70	51.82	68.93	86.05	8 (1.00)	4.77	7.40	14.32	21.25	28.18	35.11
Lognormal	4 (1.23)	2.98	18.81	36.96	55.11	73.26	91.42	4 (1.04)	2.98	12.82	24.98	37.14	49.30	61.47
Pareto	563 (1.20)	175.44	1.58	1.24	1.21	1.24	1.31	562 (1.02)	175.44	1.37	1.05	1.02	1.05	1.11
TruncatedNormal	1 (1.85)	1.85	38.01	74.45	110.89	147.33	183.78	1 (1.85)	1.85	167.50	334.08	500.67	667.25	833.83
Uniform	3 (1.01)	1.23	13.46	26.26	39.07	51.88	64.69	1 (1.23)	1.23	39.89	79.13	118.37	157.61	196.85
Beta	1 (1.08)	1.08	207.32	414.09	620.87	827.64	1034.42	1 (1.08)	1.08	39.93	79.31	118.70	158.08	197.47
BoundedPareto	14 (1.26)	5.59	5.72	10.88	16.05	21.21	26.38	14 (1.07)	5.59	4.82	9.11	13.39	17.68	21.96

4.3 Results for Scenario 2

We now present the simulation results for a probability distribution fitted to the execution time traces of a real neuroscience application (*a code for structural identification of orbital anatomy*) extracted from the Vanderbilt's medical imaging database [14]. Figure 1 shows the execution time

Table 4: Performance of All-Checkpoint-Periodic and No-Checkpoint-Periodic, normalized by Dyn-Prog-Count$(X, 0.1)$ for $C = R = 360$s, with HPC cost function.

Distribution	All-Checkpoint-Periodic							No-Checkpoint-Periodic						
	Best τ	$\tau = 1$	$\tau = 200$	$\tau = 400$	$\tau = 600$	$\tau = 800$	$\tau = 1000$	Best τ	$\tau = 1$	$\tau = 200$	$\tau = 400$	$\tau = 600$	$\tau = 800$	$\tau = 1000$
Exponential	21 (1.00)	6.38	2.43	4.20	5.97	7.74	9.52	10 (1.35)	6.38	8.14	15.59	23.04	30.50	37.95
Weibull	300 (1.04)	62.32	1.07	1.05	1.12	1.21	1.31	64 (2.29)	62.32	3.53	6.03	8.60	11.19	13.79
Gamma	12 (1.01)	4.03	3.83	6.98	10.13	13.28	16.44	6 (1.24)	4.03	10.80	20.89	30.99	41.08	51.18
Lognormal	4 (1.07)	2.42	3.74	6.63	9.52	12.41	15.30	3 (1.17)	2.42	16.55	32.24	47.93	63.62	79.32
Pareto	999 (1.00)	128.29	1.39	1.11	1.03	1.01	1.00	415 (1.28)	128.29	1.45	1.28	1.32	1.41	1.51
TruncatedNormal	2 (1.61)	1.61	6.93	12.34	17.75	23.16	28.57	1 (1.61)	1.61	207.16	413.80	620.45	827.09	1033.73
Uniform	6 (1.01)	1.28	2.98	5.07	7.17	9.26	11.36	1 (1.28)	1.28	54.39	107.91	161.42	214.93	268.44
Beta	1 (1.03)	1.03	32.95	65.29	97.62	129.96	162.29	1 (1.03)	1.03	45.95	91.22	136.49	181.76	227.03
BoundedPareto	26 (1.01)	4.69	1.74	2.68	3.62	4.57	5.52	10 (1.38)	4.69	7.22	13.62	20.03	26.44	32.85

Table 5: Performance of All-Checkpoint-Periodic and No-Checkpoint-Periodic, normalized by Dyn-Prog-Count$(X, 0.1)$ for $C = R = 1$h, with HPC cost function.

Distribution	All-Checkpoint-Periodic							No-Checkpoint-Periodic						
	Best τ	$\tau = 1$	$\tau = 200$	$\tau = 400$	$\tau = 600$	$\tau = 800$	$\tau = 1000$	Best τ	$\tau = 1$	$\tau = 200$	$\tau = 400$	$\tau = 600$	$\tau = 800$	$\tau = 1000$
Exponential	9 (1.25)	4.81	11.24	22.19	33.15	44.11	55.07	10 (1.02)	4.81	6.13	11.74	17.35	22.96	28.58
Weibull	116 (1.19)	44.78	1.30	1.83	2.42	3.03	3.65	64 (1.65)	44.78	2.54	4.33	6.18	8.04	9.91
Gamma	6 (1.29)	3.26	21.21	42.11	63.02	83.93	104.84	6 (1.00)	3.26	8.73	16.89	25.04	33.20	41.36
Lognormal	4 (1.13)	2.12	21.39	42.15	62.91	83.67	104.43	3 (1.03)	2.12	14.50	28.24	41.99	55.73	69.48
Pareto	438 (1.11)	103.29	1.27	1.11	1.13	1.21	1.31	415 (1.03)	103.29	1.17	1.03	1.06	1.13	1.21
TruncatedNormal	1 (1.45)	1.45	41.13	81.05	120.98	160.91	200.83	1 (1.45)	1.45	186.46	372.46	558.46	744.46	930.46
Uniform	2 (1.00)	1.08	15.21	29.71	44.21	58.71	73.22	1 (1.08)	1.08	45.90	91.05	136.21	181.36	226.52
Beta	1 (1.03)	1.03	263.43	526.83	790.22	1053.62	1317.02	1 (1.03)	1.03	45.65	90.62	135.59	180.56	225.53
BoundedPareto	11 (1.14)	3.64	6.50	12.51	18.53	24.55	30.57	10 (1.07)	3.64	5.59	10.56	15.53	20.50	25.46

traces of the application and its fitted Lognormal distribution. Figure 9 presents the performance of different algorithms for this fitted distribution using the ReservationOnly cost function, for different values of $C = R$. To evaluate the robustness of algorithms, we also vary the original mean μ_o (Figures 9a 9c 9e) or standard deviation σ_o (Figures 9b 9d 9f) of the distribution from their original values. For readability, all axis are in logscale. We fix $\varepsilon = 1.0$ and test checkpoint/restart costs equals to 6min, 1h and 12h. For periodic strategies, we use similar brute-force procedure as Scenario 1 to find the period that performs best. The expected costs of the algorithms are normalized by that of an *omniscient* scheduler (blue dashed line), which knows the execution time t of a job *a priori*, and thus would pay the minimum possible cost by making a single reservation of length $t_1 = t$. Averaging over all possible values of t from the distribution, the omniscient scheduler has an expected cost

$$\mathbb{E}^o = \int_0^\infty (\alpha t + \beta t + \gamma) f(t) dt = \alpha \cdot \mathbb{E}[X] + \gamma$$

We can observe that Dyn-Prog-Count always gives the best performance. As previously observed, the checkpointing cost influences the performance of No-Checkpoint and All-Checkpoint with regard to Dyn-Prog-Count. When $C = R = 600$ seconds or $C = R = 3600$s (Figure 9a - 9d), the value is low enough to allow for checkpointing all reservations, the performance of Dyn-Prog-Count and All-Checkpoint are the same and outperforms No-Checkpoint by a wide margin. However, when the checkpoint/restart overhead increases to 12h (roughly $\frac{\mu_o}{2}$), we see that checkpointing all reservations become overcostly (No-Checkpoint is better than All-Checkpoint). In that case, Dyn-Prog-Count outperforms all other strategies. One can observe that when the ratio μ/σ is large (either by increasing the mean, or decreasing the standard deviation), the solutions converge to the omniscient scheduler. This could be expected, in this case the variability becomes negligible and the job behaves similarly to a deterministic job. As for the periodic algorithms, All-Checkpoint-Periodic has better performance than No-Checkpoint-Periodic. However, both algorithms have worst performance than Dyn-Prog-Count. The results demonstrate the robustness of Dyn-Prog-Count for a practical application with different distribution parameters.

Figure 10 shows similar trends using the same setup with the HPC cost function.

5 Experiments

In this section, we conduct real experiments on an HPC platform by using three stochastic neuroscience applications. The focus is to study the performance of different reservation and checkpointing strategies for scheduling multiple jobs in a shared HPC execution environment.

5.1 Experimental setup

The chosen neuroscience applications are described in Table 6 along with their execution characteristics, which are extracted from the Vanderbilt's medical imaging database [14]. In particular, the walltime distributions are obtained by fitting the execution time traces, and the checkpointing/restart costs are obtained by analyzing and averaging their memory footprints. Note that, for these applications, the restart costs (R) are different from the checkpointing costs (C).

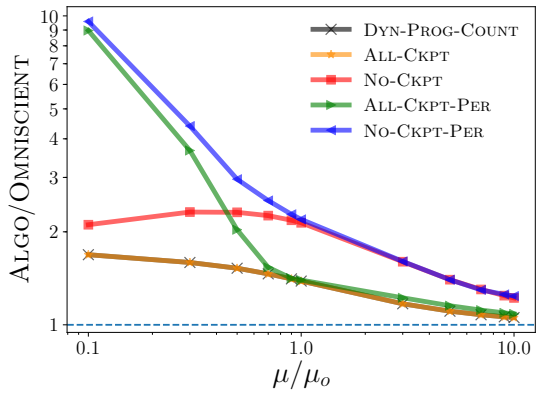
Here, we focus on the evaluation of the following two different sets of strategies:

- An HPC-for-neuroscience strategy (called HPC in Section 5.2), which uses the average of the last 5 runs as the initial reservation length and then increases it by a factor of 1.5 for each subsequent reservation. This strategy is currently used by the MASI group [20] at Vanderbilt to handle stochastic neuroscience applications.
- Our proposed Dyn-Prog-Count strategy and its All-Checkpoint variant.

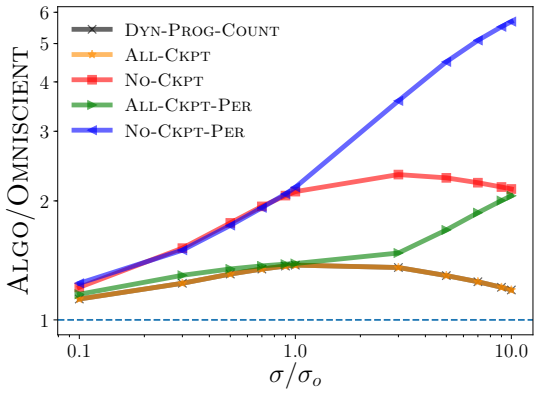
We ran the experiments on a 256-thread Intel Processor (Xeon Phi 7230, 1.30GHz) while submitting jobs through the Slurm scheduler [32]. All three neuroscience applications are sequential (i.e., uses a single hardware thread) and performs some medical imaging analysis. The variation in the execution time is due to the different characteristics of the input data. However, as we do not have access to the raw input images, we used the information in the logs to simulate the characteristics of the input data, thereby forcing a job to run for a certain walltime and saving a specific amount of data for the checkpoints. In each experiment, we submitted 500 jobs from one of the three applications, and recorded the completion time of each job. We use the average job *stretch* (defined as the ratio between the total execution time of a job and its actual walltime) to show the individual job performance, and use the *utilization* (defined as the ratio between the sum of all jobs' walltimes and the total time required to execute them) to show the performance of the system for the whole job set. During the experiments, the scheduler has complete access to the entire platform, thus corresponding to the scenario with $\alpha = 1$, $\beta = 0$, $\gamma = 0$.

5.2 Experimental results

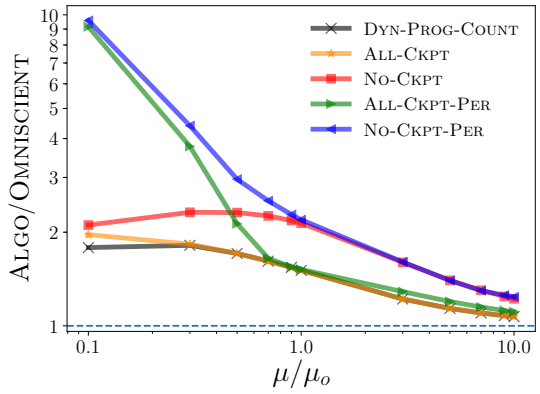
By experimenting on a real system, we investigate the robustness of our strategy: 1) when multiple applications are running concurrently; 2) when the read/write times vary due to congestion while accessing I/O and/or due to application interference; 3) when the C/R costs vary depending on when in the application the checkpoint/restart takes place (i.e., different values for different reservations). Figure 11 shows the performance of the three strategies when submitting 500 jobs from each application to the Slurm scheduler. We manually force the C/R costs to be the same (as in Table 6) for each strategy so as to study the effects of application interference and the runtime system's performance variability on our model. The findings are consistent with the simulation results (in Section 4), showing that Dyn-Prog-Count performs better than its



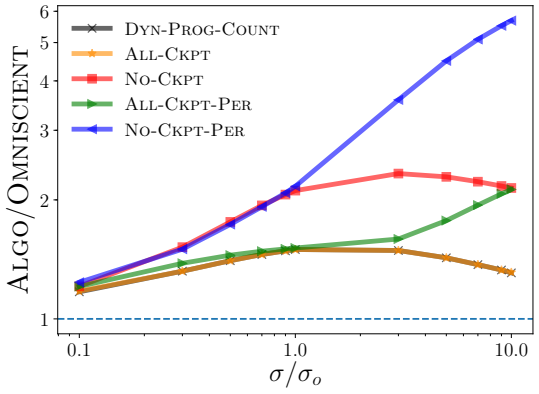
(a) Variation of μ, $\sigma = \sigma_o = 19.7$h, $C = R = 600$s



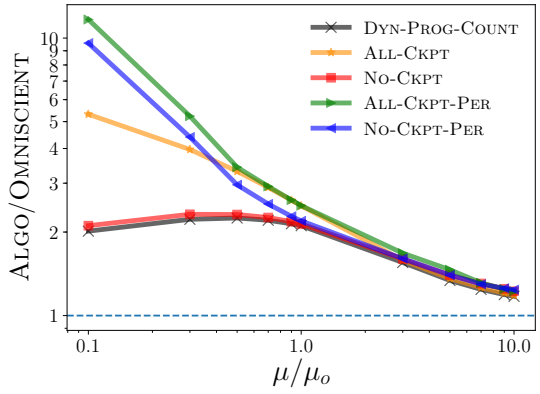
(b) Variation of σ, $\mu = \mu_o = 21.4$h, $C = R = 600$s



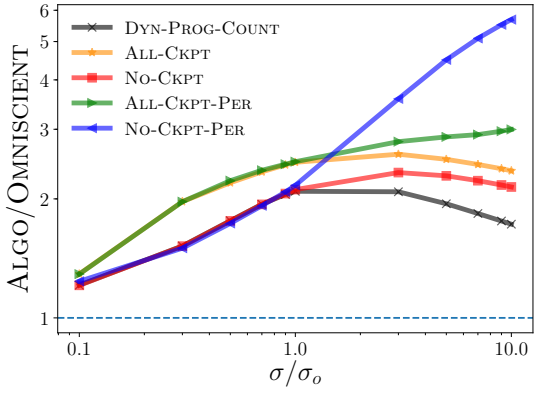
(c) Variation of μ, $\sigma = \sigma_o = 19.7$h, $C = R = 1$h



(d) Variation of σ, $\mu = \mu_o = 21.4$h, $C = R = 1$h

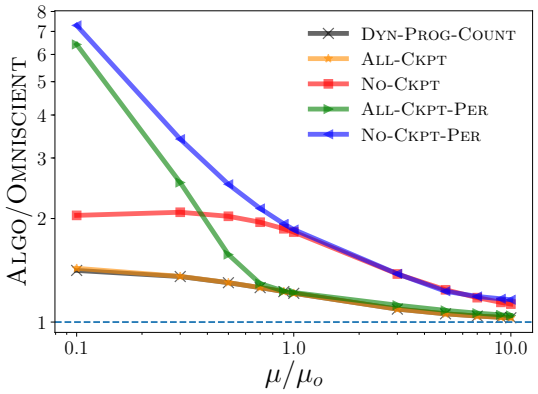


(e) Variation of μ, $\sigma = \sigma_o = 19.7$h, $C = R = 12$h

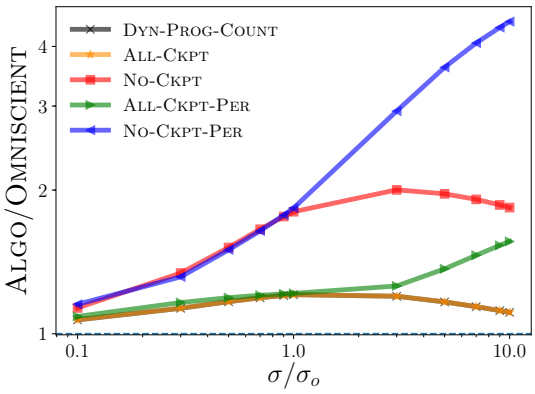


(f) Variation of σ, $\mu = \mu_o = 21.4$h, $C = R = 12$h

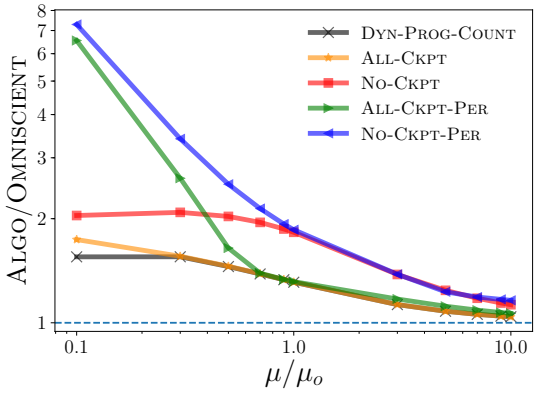
Figure 9: Normalized performance of algorithms with omniscient scheduler when μ or σ vary, using ReservationOnly cost function ($\alpha = 1.0$, $\beta = \gamma = 0$). Basis is the Lognormal distribution in Figure 1 ($\mu_o = 21.4$h, $\sigma_0 = 19.7$h). $C = R$ are set to $600, 3600$ and 43200s (12h), $\varepsilon = 1$.



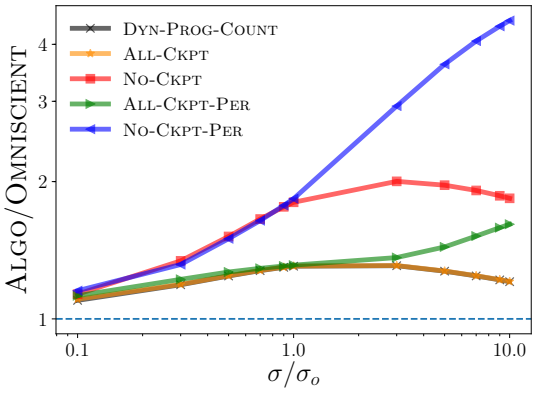
(a) Variation of μ, $\sigma = \sigma_o = 19.7$h, $C = R = 600$s



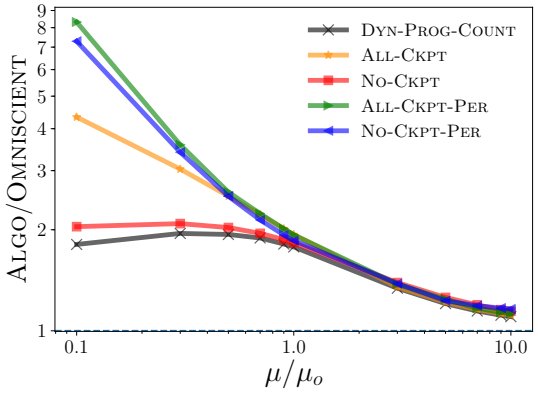
(b) Variation of σ, $\mu = \mu_o = 21.4$h, $C = R = 600$s



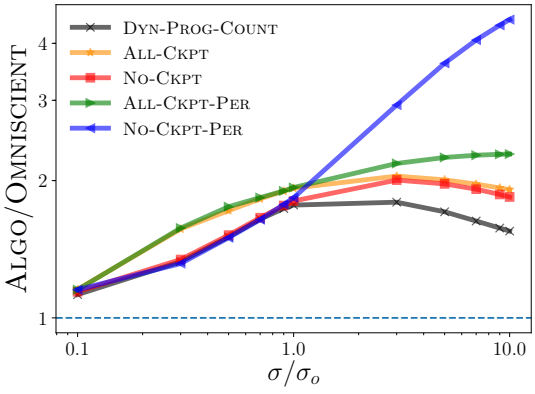
(c) Variation of μ, $\sigma = \sigma_o = 19.7$h, $C = R = 1$h



(d) Variation of σ, $\mu = \mu_o = 21.4$h, $C = R = 1$h



(e) Variation of μ, $\sigma = \sigma_o = 19.7$h, $C = R = 12$h



(f) Variation of σ, $\mu = \mu_o = 21.4$h, $C = R = 12$h

Figure 10: Normalized performance of algorithms with omniscient scheduler when μ or σ vary, using HPC cost function ($\alpha = \beta = 1.0$, $\gamma = 0$). Basis is the Lognormal distribution in Figure 1 ($\mu_o = 21.4$h, $\sigma_0 = 19.7$h). $C = R$ are set to $600, 3600$ and 43200s (12h), $\varepsilon = 1$.

Table 6: Characteristics of the chosen neuroscience applications.

Application Type	**Walltime distribution**	C	R
Diffusion model fitting (Qball)	Gamma ($k = 1.18$, $\theta = 34$, $[a, b] = [146s, 407s]$)	90s	40s
Diffusion model fitting (SD)	Weibull ($k = 1043811$, $\lambda = 1174322466$, $[a, b] = [46min, 2.3h]$)	25min	10min
Functional connectivity analysis (FCA)	Gamma ($k = 3.6$, $\theta = 72$, $[a, b] = [165s, 1003s]$)	150s	100s

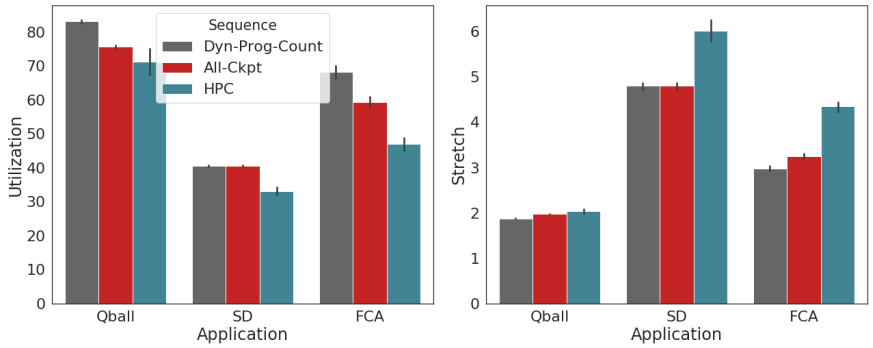


Figure 11: Utilization and average job stretch for Dyn-Prog-Count, All-Checkpoint and the HPC strategies.

All-Checkpoint variant in terms of both system utilization and average job stretch using all three applications. Moreover, the two algorithms outperform the simple HPC strategy.

Figure 12 shows the results when the C/R costs could vary for different reservations. Based on the log traces of these three applications, we noticed that their memory footprints can vary by as much as 30% depending on when the checkpoint is taken (e.g., the checkpoint time can vary between 80 and 110 seconds for Qball). While the Dyn-Prog-Count solution is computed using the average C/R costs presented in Table 6, the experimental results show that its performance is robust up to 15-20% variability in the C/R costs. Moreover, the average job stretch appears to be even more stable than the utilization, suggesting that most of the submitted jobs are not impacted by the fluctuation in the C/R costs.

Finally, we conduct experiments in a more realistic scenario by running all three applications at the same time and investigating the impact on the strategies. In particular, we submitted a total of 500 jobs (100 from Qball, and 200 each from SD and FCA), and kept the C/R costs constant across different reservations. We recorded the utilization and average job stretch when using Dyn-Prog-Count compared to the HPC strategy for 10 different runs choosing different instances from the traces each time. The results are presented in Table 7. We can see that Dyn-Prog-Count improves both utilization and average job stretch by 10% on average, and by up to 20% depending on the instances submitted. Overall, these results again illustrate the robustness of our algorithm and confirm its benefit for scheduling stochastic applications on reservation-based platforms.

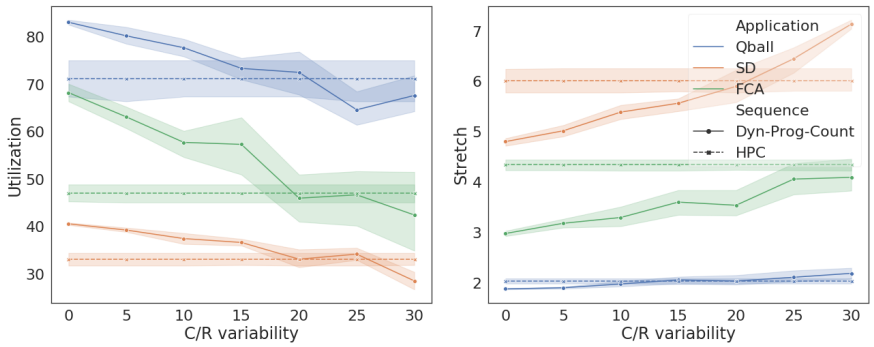


Figure 12: Utilization and average job stretch for the three applications (blue: Qball; Orange: SD; Green: FCA) when varying the C/R costs by different percentages (0 to 30%) using the Dyn-Prog-Count strategy. Horizontal lines represent the results for the HPC strategy.

Table 7: Utilization and average job stretch for 10 different runs, each using 500 jobs from all three applications. The runs are ordered by the best improvement of Dyn-Prog-Count in utilization.

Dyn-Prog-Count		**HPC**		**Improvement**	
Utilization	**Avg Stretch**	**Utilization**	**Avg Stretch**	**Utilization**	**Avg Stretch**
67	2.04	55	2.34	21%	15%
73	1.72	62	2.04	18%	19%
62	2.08	55	2.46	12%	18%
71	1.88	64	2.1	11%	12%
63	2.19	56	2.41	11%	10%
71	1.74	64	1.96	10%	12%
75	1.51	68	1.69	10%	12%
68	2.09	65	2.19	4%	5%
61	2.24	60	2.32	2%	4%
77	1.96	75	1.99	2%	2%

6 Related Work

We review some related work on reservation-based scheduling and checkpointing in HPC and cloud systems, as well as some prior work on dealing with stochastic applications.

6.1 Reservation-based scheduling

Batch schedulers are widely adopted by many resource managers in HPC systems, such as Slurm [32], Torque [27] and Moab [5]. Most batch schedulers use resource reservation in combination with backfilling [21,23,26], and rely on users to provide accurate estimates for the walltimes of the submitted jobs. While this works for applications with deterministic resource needs, it can cause resource over-estimation or under-estimation for stochastic jobs with large variations in the walltime, thus degrading system and/or application performance [11,29].

Clusters of commodity servers that use big-data frameworks such as MapReduce [8] and Dryad [18] offer alternative solutions to running HPC workloads. Schedulers for these frameworks such as YARN [28] and Mesos [15] offer distinct features (e.g., fairness, resource negotiation)

to manage the workloads, but they generally also require accurate information regarding the applications' resource demands.

Cloud computing platforms such as Amazon AWS [2] and Google GCP [13] have emerged as another option for executing HPC applications, with a variety of pricing and reservation schemes. Both on-demand and reservation models are available with the latter typically offering a lower price. Several works [1,6,9,31] have studied the pricing strategies for platform providers, as well as delay modeling and cost evaluation for the users.

6.2 Stochastic scheduling and checkpointing

Many prior works have considered stochastic scheduling for jobs with execution time uncertainty. Most research in this paradigm (e.g., [4,12,19,22,24,25]) assumes that the execution time of a job follows a known probability distribution and aims at optimizing the expected response time or makespan for a set of jobs under various distributions. Most of them, however, do not consider the problem in the context of reservation-based scheduling. In our prior work [3], we have proposed near-optimal reservation strategies for a single job in both HPC and cloud systems. The work was later extended to scheduling a set of stochastic jobs, both sequential and parallel, using backfilling in a reservation-based scheduling environment [10, 11].

Another approach to coping with stochastic applications and/or platform unavailability is through checkpoint-restart [17, 30]. To ensure the robustness of the execution, the application's state is periodically checkpointed and in case of interrupt (due to either insufficient reservation or platform failure), the application can be recovered from the last saved checkpoint. In the context of fault tolerance, a lot of work (e.g., [7,17,33]) has been devoted to deriving the optimal checkpointing interval that minimizes the checkpointing overhead or resource waste.

In this paper, we present strategies that combine reservation and checkpointing for stochastic jobs with known execution time distributions. To the best of our knowledge, this is the first result to provide performance guarantee on the expected execution time while leveraging checkpointing in reservation-based scheduling environment.

7 Conclusion and Future Work

In this paper, we have studied the problem of performing checkpointing for stochastic applications running on a reservation-based platform. We presented an optimization framework through a scheduling problem that combines a sequence of reservations and associated checkpointing decisions, for a considered job. We provided an optimal solution via a dynamic programming algorithm in the case of discrete distributions, and showed the existence of a fully polynomial-time approximation scheme for bounded continuous distributions. Through an extensive set of simulations and experiments, we have demonstrated the effectiveness of those solutions in comparison with classic strategies using both usual distributions and traces from real neuroscience applications.

For future work, we are interested in quantifying the critical checkpointing cost, below (or above) which the best strategy is to always (or never) checkpoint the reservations. This will help completely characterize the optimal solution for a given application. Incorporating stochasticity of checkpointing costs in the optimization is another interesting direction.

Through this work and our previous efforts [3, 10, 11] on scheduling stochastic applications, we would like to motivate users and systems administrators of HPC platforms to adopt our strategies in the case of stochastic job submissions. In particular, by modeling the execution time of a stochastic job with a distribution, we have demonstrated the benefits of requesting a sequence of reservations compared to a single reservation of the maximum execution time for the

job. Thus, by leveraging such knowledge on the job profiles, we believe that our strategies will lead to significant improvements in terms of both system and application performance over the existing scheduling policies used in HPC platforms.

Acknowledgments Some of the simulations presented in this paper were carried out using the PlaFRIM experimental testbed, supported by Inria, CNRS (LABRI and IMB), Université de Bordeaux, Bordeaux INP and Conseil Régional d'Aquitaine (see `https://www.plafrim.fr/en/home/`). The remaining simulation resources were provided by the computing facilities MCIA (Mésocentre de Calcul Intensif Aquitain) of the Université de Bordeaux and of the Université de Pau et des Pays de l'Adour.

References

[1] M. Afanasyev and H. Mendelson. Service provider competition: Delay cost structure, segmentation, and cost advantage. *Manufacturing & Service Operations Management*, 12(2):213–235, 2010.

[2] Amazon. AWS pricing information. `https://aws.amazon.com/ec2/pricing/`. Accessed: 2018-10-11.

[3] G. Aupy, A. Gainaru, V. Honoré, P. Raghavan, Y. Robert, and H. Sun. Reservation Strategies for Stochastic Jobs. In *IPDPS*, 2019.

[4] J. Bruno, P. Downey, and G. N. Frederickson. Sequencing tasks with exponential service times to minimize the expected flow time or makespan. *Journal of the ACM*, 28(1):100–113, 1981.

[5] N. Capit, G. D. Costa, Y. Georgiou, G. Huard, C. Martin, G. Mounie, P. Neyron, and O. Richard. A batch scheduler with high level components. In *CCGrid*, pages 776–783, 2005.

[6] S. Chen, H. Lee, and K. Moinzadeh. Pricing schemes in cloud computing: Utilization-based versus reservation-based. *Production and Operations Management*, 2017.

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

[8] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, Jan. 2008.

[9] L. Dierks and S. Seuken. Cloud pricing: the spot market strikes back. In *The Workshop on Economics of Cloud Computing*, 2016.

[10] A. Gainaru, G. Pallez, H. Sun, and P. Raghavan. Speculative scheduling for stochastic HPC applications. In *ICPP*, 2019.

[11] A. Gainaru, H. Sun, G. Aupy, Y. Huo, B. A. Landman, and P. Raghavan. On-the-fly scheduling versus reservation-based scheduling for unpredictable workflows. *Int. J. High Perf. Computing Applications*, 2019.

[12] A. Goel and P. Indyk. Stochastic load balancing and related problems. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, FOCS '99, pages 579–586, 1999.

[13] Google. GCP pricing information. `https://cloud.google.com/pricing/`. Accessed: 2018-10-16.

[14] R. L. Harrigan, B. C. Yvernault, B. D. Boyd, S. M. Damon, K. D. Gibney, B. N. Conrad, N. S. Phillips, B. P. Rogers, Y. Gao, and B. A. Landman. Vanderbilt university institute of imaging science center for computational imaging XNAT: A multimodal data archive and processing environment. *NeuroImage*, 124:1097–1101, 2016.

[15] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. Katz, S. Shenker, and I. Stoica. Mesos: A platform for fine-grained resource sharing in the data center. In *8th USENIX Conf. Networked Systems Design and Implementation*, pages 295–308, 2011.

[16] Z. Hussain, T. Znati, and R. Melhem. Partial redundancy in hpc systems with non-uniform node reliabilities. In *SC*. IEEE Press, 2018.

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

[18] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. Dryad: Distributed data-parallel programs from sequential building blocks. In *2nd ACM SIGOPS/EuroSys European Conf. Computer Systems*, 2007.

[19] J. Kleinberg, Y. Rabani, and E. Tardos. Allocating bandwidth for bursty connections. In *STOC*, pages 664–673, 1997.

[20] B. Landman. Medical-image Analysis and Statistical Interpretation (MASI) Lab. `https://my.vanderbilt.edu/masi/`.

[21] D. A. Lifka. The ANL/IBM SP Scheduling System. In *JSSPP*, pages 295–303, 1995.

[22] R. H. Möhring, A. S. Schulz, and M. Uetz. Approximation in stochastic scheduling: The power of LP-based priority policies. *Journal of the ACM*, 46(6):924–942, 1999.

[23] A. W. Mu'alem and D. G. Feitelson. Utilization, Predictability, Workloads, and User Runtime Estimates in Scheduling the IBM SP2 with Backfilling. *IEEE Trans. Parallel Distrib. Syst.*, 12(6):529–543, 2001.

[24] J. Niño Mora. Stochastic scheduling. *Encyclopedia of Optimization*, pages 3818–3824, 2009.

[25] M. L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer-Verlag New York, Inc., Third edition, 2008.

[26] J. Skovira, W. Chan, H. Zhou, and D. A. Lifka. The EASY - LoadLeveler API Project. In *JSSPP*, pages 41–47, 1996.

[27] G. Staples. Torque resource manager. In *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, SC '06, 2006.

[28] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, B. Saha, C. Curino, O. O'Malley, S. Radia, B. Reed, and E. Baldeschwieler. Apache hadoop yarn: Yet another resource negotiator. In *the 4th Annual Symposium on Cloud Computing*, pages 5:1–5:16, 2013.

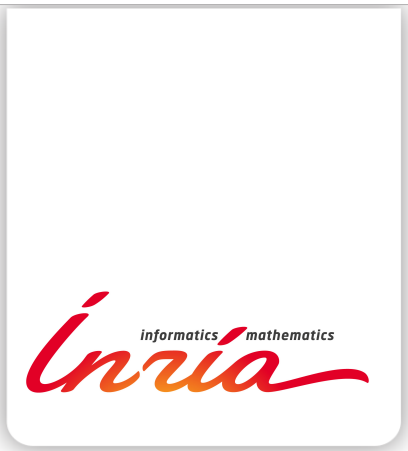
[29] O. Weidner, M. Atkinson, A. Barker, and R. Filgueira Vicente. Rethinking high performance computing platforms: Challenges, opportunities and recommendations. In *Proceedings of the ACM International Workshop on Data-Intensive Distributed Computing*, pages 19–26, 2016.

[30] K. Wolter, editor. *Stochastic Models for Fault Tolerance, Restart, Rejuvenation, and Checkpointing*. Springer Verlag, 2010.

[31] H. Xu and B. Li. Dynamic cloud pricing for revenue maximization. *IEEE Transactions on Cloud Computing*, 1(2):158–171, July 2013.

[32] A. B. Yoo, M. A. Jette, and M. Grondona. Slurm: Simple linux utility for resource management. In *JSSPP*, pages 44–60, 2003.

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



RESEARCH CENTRE
BORDEAUX – SUD-OUEST

200 avenue de la Vieille Tour
33405 Talence Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399