HAL
open science

Reservation Strategies for Stochastic Jobs

Guillaume Aupy, Ana Gainaru, Valentin Honoré, Padma Raghavan, Yves Robert, Hongyang Sun

To cite this version:

Guillaume Aupy, Ana Gainaru, Valentin Honoré, Padma Raghavan, Yves Robert, et al.. Reservation Strategies for Stochastic Jobs. IPDPS 2019 - 33rd IEEE International Parallel and Distributed Processing Symposium, May 2019, Rio de Janeiro, Brazil. pp.166-175, 10.1109/IPDPS.2019.00027 . hal-01968419v2

HAL Id: hal-01968419

https://inria.hal.science/hal-01968419v2

Submitted on 28 Feb 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Reservation Strategies for Stochastic Jobs

Guillaume Aupy*, Ana Gainaru†, Valentin Honoré*, Padma Raghavan†, Yves Robert‡, Hongyang Sun†
*Inria & Labri, Univ. of Bordeaux, Talence, France
†Department of EECS, Vanderbilt University, Nashville, TN, USA
‡Laboratoire LIP, ENS Lyon, France & University of Tennessee Knoxville, USA

***Abstract*—In this paper, we are interested in scheduling stochastic jobs on a reservation-based platform. Specifically, we consider jobs whose execution time follows a known probability distribution. The platform is reservation-based, meaning that the user has to request fixed-length time slots. The cost then depends on both (i) the request duration (pay for what you ask); and (ii) the actual execution time of the job (pay for what you use).**

A reservation strategy determines a sequence of increasing-length reservations, which are paid for until one of them allows the job to successfully complete. The goal is to minimize the total expected cost of the strategy. We provide some properties of the optimal solution, which we characterize up to the length of the first reservation. We then design several heuristics based on various approaches, including a brute-force search of the first reservation length while relying on the characterization of the optimal strategy, as well as the discretization of the target continuous probability distribution together with an optimal dynamic programming algorithm for the discrete distribution.

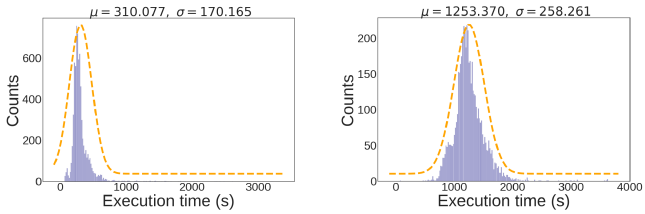
We evaluate these heuristics using two different platform models and cost functions: The first one targets a cloud-oriented platform (e.g., Amazon AWS) using jobs that follow a large number of usual probability distributions (e.g., Uniform, Exponential, LogNormal, Weibull, Beta), and the second one is based on interpolating traces from a real neuroscience application executed on an HPC platform. An extensive set of simulation results show the effectiveness of the proposed reservation-based approaches for scheduling stochastic jobs.

***Index Terms*—scheduling, stochastic job, reservation-based platform, sequence of requests, neuroscience applications**

I. Introduction

Scheduling a job onto a computing platform typically involves making a reservation of the required resources, say of duration t_1 seconds, and running the job on the platform until either the job has successfully completed, or the reservation time has elapsed, whichever comes first.

While in some instances the exact duration of the job may be known, in many other cases it is not known a priori (see, for examples, two neuroscience applications shown in Fig. 1, whose execution times have been characterized to exhibit input-dependent yet unpredictable behavior). In the latter case, the user has to guess a good value for t_1. Indeed, if the job does not complete successfully within these t_1 seconds, the user has to resubmit the job, this time requiring a longer reservation, say of length $t_2 > t_1$. If the job still does not complete successfully within t_2 seconds, the user has to try again, using a reservation of length $t_3 > t_2$, and so on until the job would succeed eventually. The cost to the user is then the cost associated with all the reservations that were necessary to the successful completion of the job.



(a) Functional MRI quality assurance (fMRIQA) [11]

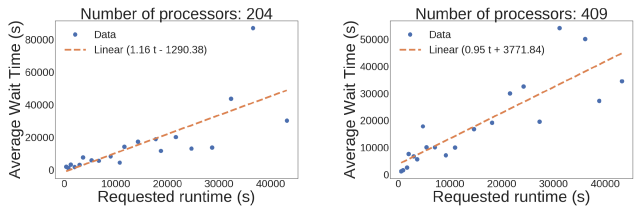
(b) Voxel-based morphometry quality assurance (VBMQA) [17]

Fig. 1. Traces of over 5000 runs (historgrams in purple) from July 2013 to October 2016 of two neuroscience applications from the Vanderbilt's medical imaging database [15]. We fit the data to LogNormal distributions (dotted lines in orange) with means and standard deviations shown on top.

A reservation strategy could well depend on the context, the type of jobs and the platform. As an example, the MASI Lab [1] at Vanderbilt takes the average execution time from the last few instances of a neuroscience job to determine the first reservation time for its next instance. If the reservation is not enough, a standard practice is to resubmit the job using between 1.5x and 2x the requested time in the last failed run, effectively doubling the reservation time. As another example, on HPC platforms, some users tend to reserve a walltime that "guarantees" execution success (say up to the 99th execution quantile). If this is not enough, they can ask for the 99th execution quantile of the remaining possibilities, etc.

This reservation-based approach is agnostic of the type of the job (sequential or parallel; single task or workflow) and of the nature of the required computing resources (processors of a large supercomputer, virtual machines on a cloud platform, etc.). The user just needs to make good guesses for the values of successive reservation durations, hoping to minimize the associated cumulated cost. Here, we refer to cost as a generic metric. It could be paid either in terms of budget (e.g., a monetary amount as a function of what is requested and/or used in a cloud service), or in terms of time (e.g., the waiting time of the job in an HPC queue that depends on the requested runtime as shown in Fig. 2).

The cost is usually proportional to the reservation length, with a possible initial and fixed value (start-up overhead). One example is the *Reserved Instance* model available on Amazon AWS [3], which is up to 75% cheaper than the flexible *On-Demand* model that does not require advanced reservations. We also investigate scenarios where an additional cost is paid in proportion to the actual execution time, again with



(a) Jobs that requested 204 procs. (b) Jobs that requested 409 procs.

Fig. 2. Average wait times of the jobs run on the same number of processors (204 and 409) as a function of the requested runtimes (data from [22]). All jobs are clustered into 20 groups, each with similar requested runtimes. Each point (in blue) shows the average wait time of all jobs in a group and the dotted lines (in orange) represent affine functions that fit the data.

a possible start-up overhead. This latter scenario is relevant when submitting jobs to large supercomputing platforms, where each user requests a set of resources for a given number of hours, but only pays for the hours actually spent; however, the assigned waiting queue, and hence the job's waiting time, both depend upon the number of hours asked for in the request.

Altogether, the cost function[1] for a job with a reservation of length t_1 and an actual execution duration of length t can be expressed as:

$$\alpha t_1 + \beta \min(t_1, t) + \gamma \tag{1}$$

where α, β and γ are constant parameters that depend on the platform and the cost model. Again, if $t > t_1$, another reservation should be made and paid for.

Although we do not know the exact execution time of the job to be scheduled, we do not schedule completely in the dark. Instead, we assume that there are many jobs of similar type and that their execution times obey the same (known) probability distribution (e.g., see Fig. 1). Each job is deterministic, meaning that a second execution of the same job will last exactly as long as the first one. However, the exact execution time of a given job is not known until that job has successfully completed. Our only assumption is that job execution times are randomly and uniformly sampled from a target probability distribution.

While the core of the theoretical results of this paper are valid for general continuous probability distributions, we focus on the usual distributions for the evaluation. In particular, we consider Uniform, Beta and Bounded Pareto distributions if the execution times are upper-bounded, i.e., they belong to some interval $[a, b]$; and we consider Exponential, Weibull, LogNormal and a few others if there is no upper bound for the execution times (see Section V for details). Note that the LogNormal distribution has been advocated to model file sizes [10], and we assume that job durations could naturally obey this distribution too. Note that we only consider distributions whose support is included in $[0, \infty)$, because execution times must have positive values. This precludes the use of Normal distribution, for instance.

This paper aims at proposing effective strategies to the following reservation problem: given a probability distribution, determine a (possibly infinite) sequence of reservations $S = (t_1, t_2, \ldots, t_i, t_{i+1}, \ldots)$ such that the expected cost to execute a job, whose execution time is randomly and uniformly sampled from the distribution, is minimized. Of course, any reservation sequence induces a greedy scheduling algorithm: for any given job, make (and pay for) a reservation of length t_1, then a reservation of length t_2 if the job has not succeeded (meaning its execution time t was greater than t_1), and so forth until success. The natural objective is to minimize the average cost of this algorithm over all possible job durations, hence the quest for a reservation sequence whose expected cost is minimal.

From a theoretical perspective, it is not clear that there always exists a reservation sequence with finite expected cost. However, we show that it is true for any continuous distribution with finite expectation and variance, which is the case for all the distributions considered in this work.

The main contributions of this work are the following:

- The characterization of an optimal reservation sequence up to the value of its first reservation duration t_1. While we do not know how to compute t_1 analytically, we provide an upper bound that allows us to limit the range of a numerical search for its value;
- The design of several heuristics based on various approaches: one explores a brute-force search for t_1 while relying on the optimal characterization mentioned above; one discretizes the target continuous distribution and uses an optimal dynamic programming algorithm for the discrete distribution; and some rely on standard measures (e.g., mean, variance, quantiles) of the distribution.
- An extensive set of simulation results under two different platform models and cost functions that show the effectiveness of the proposed strategies. The first one targets a cloud-oriented platform using jobs that follow a large number of usual distributions and the second one is based on interpolating traces from a real neuroscience application executed on an HPC platform.

The rest of the paper is organized as follows. Section II introduces the framework and main notations. Section III discusses the properties of the optimal solution. We propose several heuristics in Section IV, and evaluate their performance under two platform models in Section V. Section VI is dedicated to related work. Finally, we provide concluding remarks and hints for future work in Section VII.

II. Framework

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

A. Stochastic jobs

We consider stochastic jobs whose execution times are unknown but (i) deterministic, so that two successive executions

[1]Other cost functions could be envisioned. In particular, the cost for a reservation could be a more general function than a simple affine one. Several results of this paper can be extended to convex cost functions. We focus on affine costs because of their wide applicability under various scenarios.

of the same job will have the same duration; and (ii) randomly and uniformly sampled from a given probability distribution law $\mathcal{D}$, whose PDF is f and cumulative distribution function (CDF) is F. The probability distribution is assumed to be nonnegative, since we model execution times, and is defined either on a finite support $[a, b]$, where $0 \leq a < b$, or on an infinite support $[a, \infty)$ where $0 \leq a$. 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 $t \in [0, a]$.

B. Cost model

To execute a job, the user makes a series of reservations, until the job successfully executes within the length of the last reservation. For a reservation of length t_1, and for an actual duration t of the job, the cost is $\alpha t_1 + \beta \min(t_1, t) + \gamma$, as stated in Equation (1), where $\alpha > 0$, $\beta \geq 0$ and $\gamma \geq 0$. If $t > t_1$, another reservation should be paid for. Hence, the user needs to make a (possibly infinite) sequence of reservations $S = (t_1, t_2, \dots, t_i, t_{i+1}, \dots)$, where:

1) $t_i < t_{i+1}$ for all $i \geq 1$. Indeed, because jobs are deterministic, it is redundant to have a duration in the sequence that is not strictly larger than the previous one, hence that duration can be removed from the sequence;
2) all possible execution times of the job are indeed smaller than or equal to some t_i in the sequence. This simply means that the sequence must tend to infinity if job execution times are not upper-bounded.

Throughout the paper, we assume that both properties hold when speaking of a reservation sequence. For notational convenience, we define $t_0 = 0$, in order to simplify summations.

Now, for a sequence $S = (t_1, t_2, \dots, t_i, t_{i+1}, \dots)$, and for a job execution time t, the cost is

$$C(k, t) = \sum_{i=1}^{k-1} (\alpha t_i + \beta t_i + \gamma) + \alpha t_k + \beta t + \gamma \qquad (2)$$

where k is the smallest index in the sequence such that $t \leq t_k$ (or equivalently, $t_{k-1} < t \leq t_k$; recall that $t_0 = 0$).

C. Objective

The goal is to find a scheduling strategy, i.e., a sequence of increasing reservation durations, that minimizes the cost in expectation. Formally, the expected cost for a sequence $S = (t_1, t_2, \dots, t_i, t_{i+1}, \dots)$ can be written as:

$$\mathbb{E}(S) = \sum_{k=1}^{\infty} \int_{t_{k-1}}^{t_k} C(k, t) f(t) dt \qquad (3)$$

Indeed, when $t_{k-1} < t \leq t_k$, the cost is $C(k, t)$, weighted with the corresponding probability. Here are two examples:

• UNIFORM(a, b): for a uniform distribution over the interval $[a, b]$ where $0 < a < b$, we have $f(t) = \frac{1}{b-a}$ if $a \leq t \leq b$, and $f(t) = 0$ otherwise. Given a finite sequence $S = (\frac{a+b}{2}, b)$, the expected cost is

$$\begin{aligned}\mathbb{E}(S) &= \int_a^{\frac{a+b}{2}} (\alpha \tfrac{a+b}{2} + \beta t + \gamma) \tfrac{1}{b-a} dt \\ &\quad + \int_{\frac{a+b}{2}}^{b} \left((\alpha \tfrac{a+b}{2} + \beta \tfrac{a+b}{2} + \gamma) + (\alpha b + \beta t + \gamma) \right) \tfrac{1}{b-a} dt\end{aligned}$$

The first term is for values of t that are in $[a, \frac{a+b}{2}]$ and the second term is for larger values of t in $[\frac{a+b}{2}, b]$. For the latter term, we pay a constant cost $\alpha \frac{a+b}{2} + \beta \frac{a+b}{2} + \gamma$ for the first unsuccessful reservation, and then a cost that depends upon the value of t for the second reservation if $\beta \neq 0$.

• EXP(λ): for an exponential distribution with rate λ and support in $[0, \infty)$, we have $f(t) = \lambda e^{-\lambda t}$ for all $t \geq 0$. Given an infinite and unbounded sequence $S = (\frac{1}{\lambda}, \frac{2}{\lambda}, \dots, \frac{i}{\lambda}, \frac{i+1}{\lambda}, \dots)$, the expected cost is

$$\mathbb{E}(S) = \sum_{k=1}^{\infty} \int_{\frac{k-1}{\lambda}}^{\frac{k}{\lambda}} \Big(\sum_{i=1}^{k-1} (\alpha \tfrac{i}{\lambda} + \beta \tfrac{i}{\lambda} + \gamma) + \alpha \tfrac{k}{\lambda} + \beta t + \gamma \Big) \lambda e^{-\lambda t} dt$$

Again, when $t \in [\frac{k-1}{\lambda}, \frac{k}{\lambda}]$, we pay a fixed cost for the $k-1$ first reservations, and a possibly variable cost for the k-th reservation. Looking at the expression of $\mathbb{E}(S)$ above, we easily see that the given sequence S has a finite expected cost $\mathbb{E}(S)$. In fact, there are many sequences with finite expected cost, such as those defined by $t_i = ui + v$ for $i \geq 1$, where u and v are positive constants.

We are now ready to state the optimization problem:

Definition 1 (STOCHASTIC)**.** Given a probability distribution (with CDF F) for the execution times of stochastic jobs, and given a cost function given by Equation (1) (with parameters α, β and γ), find a reservation sequence S with minimal expected cost $\mathbb{E}(S)$ as given in Equation (3).

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.

III. CHARACTERIZING THE OPTIMAL SOLUTION

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

A. Cost function

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

Theorem 1. *Given a sequence $S = (t_1, t_2, \dots, t_i, t_{i+1}, \dots)$, the cost function given by Equation* (3) *(with parameters α, β and γ) can be rewritten as (with $t_0 = 0$):*

$$\mathbb{E}(S) = \beta \cdot \mathbb{E}[X] + \sum_{i=0}^{\infty} (\alpha t_{i+1} + \beta t_i + \gamma) \, \mathbb{P}(X \geq t_i) \qquad (4)$$

Proof. We first expand Equation (3) as follows:

$$\mathbb{E}(S)=\sum_{k=1}^{\infty}\left(\int_{t_{k-1}}^{t_k}\Big(\sum_{i=1}^{k}(\alpha t_i+\gamma)+\sum_{i=1}^{k-1}\beta t_i+\beta t\Big)f(t)dt\right) \quad (5)$$

We compute the three terms on the right-hand side separately. By defining $t_0 = 0$, the first term can be expressed as:

$$\begin{aligned}&\textstyle\sum_{k=1}^{\infty}\left(\int_{t_{k-1}}^{t_k}\left(\sum_{i=1}^{k}(\alpha t_i+\gamma)\right)f(t)dt\right)\\ &\quad=\textstyle\sum_{k=1}^{\infty}\sum_{i=1}^{k}(\alpha t_i+\gamma)\int_{t_{k-1}}^{t_k}f(t)dt\\ &\quad=\textstyle\sum_{k=1}^{\infty}\sum_{i=1}^{k}(\alpha t_i+\gamma)\,\mathbb{P}(X\in[t_{k-1},t_k])\\ &\quad=\textstyle\sum_{i=1}^{\infty}\sum_{k=i}^{\infty}(\alpha t_i+\gamma)\,\mathbb{P}(X\in[t_{k-1},t_k])\\ &\quad=\textstyle\sum_{i=1}^{\infty}(\alpha t_i+\gamma)\,\mathbb{P}(X\geq t_{i-1})\end{aligned}$$

Similarly, we obtain the second term:

$$\textstyle\sum_{k=1}^{\infty}\left(\int_{t_{k-1}}^{t_k}\left(\sum_{i=1}^{k-1}\beta t_i\right)f(t)dt\right)=\sum_{i=1}^{\infty}\beta t_i\mathbb{P}(X\geq t_i)$$

and the third term:

$$\textstyle\sum_{k=1}^{\infty}\left(\int_{t_{k-1}}^{t_k}\beta t f(t)dt\right)=\beta\cdot\mathbb{E}[X]$$

Plugging these three terms back into Equation (5), we get the desired expression for the cost function as given by Equation (4). □

B. Upper bound on t_1^o and finite expected cost

In this section, we extract an upper bound for the first request t_1^o of an optimal sequence S^o to STOCHASTIC, which allows us to show that the expected cost $\mathbb{E}(S^o)$ is upper bounded too, and hence finite. This result holds in a general setting, namely, for any distribution $\mathcal{D}$ such that $\mathbb{E}(X^2) < \infty$.

Obviously, if the distribution's support is upper bounded, such as for UNIFORM(a, b), a solution is to choose that upper bound for t_1^o (e.g., $t_1^o \leq b$ for UNIFORM(a, b)). Hence, we focus on distributions with infinite support $[a, \infty)$ and aim at restricting the search for an optimal t_1^o to a bounded interval $[a, A_1]$ for some A_1. We derive the following result.

Theorem 2. *For any distribution $\mathcal{D}$ with infinite support $[a, \infty)$ such that $\mathbb{E}[X^2] < \infty$, the value t_1^o of an optimal sequence $S^o = (t_1^o, t_2^o, \dots, t_i^o, t_{i+1}^o, \dots)$ satisfies $t_1^o \leq A_1$, and $\mathbb{E}(S^o) \leq A_2$, where*

$$A_1=\mathbb{E}[X]+1+\frac{\alpha+\beta}{2\alpha}(\mathbb{E}[X^2]-a^2)+\frac{\alpha+\beta+\gamma}{\alpha}(\mathbb{E}[X]-a) \quad (6)$$

$$A_2=\beta\cdot\mathbb{E}(X)+\alpha A_1+\gamma \quad (7)$$

Proof. We consider the sequence $S = (t_1, t_2, \dots, t_i, t_{i+1}, \dots)$ with $t_i = a + i$ for $i \geq 1$ (and $t_0 = 0$), and compute

$$\begin{aligned}&\mathbb{E}(S)-\beta\cdot\mathbb{E}[X]=\textstyle\sum_{i=0}^{\infty}(\alpha t_{i+1}+\beta t_i+\gamma)\mathbb{P}(X\geq t_i)\\ &\quad=\textstyle\sum_{i=0}^{\infty}(\alpha(a+i+1)+\beta(a+i)+\gamma)\mathbb{P}(X\geq a+i)\\ &\quad=\alpha(a+1)+\gamma+\textstyle\sum_{i=1}^{\infty}(\alpha+\beta)(a+i)\mathbb{P}(X\geq a+i)\\ &\qquad+(\alpha+\gamma)\textstyle\sum_{i=1}^{\infty}\mathbb{P}(X\geq a+i)\\ &\quad=\alpha(a+1)+\gamma+(\alpha+\beta)\textstyle\sum_{i=1}^{\infty}\int_{a+i-1}^{a+i}(a+i)\mathbb{P}(X\geq a+i)dt\\ &\qquad+(\alpha+\gamma)\textstyle\sum_{i=1}^{\infty}\int_{a+i-1}^{a+i}\mathbb{P}(X\geq a+i)dt\end{aligned}$$

Note that for all $t \in [a+i-1, a+i]$, we have both $a+i \leq t+1$ and $\mathbb{P}(X \geq a + i) \leq \mathbb{P}(X \geq t)$, thus

$$(a+i)\mathbb{P}(X\geq a+i)\leq(t+1)\mathbb{P}(X\geq t)$$

Hence, we can write:

$$\begin{aligned}&\mathbb{E}(S)-\beta\cdot\mathbb{E}[X]\\ &\quad\leq\alpha(a+1)+\gamma+(\alpha+\beta)\textstyle\sum_{i=1}^{\infty}\int_{a+i-1}^{a+i}(t+1)\mathbb{P}(X\geq t)dt\\ &\qquad+(\alpha+\gamma)\textstyle\sum_{i=1}^{\infty}\int_{a+i-1}^{a+i}\mathbb{P}(X\geq t)dt\\ &\quad=\alpha(a+1)+\gamma+(\alpha+\beta)\textstyle\int_{a}^{\infty}(t+1)\mathbb{P}(X\geq t)dt\\ &\qquad+(\alpha+\gamma)\textstyle\int_{a}^{\infty}\mathbb{P}(X\geq t)dt\\ &\quad\leq\alpha(a+1)+\gamma+(\alpha+\beta)\textstyle\int_{a}^{\infty}t\cdot\mathbb{P}(X\geq t)dt\\ &\qquad+(2\alpha+\beta+\gamma)\textstyle\int_{a}^{\infty}\mathbb{P}(X\geq t)dt\end{aligned}$$

For the last inequality, we have split $\int_a^\infty (t + 1)\mathbb{P}(X \geq t)dt$ into $\int_a^\infty t\mathbb{P}(X \geq t)dt$ and $\int_a^\infty \mathbb{P}(X \geq t)dt$.

Extending the support of $\mathcal{D}$ to $[0, \infty)$ by letting $f(t) = 0$ for $0 \leq t \leq a$, and hence $\mathbb{P}(X \geq t) = 1$ for $0 \leq t \leq a$, we have the following property for any integer $p \geq 1$:

$$\begin{aligned}&\textstyle\int_0^\infty t^{p-1}\cdot\mathbb{P}(X\geq t)dt=\int_{t=0}^{\infty}t^{p-1}\int_{x=t}^{\infty}f(x)dxdt\\ &\quad=\textstyle\int_{x=0}^{\infty}f(x)\int_{t=0}^{x}t^{p-1}dtdx=\int_0^\infty\frac{x^p}{p}f(x)dx=\frac{\mathbb{E}[X^p]}{p}\end{aligned}$$

Hence, using $p = 1$, we have:

$$\textstyle\int_a^\infty\mathbb{P}(X\geq t)dt=\int_0^\infty\mathbb{P}(X\geq t)dt-\int_0^a\mathbb{P}(X\geq t)dt=\mathbb{E}[X]-a$$

and using $p = 2$, we get

$$\begin{aligned}\textstyle\int_a^\infty t\cdot\mathbb{P}(X\geq t)dt&=\textstyle\int_0^\infty t\cdot\mathbb{P}(X\geq t)dt-\int_0^a t\cdot\mathbb{P}(X\geq t)dt\\ &=\frac{\mathbb{E}[X^2]-a^2}{2}\end{aligned}$$

Altogether, we derive that

$$\mathbb{E}(S)\leq\beta\cdot\mathbb{E}[X]+\alpha A_1+\gamma \quad (8)$$

where A_1 is given by Equation (6). From Equation (4), the expected cost of any sequence S satisfies $\mathbb{E}(S) \geq \beta \cdot \mathbb{E}[X] + \alpha t_1 + \gamma$ (cost of expected execution time and cost of first request). Hence, necessarily in an optimal sequence, the first reservation t_1^o satisfies to $t_1^o \leq A_1$. Thus, Equation (6) gives the desired bound on t_1^o. □

C. Properties of optimal sequences

We now derive a recurrence relation between the successive requests in the optimal sequence for STOCHASTIC.

Theorem 3. *Let $S^o = (t_i^o)_{i\geq 1}$ denote an optimal sequence for* STOCHASTIC. *For all $i \geq 1$, if $F(t_i^o) \neq 1$, we have the following property:*

$$\alpha t_{i+1}^o+\beta t_i^o+\gamma=\alpha\frac{1-F(t_{i-1}^o)}{f(t_i^o)}+\beta\frac{1-F(t_i^o)}{f(t_i^o)} \quad (9)$$

Proof. We fix an index $j \geq 1$ such that $F(t_j^o) \neq 1$ and consider the expected cost when we replace t_j^o by an arbitrary value $t \in [t_{j-1}^o, t_{j+1}^o]$. This amounts to using the sequence $S_j^o(t) = (t_1^o, t_2^o, \cdots, t_{j-1}^o, t, t_{j+1}^o, \cdots)$ whose expected cost, according to Equation (4), is the following:

$$\begin{aligned}\mathbb{E}(S_j^o(t))=\beta\cdot\mathbb{E}[X]+&\sum_{i\neq j-1,j}(\alpha t_{i+1}^o+\beta t_i^o+\gamma)\mathbb{P}(X\geq t_i^o)\\ &+(\alpha t+\beta t_{j-1}^o+\gamma)\mathbb{P}(X\geq t_{j-1}^o)\\ &+(\alpha t_{j+1}^o+\beta t+\gamma)\mathbb{P}(X\geq t)\end{aligned}$$

which we can rewrite as:

$$\mathbb{E}(S_j^o(t))=C_j+\alpha t(1-F(t_{j-1}^o))+(\alpha t_{j+1}^o+\beta t+\gamma)(1-F(t))$$

where C_j is some constant independent of t. By definition, the minimum of $\mathbb{E}(S_j^o(t))$ on $[t_{j-1}^o, t_{j+1}^o]$ is achieved at $t = t_j^o$ (and potentially at other values). Because $\mathbb{E}(S_j^o(t))$ is smooth, we have that its derivative at t_j^o, which is not an extremity of the interval $[t_{j-1}^o, t_{i+1}^o]$, must be equal to zero, i.e., $\frac{\partial \mathbb{E}(S_j^o(t))}{\partial t} = 0$. This gives:

$$\alpha(1 - F(t_{j-1}^o)) + \beta(1 - F(t_j^o)) - (\alpha t_{j+1}^o + \beta t_j^o + \gamma) f(t_j^o) = 0 \quad (10)$$

To get the final result, it remains to show that $f(t_j^o) \neq 0$. Otherwise, we would get from Equation (10) that $\alpha(1 - F(t_{j-1}^o)) + \beta(1 - F(t_j^o)) = 0$, which implies that $F(t_{j-1}^o) = 1$ because $\alpha > 0$ (and $\beta(1 - F(t_j^o)) \geq 0$). But then, $F(t_j^o) \geq F(t_{j-1}^o) = 1$, which contradicts the initial assumption. Hence, $f(t_j^o) \neq 0$, and rewriting Equation (10) directly leads to Equation (9). □

Note that the condition $F(t_i^o) \neq 1$ in Theorem 3 applies to distributions with finite support, such as UNIFORM(a, b), where $F(b) = 1$. For the usual distributions with infinite support, such as EXP(λ), we have $F(t) < 1$ for all $t \in [0, \infty)$ and an optimal sequence must be infinite. In essence, Theorem 3 suggests that an optimal sequence is characterized solely by its first value t_1^o:

Proposition 1. *For a smooth distribution with infinite support, solving* STOCHASTIC *reduces to finding t_1^o that minimizes*

$$\sum_{i=0}^{\infty} (\alpha t_{i+1} + \beta t_i + \gamma)\, \mathbb{P}(X \geq t_i)$$

where $t_0^o = 0$, and for all $i \geq 2$,

$$t_i^o = \frac{1 - F(t_{i-2}^o)}{f(t_{i-1}^o)} + \frac{\beta}{\alpha}\left(\frac{1 - F(t_{i-1}^o)}{f(t_{i-1}^o)} - t_{i-1}^o\right) - \frac{\gamma}{\alpha} \quad (11)$$

For a smooth distribution with finite support, the recurrence in Equation (11) *still holds but the optimal sequence stops as soon as it reaches t_i^o with $F(t_i^o) = 1$.*

Proposition 1 provides an optimal algorithm for general smooth distributions, up to the determination of t_1^o. However, computing the optimal t_1^o, remains a difficult problem, except for simple distributions such as UNIFORM(a, b) (see Section III-D).

D. Uniform distributions

In this section, we discuss the optimal strategy for a uniform distribution UNIFORM(a, b), where $0 < a < b$. Intuitively, one could try and make a first reservation of duration, say, $t_1 = \frac{a+b}{2}$, and then a second reservation of duration $t_2 = b$. However, we show that the best approach is to make a single reservation of duration $t_1 = b$, for any value of the parameters α, β and γ:

Theorem 4. *For a uniform distribution* UNIFORM(a, b)*, the optimal sequence for* STOCHASTIC *is $S^o = (b)$.*

Proof. We study here only the sequences of length smaller than or equal to 2, i.e., $S = (t_1, b)$, where $t_1 \leq b$. The full proof is available in the companion report [4].

Note that necessarily, in a sequence of length 2, $t_2 = b$ otherwise $t_2 < b$ and $\mathbb{E}((t_1, t_2)) = \infty$ because the interval $[t_2, b]$ has non-zero measure.

Using Equation (4) we obtain:

$$\begin{aligned}\mathbb{E}((t_1, b)) &= \beta\mathbb{E}(X) + (\alpha t_1 + \gamma)\mathbb{P}(X \geq 0) \\ &\quad + (\alpha b + \beta t_1 + \gamma)\mathbb{P}(X \geq t_1) \\ &= \beta\mathbb{E}(X) + (\alpha t_1 + \gamma) + \tfrac{b - t_1}{b - a}(\alpha b + \beta t_1 + \gamma) \\ &= \beta\mathbb{E}(X) + \alpha\tfrac{b^2 - t_1 a}{b - a} + \tfrac{b - t_1}{b - a}(\beta t_1 + \gamma) + \gamma\end{aligned}$$

We can verify easily that this is minimized when $t_1 = b$ (and that $\mathbb{E}((t_1, b)) = \mathbb{E}((t_1))$. Hence $S^o = (b)$ is optimal amongst the sequences of length smaller than or equal to 2. □

E. Exponential distributions

In this section, we provide partial results for the RESERVATIONONLY problem ($\beta = \gamma = 0$ and $\alpha = 1$) with an exponential distribution EXP(λ). From Theorem 2 (and the example in Section II-C), we know that there exist sequences of finite expected cost. We further characterize the optimal solution as follows:

Proposition 2. *Let $S_1 = (s_1, s_2, \ldots, s_i, s_{i+1}, \ldots)$ denote the optimal sequence for* RESERVATIONONLY *with an* EXP(1) *distribution. Then, $s_2 = e^{s_1}$, and for $i \geq 3$,*

$$s_i = e^{s_{i-1} - s_{i-2}} = \frac{e^{s_{i-1}}}{\Pi_{j=2}^{i-1} s_j} \quad (12)$$

The expected cost of S_1 is $\mathbb{E}(S_1) = s_1 + 1 + \sum_{i=1}^{\infty} e^{-s_i}$.

Furthermore, the optimal sequence for RESERVATIONONLY *with* EXP(λ) *distribution is the infinite sequence $S_\lambda = (t_1, t_2, \ldots, t_i, t_{i+1}, \ldots)$ such that $t_i = \frac{s_i}{\lambda}$ for $i \geq 1$. Its expected cost is $\mathbb{E}(S_\lambda) = \frac{1}{\lambda}\mathbb{E}(S_1)$.*

Proof. Consider an EXP(λ) distribution. From Equation (4), the expected cost of the optimal sequence S_λ is $\mathbb{E}(S) = \sum_{i=0}^{\infty} t_{i+1} e^{-\lambda t_i}$ where $t_0 = 0$, t_1 is unknown, and the value of t_i for $i \geq 2$ is given by Equation (11) as $t_i = \frac{e^{\lambda(t_{i-1} - t_{i-2})}}{\lambda}$ for $i \geq 2$. Introducing $s_i = \lambda t_i$ for all $i \geq 0$, we derive that

$$\mathbb{E}(S_\lambda) = \frac{1}{\lambda}\sum_{i=0}^{\infty} s_{i+1} e^{-s_i}$$

with $s_i = e^{s_{i-1} - s_{i-2}}$ for all $i \geq 2$. We have $s_{i+1} e^{-s_i} = e^{-s_{i-1}}$ for $i \geq 1$, which gives the desired value for $\mathbb{E}(S_\lambda)$.

Now, we prove Equation (12) by induction. It holds for $i = 3$, because $s_3 = e^{s_2 - s_1} = \frac{e^{s_2}}{e^{s_1}} = \frac{e^{s_2}}{s_2}$. Assume that it holds for any $j \leq i$. Then $s_{i+1} = e^{s_i - s_{i-1}} = \frac{e^{s_i}}{e^{s_{i-1}}}$, and by induction $e^{s_{i-1}} = s_i s_2 \ldots s_{i-1}$, hence the result. □

Again, the optimal sequence is fully characterized by the value of t_1 or s_1. Here, s_1 is independent of λ. In other words, the solution for EXP(1) is generic, and the solution for EXP(λ) for an arbitrary λ can be directly derived from it. Unfortunately, we do not know how to compute s_1 analytically. However, a brute-force search provides the value $s_1 \approx 0.742$, which means that the first reservation for EXP(λ) should be approximately three quarters of the mean value $\frac{1}{\lambda}$ of the distribution, for any $\lambda > 0$.

IV. Heuristics for Arbitrary Distributions

The results of the preceding section provide a strategy to compute the optimal sequence up to the determination of t_1^o, since Theorem 3 and Proposition 1 allow us to compute the subsequent t_i^o's. However, while we have derived an upper bound on t_1^o, we do not know how to compute its exact value for an arbitrary distribution. In this section, we introduce several heuristics for the STOCHASTIC problem under arbitrary probability distributions.

A. Brute-force procedure

We first present a procedure called BRUTE-FORCE that simply tries different values for the first reservation length t_1 in a sequence S, and then computes the subsequent values according to Equation (11). Specifically, we try M different values of t_1 on the interval $[a, b]$, where a is the lower bound of the distribution and b is the upper bound if the distribution is finite. Otherwise, we set $b = A_1$, which is an upper bound on the optimal t_1^o as given in Equation (6). As an example, when $\alpha = 1, \beta = \gamma = 0$, we have $A_1 \leq 1 + 2\mathbb{E}(X) + \frac{\mathbb{E}(X^2)}{2}$. For each $m = 1, \ldots, M$, we generate a sequence that starts with $t_1 = a + m \cdot \frac{b-a}{M}$. Given a sequence S, its expected cost is evaluated via a Monte-Carlo process, as described in Section V-A: we randomly draw N execution times from the distribution, and compute the expected cost incurred by the sequence over the N samples. We finally return the minimum expected cost found over all the M values of t_1. Note that some values of t_1 may not lead to any result, because the sequence computed based on it and using Equation (11) may not be strictly increasing. In this case, we simply ignore the sequence. The complexity of this heuristic is $\mathcal{O}(MN)$.

We point out that the actual optimal value for the first request t_1^o would possibly lie in between two successive values of t_1 that we try. However, because we deal with smooth probability distributions, we expect to return a t_1 and an associated expected cost that are close to the optimal when M and N are sufficiently large. In the performance evaluation, we set $M = 5000$ and $N = 1000$.

B. Discretization-based dynamic programming

We now present a heuristic that approximates the optimal solution for STOCHASTIC by first discretizing the continuous distribution and then computing an optimal sequence for the discrete problem via dynamic programming.

1) Truncating and discretizing continuous distributions: If a continuous distribution has finite support $[a, b]$, where $0 \leq a < b$, then we can directly discretize it. Otherwise, for a distribution with infinite support $[a, \infty)$, where $0 \leq a$, we need to first truncate it in order to operate on a bounded interval. In the latter case, we define $b = Q(1 - \epsilon)$, where $Q(x) = \inf\{t | F(t) \geq x\}$ is the quantile function. That is, we discard the final $\epsilon \in (0, 1)$ quantile of the distribution, which for usual distributions ensures that b is finite. In either case, the discretization will then be performed on the interval $[a, b]$. Let n denote the number of discrete values we will sample from the continuous distribution. The result will be a set of n pairs $(v_i, f_i)_{i=1\ldots n}$, where the v_i's represent the possible execution times of the jobs, and the f_i's represent the corresponding probabilities. We envision two schemes for the discretization:

- EQUAL-PROBABILITY: This scheme ensures that all the discrete execution times have the same probability. Thus, for all $i = 1, 2, \ldots, n$, we can compute $v_i = Q\big(i \cdot \frac{F(b)}{n}\big)$ and $f_i = \frac{F(b)}{n}$.
- EQUAL-TIME: This scheme makes the discrete execution times equally spaced in the interval $[a, b]$. Thus, for all $i = 1, 2, \ldots, n$, the execution times and their probabilities are computed as $v_i = a + i \cdot \frac{b-a}{n}$ and $f_i = F(v_i) - F(v_{i-1})$.

Note that when the continuous distribution has infinite support, the probabilities for the n discrete execution times do not sum up to 1, i.e., $\sum_{i=1}^{n} f_i = F(b) = 1 - \epsilon$. A smaller value of ϵ and a larger number n will provide a better sampling of the continuous distribution in either discretization scheme. In the performance evaluation, we set $\epsilon = 10^{-7}$ and $n = 1000$.

2) Dynamic programming for discrete distributions: We now present a dynamic programming algorithm to compute the optimal sequence for any discrete probability distribution. It will be used with the discretization schemes to approximate the optimal solution for an arbitrary continuous distribution.

Theorem 5 (Discrete distribution)**.** *If $X \sim (v_i, f_i)_{i=1\ldots n}$, then* STOCHASTIC *can be solved optimally in polynomial time.*

Proof. Let $\mathbb{E}_i^*$ denote the optimal expected cost given that $X \geq v_i$. In this case, to compute the optimal expected cost, the probability distribution of X needs to be first updated as $f'_k = \frac{f_k}{\sum_{j=i}^{n} f_j}, \forall k = i, \ldots, n$, which guarantees that $\sum_{k=i}^{n} f'_k = 1$.

We can then express $\mathbb{E}_i^*$ based on the following dynamic programming formulation:

$$\mathbb{E}_i^* = \min_{i \leq j \leq n} \left(\alpha v_j + \gamma + \sum_{k=i}^{j} f'_k \cdot \beta v_k + \Big(\sum_{k=j+1}^{n} f'_k \Big) \big(\beta v_j + \mathbb{E}_{j+1}^* \big) \right)$$

In particular, to compute $\mathbb{E}_i^*$, we make a first reservation of all possible discrete values $(v_j)_{j=i\ldots n}$ and select the one that incurs the minimum total expected cost. For each v_j considered, if the job's actual execution time is greater than v_j (with probability $\sum_{k=j+1}^{n} f'_k$), the total cost also includes the optimal cost $\mathbb{E}_{j+1}^*$ for making subsequent reservations.

The dynamic program is initialized with $\mathbb{E}_n^* = \alpha v_n + \beta v_n + \gamma$, and the optimal total expected cost is given by $\mathbb{E}_1^*$. The complexity is $\mathcal{O}(n^2)$, since each $\mathbb{E}_i^*$ depends on $n - i$ other expected costs, with associated probability updates and summations that can be computed in $\mathcal{O}(n - i)$ time. The optimal sequence of reservations can be obtained by backtracking the decisions made at each step. □

Note that the sequence obtained by dynamic programming always ends with the largest value $v_n = b$. When applying it back to a continuous distribution with infinite support, more values will be needed, because the sequence must tend to infinity as explained in Section II-B. In this case, additional values can be appended to the sequence by using other heuristics, such as the ones presented next in Section IV-C.

C. Other heuristics

We finally present some simple heuristics that are inspired by common resource allocation strategies in the literature. These heuristics do not explore the structure of the optimal solution nor the probability distribution, but rely on simple incremental methods to generate reservation sequences.

In the following, we will use $\mu = \mathbb{E}(X)$ to denote the mean of a given distribution, $\sigma^2 = \mathbb{E}(X^2) - \mu^2$ to denote its variance, and $m = Q(\frac{1}{2})$ to denote its median, where $Q(x) = \inf\{t|F(t) \geq x\}$ represents the quantile function. The different heuristics are defined as follows:

- MEAN-BY-MEAN: start with the mean (i.e., $t_1 = \mu$) and then make each subsequent reservation request by computing the conditional expectation of the distribution in the remaining interval, i.e., $t_i = \mathbb{E}(X|X > t_{i-1}) = \frac{\int_{t_{i-1}}^{\infty} tf(t)dt}{1-F(t_{i-1})}$ for all $i \geq 2$.
- MEAN-STDEV: start with the mean and then increment the reservation length by one standard deviation (σ) for each subsequent request, i.e., $t_i = \mu + (i-1)\sigma$ for all $i \geq 2$.
- MEAN-DOUBLING: start with the mean and then double the reservation length for each subsequent request, i.e., $t_i = 2^{i-1}\mu$ for all $i \geq 2$.
- MEDIAN-BY-MEDIAN: each request is the median of the distribution in the remaining interval, i.e., $t_i = Q(1 - \frac{1}{2^i})$ for all $i \geq 2$.

Note that deriving the sequence for MEAN-BY-MEAN is straightforward for some distributions (e.g., exponential, uniform), but more involved for others. Recursive formulas are provided in [4] to compute the sequence using this heuristic for the considered distributions, along with key parameters (e.g., mean, variance, quantile) to facilitate the sequence computations for the other heuristics.

V. PERFORMANCE EVALUATION

In this section, we evaluate the different heuristics presented in Section IV, and compare their performance. The code and setup of the experiments presented in this section are publicly available on https://gitlab.inria.fr/vhonore/ipdps_2019_stochastic-scheduling.

A. Evaluation methodology

For each heuristic that generates a reservation sequence $S = (t_1, t_2, \dots, t_i, t_{i+1}, \dots)$ under a particular probability distribution $\mathcal{D}$, we approximate its expected cost via a Monte-Carlo process[2]: we randomly sample N possible execution times from the distribution, and then average over the cost of all the N samples, i.e.,

$$\tilde{\mathbb{E}}(S) = \frac{1}{N}\sum_{i=1}^{N} C(k,t)|_{t \leftarrow \mathcal{D}} \tag{13}$$

where $C(k,t)$ is the cost for a specific execution time t drawn from the distribution, computed using Equation (2). For the presented evaluation results, we set $N = 1000$.

[2]The possibly infinite sequence prevents us from analytically evaluating its expected cost.

Table I. Probability distributions and parameter instantiations

Distribution	PDF $f(t)$	Instantiation	Support
Distributions with infinite support			
Exponential (λ)	$\lambda e^{-\lambda t}$	$\lambda = 1.0$	$t \in [0, \infty)$
Weibull(λ, κ)	$\frac{\kappa}{\lambda}\left(\frac{t}{\lambda}\right)^{\kappa-1} e^{-\left(\frac{t}{\lambda}\right)^{\kappa}}$	$\lambda = 1.0$ $\kappa = 0.5$	$t \in [0, \infty)$
Gamma(α, β)	$\frac{\beta^{\alpha}}{\Gamma(\alpha)} t^{\alpha-1} e^{-\beta t}$	$\alpha = 2.0$ $\beta = 2.0$	$t \in [0, \infty)$
LogNormal (ν, κ^2)	$\frac{1}{t\kappa\sqrt{2\pi}} e^{-\frac{(\ln t - \nu)^2}{2\kappa^2}}$	$\nu = 3.0$ $\kappa = 0.5$	$t \in (0, \infty)$
TruncatedNormal(ν, κ^2, a)	$\frac{1}{\kappa}\sqrt{\frac{2}{\pi}} \cdot \frac{e^{-\frac{1}{2}\left(\frac{t-\nu}{\kappa}\right)^2}}{1-\mathrm{erf}\left(\frac{a-\nu}{\kappa\sqrt{2}}\right)}$	$\nu = 8.0$ $\kappa^2 = 2.0$ $a = 0.0$	$t \in [a, \infty)$
Pareto(ν, α)	$\frac{\alpha\nu^{\alpha}}{t^{\alpha+1}}$	$\nu = 1.5$ $\alpha = 3.0$	$t \in [\nu, \infty)$
Distributions with finite support			
Uniform(a, b)	$\frac{1}{b-a}$	$a = 10.0$ $b = 20.0$	$t \in [a, b]$
Beta(α, β)	$\frac{t^{\alpha-1} \cdot (1-t)^{\beta-1}}{\mathrm{B}(\alpha,\beta)}$	$\alpha = 2.0$ $\beta = 2.0$	$t \in [0, 1]$
BoundedPareto(L, H, α)	$\frac{\alpha L^{\alpha} t^{-\alpha-1}}{1-\left(\frac{L}{H}\right)^{\alpha}}$	$L = 1.0$ $H = 20.0$ $\alpha = 2.1$	$t \in [L, H]$

To get uniform results, we normalize the expected cost of each heuristic by the expected cost of an *omniscient* scheduler, which knows the job execution time t a priori, and thus would make a single request of length $t_1 = t$. Averaging over all possible values of t from the distribution $\mathcal{D}$, the omniscient scheduler has an expected cost:

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

Hence, the normalized ratio will always be greater than or equal to 1, and a smaller ratio means a better result.

We perform the evaluation of the heuristics under two different reservation-based scenarios.

- RESERVATIONONLY (Section V-B): This scenario is based on the *Reserved Instance* pricing scheme available in AWS [3], where the user pays exactly what is requested. Hence, we set $\alpha = 1, \beta = \gamma = 0$. We consider nine probability distributions in this case, six of which have infinite support and the remaining three have finite support. Table I lists these distributions with instantiations of their parameters used in the evaluation.
- NEUROHPC (Section V-C): This scenario is based on executing large jobs on HPC platforms, where the cost, as represented by the total turnaround time of a job, is the sum of its waiting time in the queue and its actual execution time. We set $\beta = 1$ for the execution time and instantiate the waiting time function (α, γ) by curve-fitting the data from Fig. 2b. The probability distribution is derived from the execution traces of the neuroscience application shown in Fig. 1b.

B. Results for RESERVATIONONLY scenario

Table II presents, for each heuristic, the normalized expected cost, i.e., $\tilde{\mathbb{E}}(S)/\mathbb{E}^o$, under different probability distributions. The BRUTE-FORCE heuristic tries $M = 5000$ values of t_1, and both discretization heuristics set the truncation parameter to be $\epsilon = 10^{-7}$ and use $n = 1000$ samples. First, the normalized costs allow us to compare the performance of these heuristics

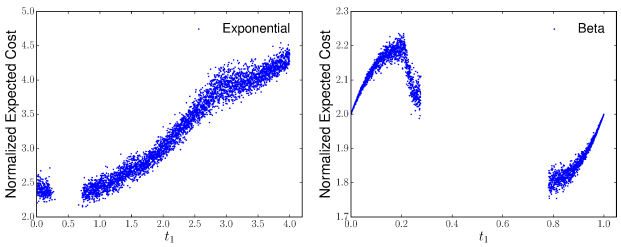


Fig. 3. Monte-Carlo simulations of the normalized costs of the BRUTE-FORCE heuristic with different values of t_1 in the RESERVATIONONLY scenario under exponential and beta distributions.

with that of the omniscient scheduler to access the relative benefits of using *Reserved Instance* (RI) vs. *On-Demand* (OD). Indeed, if the per-hour rate for RI is c_{RI} and the corresponding rate for OD is c_{OD}, it is beneficial to use RI and compute a reservation sequence S, if $c_{\text{RI}} \cdot \tilde{\mathbb{E}}(S) \leq c_{\text{OD}} \cdot \mathbb{E}^o$, that is $\tilde{\mathbb{E}}(S)/\mathbb{E}^o \leq c_{\text{OD}}/c_{\text{RI}}$. In the case of Amazon AWS [3], the rates for the two types of services can differ by a factor of 4, i.e., $c_{\text{OD}}/c_{\text{RI}} = 4$. We can see in the table that the normalized costs of all heuristics satisfy $\tilde{\mathbb{E}}(S)/\mathbb{E}^o < 4$ for all distributions. Overall, the results show the benefit of using reservations over the on-demand approach. We also observe that, compared with other heuristics, BRUTE-FORCE has better performance (see values in the brackets in the table), and this is because it computes a reservation sequence by exploring the properties of the optimal solution (Section III-C).

We now study the BRUTE-FORCE heuristic in more detail. Fig.3 shows the Monte-Carlo simulations of its normalized costs using different values of t_1 in the search interval under two distributions, namely, exponential and beta (the plots for other distributions can be found in the full version of the paper [4]). First, we can see that some values of t_1 can lead to invalid sequences (that are not increasing). These are indicated by the "gaps" in the figure. Moreover, even if a sequence is valid, randomly guessing a t_1 can result in very poor performance compared to the one returned by our heuristic. This shows the importance of finding the "right" initial reservation request. We point out that more efficient algorithms may exist to search for the best t_1, but our BRUTE-FORCE procedure takes just a few seconds to run on an Intel i7 core with $M = 5000$ and $N = 1000$, thus providing a practical solution that is close to the optimal, as demonstrated by the figures.

C. Results for NEUROHPC scenario

We now present the evaluation results for the NEUROHPC scenario while running a real application under the HPC cost model. The probability distribution is generated from the execution traces of a neuroscience application (VBMQA [17], see Fig. 1b). It follows a LogNormal law with parameters ($\nu = 7.1128, \kappa = 0.2039$) obtained by fitting the execution time data to the distribution curve, and this gives a mean of $\mu = 1253.37\text{s} \approx 0.348$ hour and a standard deviation of $\sigma = 258.261\text{s} \approx 0.072$ hour. The average waiting time function is obtained by analyzing the logs from 20 groups of jobs

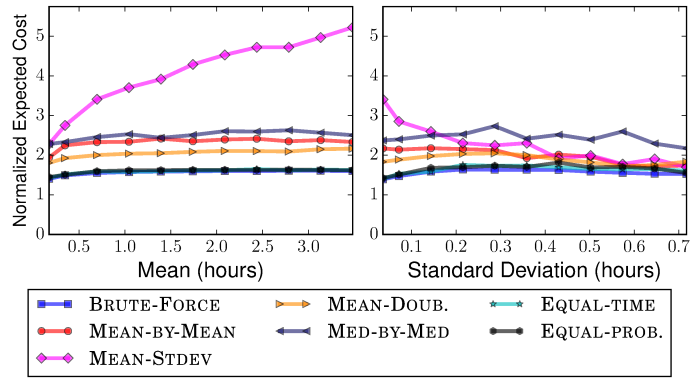


Fig. 4. Normalized expected costs of the different heuristics in the NEUROHPC scenario with different values for the mean (in hours) and standard deviation (in hours) of the LogNormal distribution ($\nu = 7.1128, \kappa = 0.2039$) with $\alpha = 0.95,\ \beta = 1.0,\ \gamma = 1.05$.

run on 409 processors of Intrepid [22] with different requested runtimes (see Fig. 2b). We get an affine waiting time function with parameters ($\alpha = 0.95, \gamma = 3771.84\text{s} \approx 1.05$ hour) obtained also by curve fitting. The execution time parameter is set to $\beta = 1$.

Figure 4 plots the normalized expected costs of different heuristics in this scenario. To evaluate the robustness of the results, we also vary the distribution parameters so that its mean and standard deviation are increased by up to a factor of 10 from their original values[3], i.e., up to $\mu \approx 3.48$ hours and $\sigma \approx 0.72$ hour. We can see from the figure that, regardless of the parameter variations, BRUTE-FORCE and the two discretization-based heuristics (EQUAL-TIME and EQUAL-PROBABILITY) have very close performance, which is significantly better than the performance of the other heuristics. The results are consistent with those observed in Section V-B for the RESERVATIONONLY scenario, and altogether they demonstrate the effectiveness and robustness of the proposed BRUTE-FORCE and discretization schemes for the STOCHASTIC problem.

VI. RELATED WORK

In this section, we review some related work on HPC/cloud resource scheduling and cost models, as well as on stochastic scheduling of jobs with uncertain execution times.

a) HPC resource scheduling: Most schedulers for HPC systems use an iterative repetitive algorithm triggered by state changes, such as new job submission, job starting or ending, or timeout. They use different policies to determine which job should execute when and on what resources. Jobs are usually placed in one or multiple queues with different priorities before being scheduled onto the available resources. For example, the Slurm scheduler [28] uses two queues, one for high-priority jobs and the other for low-priority jobs. A job is placed in a queue based on its resource requirement,

[3]Given a desired mean μ and a standard deviation σ, the LogNormal distribution can be instantiated with parameters $\kappa = \sqrt{\ln((\frac{\sigma}{\mu})^2 + 1)}$ and $\nu = \ln\left(\mu - \frac{\kappa^2}{2}\right)$.

Table II. Normalized expected costs of different heuristics in the RESERVATIONONLY scenario under different distributions. The values in the brackets show the expected costs normalized by those of the BRUTE-FORCE heuristic (best online heuristic).

Distribution	BRUTE-FORCE	MEAN-BY-MEAN	MEAN-STDEV	MEAN-DOUB.	MED-BY-MED	EQUAL-TIME	EQUAL-PROB.
Exponential	2.15	2.36 (1.10)	2.39 (1.11)	2.42 (1.13)	2.83 (1.32)	2.31 (1.07)	2.36 (1.10)
Weibull	2.12	2.76 (1.30)	3.58 (1.69)	3.03 (1.43)	3.05 (1.44)	2.40 (1.13)	2.22 (1.05)
Gamma	2.02	2.26 (1.12)	2.18 (1.08)	2.24 (1.11)	2.51 (1.24)	2.20 (1.09)	2.13 (1.05)
Lognormal	1.85	2.19 (1.19)	2.09 (1.13)	1.95 (1.06)	2.30 (1.24)	1.87 (1.01)	1.93 (1.04)
TruncatedNormal	1.36	1.98 (1.46)	1.83 (1.35)	1.98 (1.46)	2.16 (1.60)	1.38 (1.02)	1.36 (1.00)
Pareto	1.62	1.82 (1.12)	2.18 (1.34)	1.75 (1.08)	2.26 (1.39)	1.71 (1.05)	1.66 (1.03)
Uniform	1.33	2.21 (1.66)	1.90 (1.43)	1.67 (1.26)	2.21 (1.66)	1.33 (1.00)	1.33 (1.00)
Beta	1.75	2.02 (1.15)	2.11 (1.20)	1.98 (1.13)	2.45 (1.40)	1.79 (1.02)	1.80 (1.02)
BoundedPareto	1.80	1.84 (1.02)	2.09 (1.16)	1.83 (1.01)	2.81 (1.56)	2.00 (1.11)	1.91 (1.06)

generally with long-running jobs that require a large amount of resources having higher priorities. Jobs that are kept in the waiting queue for a long period of time could also be upgraded and moved up in the queue. Slurm schedules the jobs from the top of the high-priority queue and moves down. Even though larger jobs (in term of time and space) have higher priorities, generally the lack of resource availability in the system leads to longer wait times. On the other hand, smaller jobs, despite having lower priorities, are usually scheduled quickly thanks to the backfilling algorithms that place them in the unused time slots between successive large jobs.

Some studies (e.g., [18], [21], [27]) have analyzed the impact of scheduling strategies on the performance of applications in large HPC centers. Some of these studies show that the penalty for jobs with longer requested walltimes and/or larger numbers of nodes is higher than that for jobs with shorter elapsed times and smaller numbers of nodes. This is observed, for example, in [27] for the K computer from Riken Advanced Institute for Computational Science. The study shows that, for applications requesting similar computing resources, the wait time generally increases with larger requested processing times and can cause delays of hours for large scientific applications, although it is also dependent on other workloads submitted to the system. Some HPC centers divide the resources into seasons for users to utilize the reserved resources. Users tend to submit more jobs toward the end of a season causing contention at the scheduler level which results in even longer waiting times. The study in [21] presents a trend of the evolution of the workload of HPC systems and the corresponding scheduling policies as we move from monolithic MPI applications to high-throughput and data-intensive jobs. The paper shows that the cost paid in terms of the wait time of applications in the queue has generally increased over the years with less uniform workloads. The study in [18] shows that systems that give each job a partition of the resources for exclusive use and allocate such partitions in the order of job arrivals could suffer from severe fragmentation, leading to low utilization. The authors propose an aggressive backfilling algorithm for dealing with such fragmentation. However, users are still expected to provide accurate runtime estimates. The study shows that over-estimation may lead to a long wait time and possibly excessive CPU quota loss, while under-estimations can lead to job terminations before completion. Some recent schedulers [19] consider the distribution of execution time of the submitted jobs to take their scheduling decision in order to increase their overall utility.

b) Stochastic job scheduling: Many works deal with stochastic job scheduling (e.g., [6], [9], [23]–[25]). Various models [5] have been proposed to model the performance of executing stochastic jobs on computing platforms. For instance, in [16], stochastic jobs are modeled as a DAG of tasks whose execution times and communication times are stochastically independent. In this paper, we model jobs by an execution time following a probability distribution. The authors in [23] propose a model based on resource load in grid systems. Several refinements can be envisioned, such that improving scheduler performances by including distribution features in order to optimize final performance. Also, dealing with heterogeneous nodes increases problem complexity [24]. We refer the reader to the book by Pinedo [20] which contains a comprehensive survey of stochastic scheduling problems, and to the book chapter [12] for a detailed comparison of stochastic task-resource systems.

c) Pricing and reservation schemes for cloud computing: Cloud computing platforms have emerged as another option for executing HPC applications. Job scheduling in the cloud has an even bigger challenge [14], since it needs to deal with highly heterogeneous resources with a wide range of processor configurations, interconnects, virtualization environments, etc.

Different pricing and reservation schemes are also available for users who submit jobs to a cloud service. Several works have been conducted to study these schemes in the cloud, and from a computer science perspective, many of these studies focus on the pricing strategies and service management of platform providers [2], [7], [8], [26]. Some works consider modeling the delays for users [2] and how providers manage the idle resources [8]. The work in [26] studies the pricing practices of Amazon AWS [3] when the price is dynamically adapted to real-time demand and idle resources. In [7], authors provide an analytical model of pricing for reservation-based scheme (used by Amazon AWS) and utilization-based scheme (used by Google GCP [13]). They show that the effective price mainly depends on the variation of platform usage and the competition for customers. Some tools are also provided for users to perform cost evaluation in order to select which type of platform to use. They show that users with high-volatility demand should consider using AWS offers while one should

use GCP in the other case. The experimental results in this paper suggest that, compared with on-demand or utilization-based services, reservation strategies can provide cost-effective options for executing stochastic jobs when there is significant difference in the offered price.

VII. Conclusion

In this paper, we have studied the problem of scheduling stochastic jobs on a reservation-based platform. We have shown the existence of an optimal reservation sequence when the job execution time follows a set of classical distributions, and we have characterized the optimal solution up to the duration of the first reservation. We do not know how to compute this duration analytically, but we have provided an upper bound and a brute-force procedure to generate a solution that is close to the optimal. We have also introduced several heuristics, one based upon discretizing the continuous distribution, and some relying on standard measures, such as the mean, variance and quantiles of the distribution. We have demonstrated the effectiveness of these heuristics via comprehensive simulations conducted using both classical distributions and execution traces of a real neuroscience application.

Future work will include allowing requests with variable amount of resources, hence offering a combination of a reservation time and a number of processors. Another interesting direction is to include checkpoint snapshots at the end of some reservations. We expect the solutions such as the one introduced in this work not to work because of the difficulty of choosing which reservations to checkpoint. Indeed we do not expect the strategy "checkpoint all reservations" to be optimal. Hence the checkpointing approach calls for a complicated trade-off between doing useful work through the reservations and sacrificing some time/budget in order to avoid restarting the job whenever its execution time exceeds the length of the current reservation. The cost will then depend both on the length of the reservation and on a conditional probability based on previous checkpointing decisions.

Acknowledgments: We thank Bennett Landman and his MASI Lab at Vanderbilt for sharing the medical imaging database used to extract the execution time distributions. This research was supported in part by the National Science Foundation grant CCF1719674, Vanderbilt Institutional Fund, and Inria-Vanderbilt associated team *Keystone*. Part of this work was done while Valentin Honoré was visiting Vanderbilt University.

References

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

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

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

[4] G. Aupy, A. Gainaru, V. Honoré, P. Raghavan, Y. Robert, and H. Sun. Reservation Strategies for Stochastic Jobs. Research Report RR-9211, INRIA, 2018.

[5] L.-C. Canon, A. K. W. Chang, Y. Robert, and F. Vivien. Scheduling independent stochastic tasks under deadline and budget constraints. Research Report 9178, INRIA, June 2018.

[6] L.-C. Canon and E. Jeannot. Evaluation and optimization of the robustness of dag schedules in heterogeneous environments. *IEEE Transactions on Parallel and Distributed Systems*, 21(4):532–546, 2010.

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

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

[9] F. Dong, J. Luo, A. Song, and J. Jin. Resource load based stochastic DAGs scheduling mechanism for Grid environment. In *2010 IEEE 12th International Conference on High Performance Computing and Communications (HPCC)*, pages 197–204, Sept 2010.

[10] D. Feitelson. Workload modeling for computer systems performance evaluation. *Version 1.0.3*, pages 1–607, 2014.

[11] L. Friedman and G. H. Glover. Report on a multicenter fMRI quality assurance protocol. *Journal of Magnetic Resonance Imaging*, 23(6):827–839, 2006.

[12] B. Gaujal and J.-M. Vincent. Comparisons of stochastic task-resource systems. In *Introduction to Scheduling*, page Chapter 10. Springer, 2009.

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

[14] A. Gupta, P. Faraboschi, F. Gioachin, L. V. Kale, R. Kaufmann, B. Lee, V. March, D. Milojicic, and C. H. Suen. Evaluating and improving the performance and scheduling of HPC applications in cloud. *IEEE Transactions on Cloud Computing*, 4(3):307–321, July 2016.

[15] 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.

[16] K. Li, X. Tang, B. Veeravalli, and K. Li. Scheduling precedence constrained stochastic tasks on heterogeneous cluster systems. *IEEE Transactions on Computers*, 64(1):191–204, 2015.

[17] A. Mechelli, C. J. Price, K. J. Friston, and J. Ashburner. Voxel-based morphometry of the human brain: methods and applications. *Current Medical Imaging Reviews*, 1:105–113, 2005.

[18] A. W. Mu'alem and D. G. Feitelson. Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling. *IEEE Transactions on Parallel and Distributed Systems*, 12(6):529–543, June 2001.

[19] J. W. Park, A. Tumanov, A. Jiang, M. A. Kozuch, and G. R. Ganger. 3sigma: distribution-based cluster scheduling for runtime uncertainty. In *Proceedings of the Thirteenth EuroSys Conference*, page 2. ACM, 2018.

[20] M. L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer, 3rd edition, 2008.

[21] G. P. Rodrigo Álvarez, P.-O. Östberg, E. Elmroth, K. Antypas, R. Gerber, and L. Ramakrishnan. HPC system lifetime story: Workload characterization and evolutionary analyses on NERSC systems. In *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '15, pages 57–60, New York, NY, USA, 2015. ACM.

[22] W. Tang, Z. Lan, N. Desai, D. Buettner, and Y. Yu. Reducing fragmentation on torus-connected supercomputers. In *Parallel & Distributed Processing Symposium (IPDPS), 2011 IEEE International*, pages 828–839. IEEE, 2011.

[23] X. Tang, K. Li, G. Liao, K. Fang, and F. Wu. A stochastic scheduling algorithm for precedence constrained tasks on grid. *Future Gener. Comput. Syst.*, 27(8):1083–1091, Oct. 2011.

[24] H. Topcuoglu, S. Hariri, and M.-Y. Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE TPDS*, 13(3):260–274, March 2002.

[25] G. Weiss. Turnpike optimality of smith's rule in parallel machines stochastic scheduling. *Math. Oper. Res.*, 17(2):255–270, May 1992.

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

[27] K. Yamamoto and al. The K computer operations: Experiences and statistics. *Procedia Computer Science*, 29:576 – 585, 2014.

[28] A. B. Yoo, M. A. Jette, and M. Grondona. Slurm: Simple linux utility for resource management. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 44–60. Springer, 2003.