On scheduling the checkpoints of exascale applications

Marin Bougeret, Henri Casanova, Mikaël Rabie,
Yves Robert, and Frédéric Vivien

INRIA, École normale supérieure de Lyon, France
Univ. of Hawai'i at Mānoa

Motivation and framework

Framework

- A **very very** large number of processing elements (e.g., 2^{20})
- A platform that may fail (like any realistic platform)
- A very large application to be executed

$\Longrightarrow$ a failure may occur before completion

Questions

- When should we checkpoint the application?
- Should we always use all processors?

Hypotheses and notations

- Overall size of work: $\mathcal{W}$
- Checkpoints of fixed cost: c
  (e.g., write on disk the contents of each processor memory)
- Recovery cost after failure: r
- Homogeneous platform
  (processing elements have same speed and same failure distribution)

State of the art

Applications should be checkpointed periodically

Several proposed values for the period

- Young: $\sqrt{2 \times c \times \text{MTBF}}$ (1st order approximation)
- Daly (1): $\sqrt{2 \times c \times (r + \text{MTBF})}$ (1st order approximation)
- Daly (2): $\eta \times \text{MTBF} - c$, where $\eta = \xi^2 + 1 + \text{Lambert}(-e^{-(2\xi^2+1)})$, and $\xi = \sqrt{\frac{c}{2 \times \text{MTBF}}}$ (higher order approximation)

State of the art

Applications should be checkpointed periodically

Is that the optimal behavior?

Several proposed values for the period

- Young: $\sqrt{2 \times c \times \text{MTBF}}$ (1st order approximation)
- Daly (1): $\sqrt{2 \times c \times (r + \text{MTBF})}$ (1st order approximation)
- Daly (2): $\eta \times \text{MTBF} - c$, where $\eta = \xi^2 + 1 + \text{Lambert}(-e^{-(2\xi^2+1)})$, and $\xi = \sqrt{\frac{c}{2 \times \text{MTBF}}}$ (higher order approximation)

How good are these approximations?

What is the optimal value?

What about failures not following an exponential distribution ?

Presentation outline

Plan

Principle of recursive approach (1)

Notation

- $\mathbb{E}_{opt}(\mathcal{W}, t)$: optimal expectation of makespan, for a work of size $\mathcal{W}$, knowing that the last failure happened t units of time ago.
- $W_1(\mathcal{W}, t)$: size of first chunk, for a work of size $\mathcal{W}$, knowing that the last failure happened t units of time ago.
- $\mathcal{P}_{\text{succ}}(W, t)$: probability that a work of size W is completed before next failure, knowing that the last failure happened t units of time ago.

Underlying hypothesis

- The history of failures does not have any impact, only the time elapsed since the last failure does (renewal process).

Principle of recursive approach (2)

$$\mathbb{E}_{opt}(\mathcal{W}, t) =$$

Principle of recursive approach (2)

$$\mathbb{E}_{opt}(\mathcal{W}, t) = \overbrace{\mathcal{P}_{\text{succ}}(W_1(\mathcal{W}, t) + c, t)}^{\text{Probability of success}} \Big(\overbrace{W_1(\mathcal{W}, t) + c}^{\substack{\text{Time needed}\\ \text{to compute}\\ \text{the 1st chunk}}} + \overbrace{\mathbb{E}_{opt}(\mathcal{W} - W_1(\mathcal{W}, t), t + W_1(\mathcal{W}, t) + c)}^{\substack{\text{Time needed to}\\ \text{compute the remainder}}} \Big)$$

Principle of recursive approach (2)

$$\mathbb{E}_{opt}(\mathcal{W}, t) = \begin{array}{l} \overbrace{\mathcal{P}_{\text{succ}}(W_1(\mathcal{W},t)+c,t)}^{\text{Probability of success}} \left(\overbrace{W_1(\mathcal{W},t)+c}^{\substack{\text{Time needed}\\ \text{to compute}\\ \text{the 1st chunk}}} + \overbrace{\mathbb{E}_{opt}(\mathcal{W}-W_1(\mathcal{W},t), t+W_1(\mathcal{W},t)+c)}^{\substack{\text{Time needed to}\\ \text{compute the remainder}}}\right) \\ + \\ \underbrace{(1-\mathcal{P}_{\text{succ}}(W_1(\mathcal{W},t)+c,t))}_{\text{Probability of failure}} \left(\underbrace{\mathbb{E}_{lost}(W_1(\mathcal{W},t)+c,t)}_{\substack{\text{Time elapsed}\\ \text{before the failure}\\ \text{occured}}} + r + \underbrace{\mathbb{E}_{opt}(\mathcal{W},0)}_{\substack{\text{Time needed}\\ \text{to compute } \mathcal{W}\\ \text{from scratch}}}\right) \end{array}$$

Failures following an exponential distribution

General expression

$$
\begin{aligned}
&\mathbb{E}_{opt}(\mathcal{W}, t) = \\
&\quad \mathcal{P}_{\text{succ}}(W_1(\mathcal{W}, t) + c, t) \\
&\qquad\qquad \times (W_1(\mathcal{W}, t) + c + \mathbb{E}_{opt}(\mathcal{W} - W_1(\mathcal{W}, t), t + W_1(\mathcal{W}, t) + c)) \\
&\quad +(1 - \mathcal{P}_{\text{succ}}(W_1(\mathcal{W}, t) + c, t)) \left(\mathbb{E}_{lost}(W_1(\mathcal{W}, t) + c, t) + r + \mathbb{E}_{opt}(\mathcal{W}, 0)\right)
\end{aligned}
$$

Simplified with memoryless property

$$
\begin{aligned}
&\mathbb{E}_{opt}(\mathcal{W}) = \\
&\quad \mathcal{P}_{\text{succ}}(W_1(\mathcal{W}) + c) \\
&\qquad\qquad \times (W_1(\mathcal{W}) + c + \mathbb{E}_{opt}(\mathcal{W} - W_1(\mathcal{W}))) \\
&\quad +(1 - \mathcal{P}_{\text{succ}}(W_1(\mathcal{W}) + c)) \left(\mathbb{E}_{lost}(W_1(\mathcal{W}) + c) + r + \mathbb{E}_{opt}(\mathcal{W})\right)
\end{aligned}
$$

Remarks

- The first chunk successfully executed will be of size $W_1(\mathcal{W})$
- Whatever the scenario, the size of the chunks that will be executed successfully are known before hand. There are $n_0(\mathcal{W})$ chunks.

Optimal checkpointing policy

$$\mathbb{E}_{opt}(\mathcal{W}) = \sum_{i=1}^{n_0(\mathcal{W})} \left(W_i(\mathcal{W}) + c + \frac{1 - \mathcal{P}_{\text{succ}}(W_i(\mathcal{W}) + c)}{\mathcal{P}_{\text{succ}}(W_i(\mathcal{W}) + c)} (\mathbb{E}_{lost}(W_i(\mathcal{W}) + c) + r) \right)$$

$$\mathbb{E}_{opt}(\mathcal{W}) = \left(\frac{1}{\lambda} + r \right) \sum_{i=1}^{n_0(\mathcal{W})} (\mathrm{e}^{\lambda(W_i(\mathcal{W}) + c)} - 1)$$

Theorem

The expectation of the makespan is minimized when checkpoints are periodic of period $T_{opt} = \frac{1 + Lambert(-e^{-(1+\lambda c)})}{\lambda}$ *and* $n_0(\mathcal{W}) = \frac{\mathcal{W}}{T_{opt}}$, *with* $Lambert(x) e^{Lambert(x)} = x$. *Then,*

$$\mathbb{E}_{opt}(\mathcal{W}) = \frac{\mathrm{e}^{\lambda(T_{opt} + c)} - 1}{T_{opt}} \mathcal{W} \left(\frac{1}{\lambda} + r \right)$$

Approximation and dynamic programming

Idea

- Time discretization: chunk sizes must be a multiple of a quantum u

Dynamic programming solution

$$\mathbb{E}_{opt}(\mathcal{W}, t) = \min_{\substack{W_1 = i.u \\ 1 \leq i \leq \frac{\mathcal{W}}{u}}} \begin{cases} \mathcal{P}_{\text{succ}}(W_1 + c, t)\,(W_1 + c + \mathbb{E}_{opt}(\mathcal{W} - W_1, t + W_1 + c)) \\ +(1 - \mathcal{P}_{\text{succ}}(W_1 + c, t))\,(\mathbb{E}_{lost}(W_1 + c, t) + r + \mathbb{E}_{opt}(\mathcal{W}, 0)) \end{cases}$$

Theorem

We have an algorithm in $O\left(\left(\frac{\mathcal{W}}{u}\right)^3 \left(1 + \frac{c}{u}\right)\right)$ *to compute* $\mathbb{E}_{opt}(\mathcal{W})$.

$\Longrightarrow$ numerical approximations

Numerical application

Some meaningless values: $\mathcal{W} = 40$, $c = r = 0.5$, MTBF=20

Chunk sizes for exponential law

4.45 4.45 4.45 4.45 4.45 4.45 4.45 4.45 4.45

Chunk sizes for Weibull law (with $k = 0.5$)

3.15 4 4.65 5.1 5.45 5.75 5.85 6.1

(If no failure occurs during the execution)

Plan

Motivation

Context

- A **very very** large number m of **identical** processors (same processing speed, same failure distribution)

The questions

- On how many processors ($\leq m$) the application must be executed to minimize the expectation of the makespan?
- Could task duplication decrease the expectation of the makespan?

Hypotheses

Parallelization

- Application is perfectly parallelizable $T_{\text{par}}(p) = \frac{T_{\text{seq}}}{p}$

Duplication

- We have g groups of p processors ($g \times p \leq m$)

Failures

- Follow Exponential distribution of parameter λ

Property

- The failure distribution for a group follows an exponential distribution of parameter $p\lambda$
  $\Longrightarrow$ when no duplication, reuse the one processor solution

About illustration examples

For illustration purposes

- Sequential application divided in 3 chunks and run sequentially on a single processor.
- Each "row" on the Gantt charts should be viewed as a group of p processors

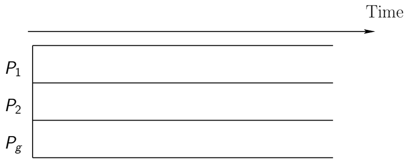
Legend

Failed attempt

Successful attempt

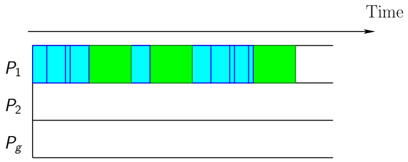
Most naive approach

Principle: no synchronization between groups



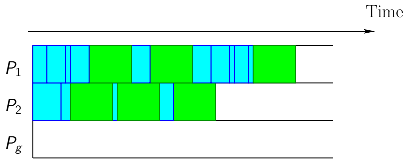
Most naive approach

Principle: no synchronization between groups



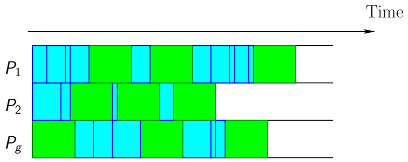
Most naive approach

Principle: no synchronization between groups



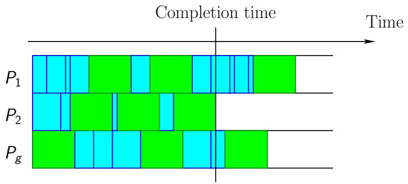
Most naive approach

Principle: no synchronization between groups



Most naive approach

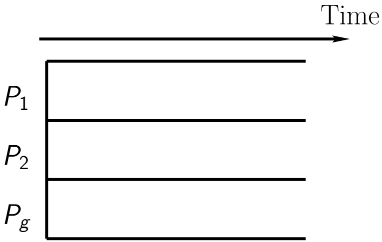
Principle: no synchronization between groups



- To compute the completion time: need the distribution of the completion time on a single processor
- Naive, inefficient, and no analytical solution

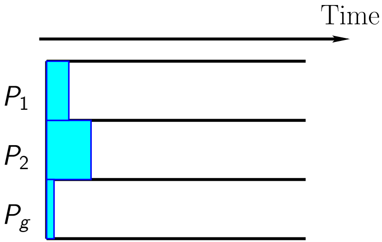
Synchronized restart after failure

Principle: if the first attempt to execute a chunk fails on all groups, they all retry, simultaneously starting after the last failure



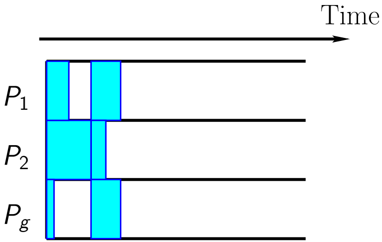
Synchronized restart after failure

Principle: if the first attempt to execute a chunk fails on all groups, they all retry, simultaneously starting after the last failure



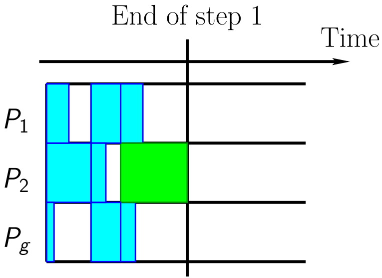
Synchronized restart after failure

Principle: if the first attempt to execute a chunk fails on all groups, they all retry, simultaneously starting after the last failure



Synchronized restart after failure

Principle: if the first attempt to execute a chunk fails on all groups, they all retry, simultaneously starting after the last failure



Synchronized restart after failure: expectation

Probability that at least one *group* runs at least a time t before failing:

$\Pr(X^{(g)} \geq t) = \Pr\left(\max_{i \leq g}(X_i) \geq t\right)$.

Proposition: distribution law

The distribution law of the system is:

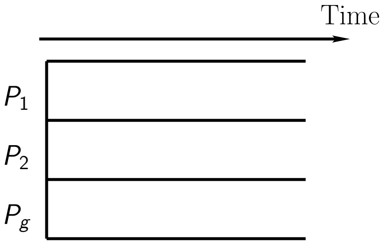
$$d\Pr(X^{(g)} = u) = g\Pr(X_1 < u)^{g-1} d\Pr(X_1 = u)$$

Where X_1 is the random variable for a group, and g the number of duplications.

Expectation of makespan: previous distribution should be substituted in the recursion formula...

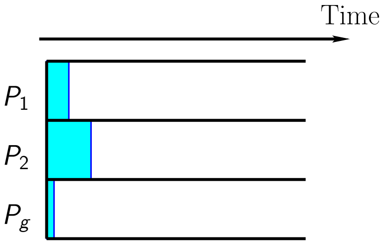
Non-synchronized restarts

Principle: each group re-tries to execute the chunk as soon as possible after each of its failures, as long as no group succeeds



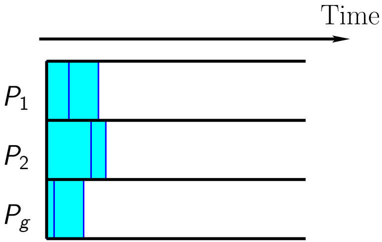
Non-synchronized restarts

Principle: each group re-tries to execute the chunk as soon as possible after each of its failures, as long as no group succeeds



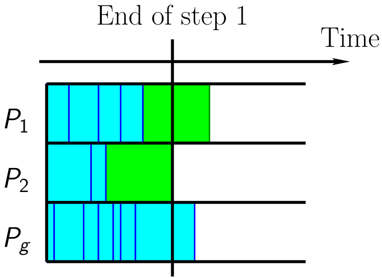
Non-synchronized restarts

Principle: each group re-tries to execute the chunk as soon as possible after each of its failures, as long as no group succeeds



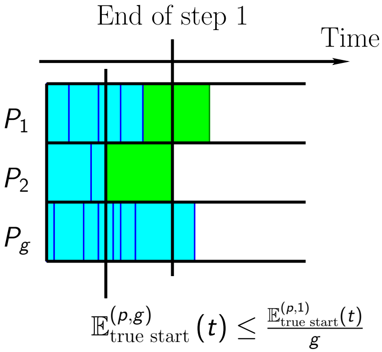
Non-synchronized restarts

Principle: each group re-tries to execute the chunk as soon as possible after each of its failures, as long as no group succeeds



Non-synchronized restarts

Principle: each group re-tries to execute the chunk as soon as possible after each of its failures, as long as no group succeeds



Majoration of the expectation of the makespan

Expectation of the makespan for a single chunk of size W

$$\mathbb{E}^{(p,g)}(W) = \mathbb{E}^{(p,g)}_{\text{true start}}\left(\frac{W}{p}+c, r\right)+\frac{W}{p}+c \leq \frac{\mathbb{E}^{(p,1)}_{\text{true start}}\left(\frac{W}{p}+c, r\right)}{g}+\frac{W}{p}+c$$

Over-approximation

$$\mathbb{E}^{(p,g)}(W) = \frac{\mathbb{E}^{(p,1)}_{\text{true start}}\left(\frac{W}{p}+c, r\right)}{g}+\frac{W}{p}+c$$

Expectation for the overall work

$$\mathbb{E}^{(p,g)}(\mathcal{W}) = \sum_{i=1}^{n_0(\mathcal{W})}\left(\frac{\mathbb{E}^{(p,1)}_{\text{true start}}\left(\frac{W_i(\mathcal{W})}{p}+c, r\right)}{g}+\frac{W_i(\mathcal{W})}{p}+c\right)$$

Computing $\mathbb{E}^{(p,1)}_{\text{true start}}(t, r)$

Random variables

- X_i: time elapsed between the $(i-1)$-th and i-th failures
- N such that: $X_N \geq t$, $X_1 < t$, ..., $X_{N-1} < t$

$$\begin{aligned}
\mathbb{E}^{(p,1)}_{\text{true start}}(t, r) &= \mathbb{E}(X_1 + r + X_2 + r + \ldots + X_{N-1} + r) \\
&= \mathbb{E}\left(\left(\sum_{i=1}^{N} X_i\right) + (N-1)r - X_N\right) \\
&= \mathbb{E}\left(\sum_{i=1}^{N} X_i\right) + (\mathbb{E}(N) - 1)r - \mathbb{E}(X_N) \\
&= \mathbb{E}(X_1)\,\mathbb{E}(N) + (\mathbb{E}(N) - 1)r - \mathbb{E}(X_N) \quad \text{(Wald)}
\end{aligned}$$

As $\mathbb{E}(N) = e^{p\lambda t}$, $\mathbb{E}(X_1) = \frac{1}{p\lambda}$, and $\mathbb{E}(X_N) = \frac{1}{p\lambda} + t$, we find:

$$\mathbb{E}^{(p,1)}_{\text{true start}}(t, r) = \frac{1}{p\lambda} e^{p\lambda t} + \left(e^{p\lambda t} - 1\right) r - \frac{1}{p\lambda} - t.$$

"Best" solution

Theorem

The best policy is to have periodic checkpoints of period T such that

$$T_{opt} = \min\left\{T_{cand}, \frac{\mathcal{W}}{p}\right\} \textit{ with}$$

$$\left(T_{cand} - \frac{1}{p\lambda}\right) e^{p\lambda(T_{cand}+c)}(1+p\lambda r) = (g-1)c - r - \frac{1}{p\lambda}.$$

The expectation of the makespan is then:

$$\mathbb{E}^{(p,g)}(\mathcal{W}) = \frac{\mathcal{W}}{\lambda p^2 g T_{opt}} \left(\begin{array}{l} e^{p\lambda(T_{opt}+c)} + p(g-1)\lambda(T_{opt}+c) \\ +p\lambda r\left(e^{p\lambda(T_{opt}+c)} - 1\right) - 1 \end{array} \right)$$

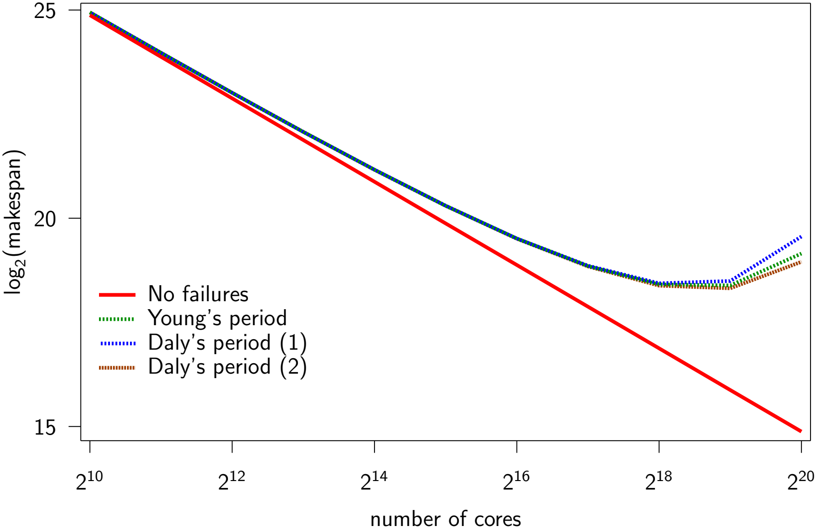
Best = optimal for the over-approximation of the expectation of makespan

Plan

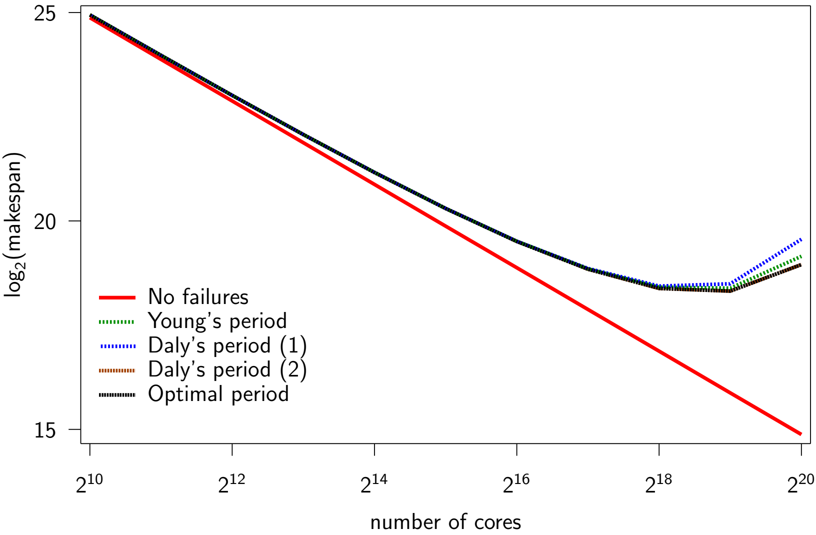
Simulation settings

- $c = 5$ minutes
- $r = 5$ minutes
- $\mathcal{W} = 1000$ years
- $m = 2^{20}$ cores
- Mean Time Between Failures $= 1$, 10, or 100 years

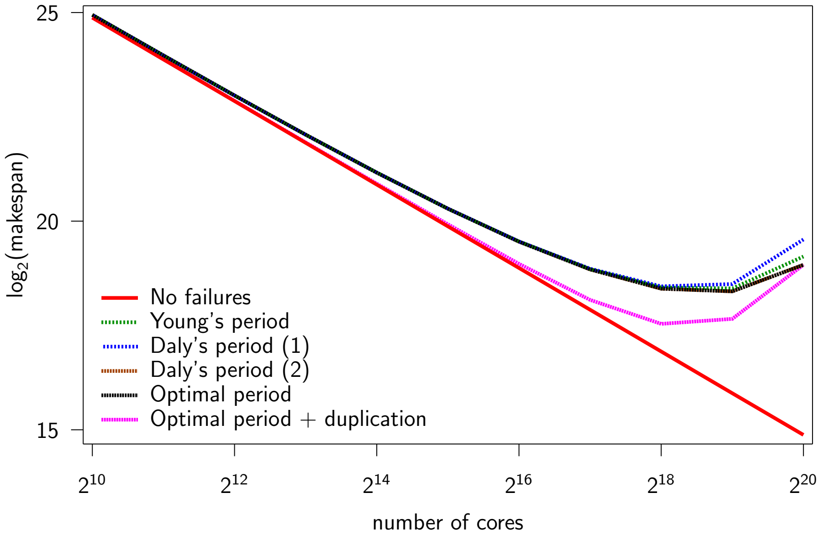
Exponential distribution (MTBF = 10 years)



Exponential distribution (MTBF = 10 years)



Exponential distribution (MTBF = 10 years)



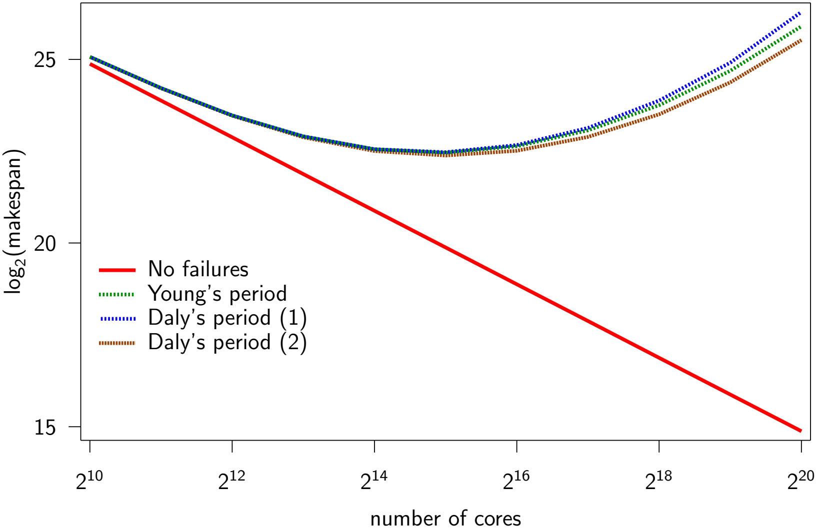
Impact of duplication

MTBF = 10 years. Best makespan (without duplication) reached using 2^{19} cores

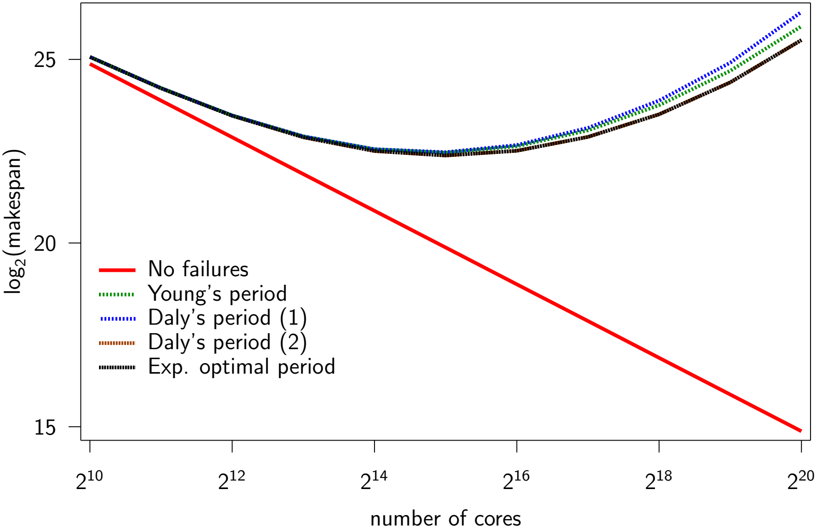
	Without Duplication	With Duplication
Number of cores	2^{19}	2×2^{18}
Average makespan	344,493	206,718

Compared to a full parallelization, a duplication using the same number of processor leads to a gain of 25.6% (on average).

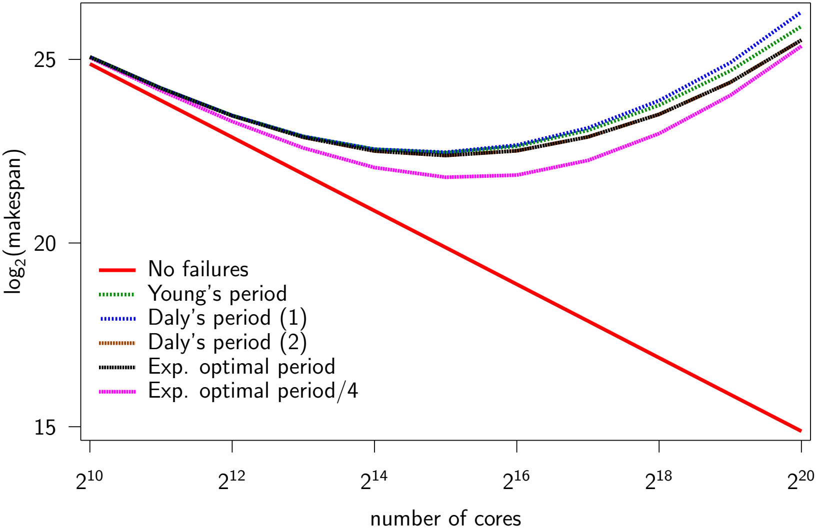
Weibull distribution (k=0.5, MTBF = 10 years)



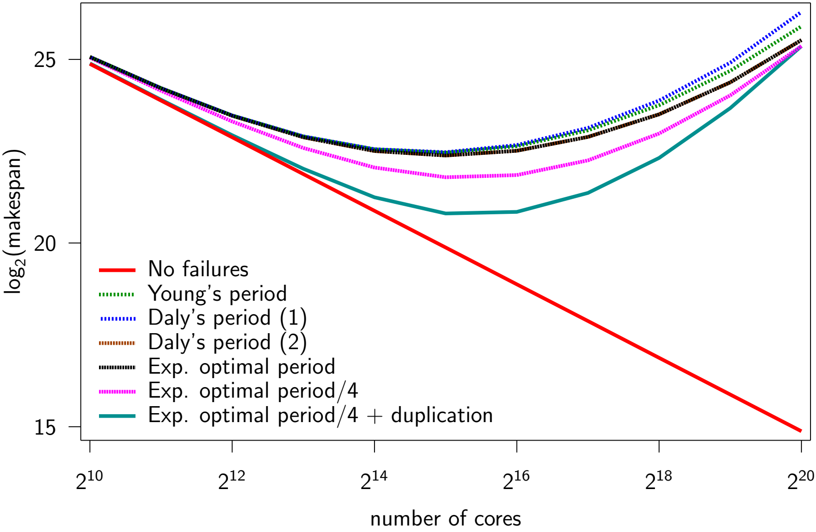
Weibull distribution (k=0.5, MTBF = 10 years)



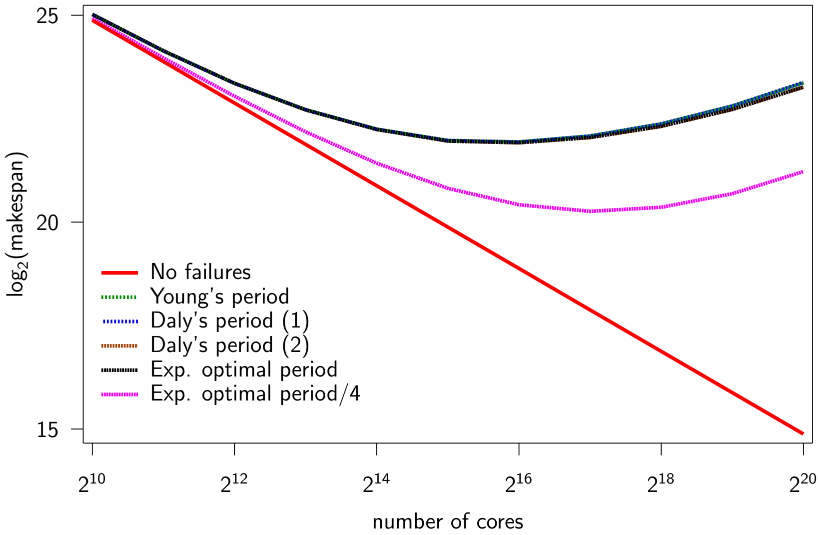
Weibull distribution (k=0.5, MTBF = 10 years)



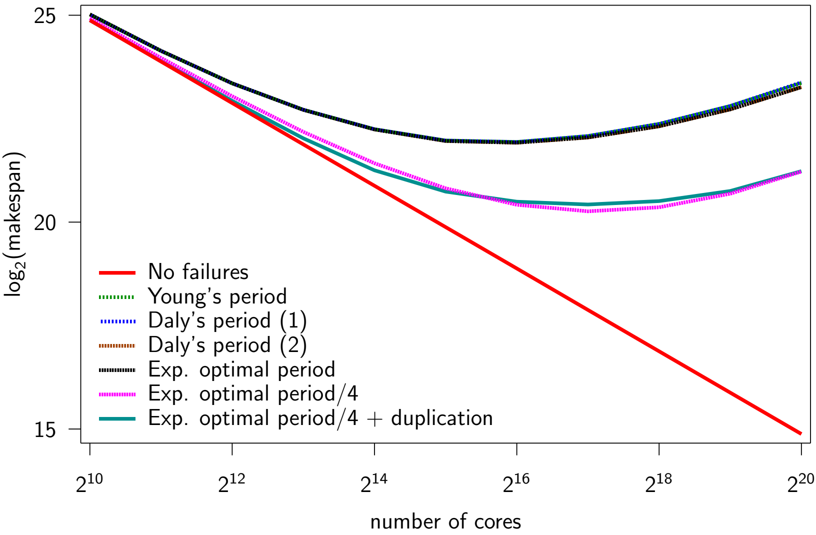
Weibull distribution (k=0.5, MTBF = 10 years)



Weibull distribution (k=0.5, MTBF = 1000 years)



Weibull distribution (k=0.5, MTBF = 1000 years)



Plan

Conclusions

- (Yet another) clean proof for the optimal checkpointing for failures following an exponential distribution
- A duplication scheme that can (almost) be analytically optimized
- When the platform is sufficiently large, the checkpointing cost sufficiently expensive, or the failures frequent enough, one should limit the application parallelism and duplicate tasks
- For Weibull distributions, checkpointing intervals defined for exponential laws of same MTBF are suboptimal, no existing formula delivers good performance

Perspectives

- What is the optimal period for a Weibull distribution?

- What should be the checkpointing policy for a Weibull distribution ?