Multi-criteria Checkpointing Strategies: Response-time versus Resource Utilization

Aurélien Bouteiller[1],
Franck Cappello[2],
Jack Dongarra[1], Amina Guermouche[3],
Thomas Hérault[1], Yves Robert[1,4],

1. University of Tennessee Knoxville, USA
2. INRIA & University of Illinois at Urbana Champaign, USA
3. Université de Versailles Saint Quentin, France
4. Ecole Normale Supérieure de Lyon & INRIA, France

June 13^{th}, 2013

Introduction

- **Very very** large number of processing elements (e.g., 2^{20})
  $\Longrightarrow$ Probability of failures dramatically increases
- Large application to be executed on whole platform
  $\Longrightarrow$ Failure(s) will most likely occur before completion!
- Resilience provided through checkpointing
  1. Coordinated checpkointing protocols
     - ☹ I/O overhead
  2. Uncoordinated checkpointing protocols with message logging

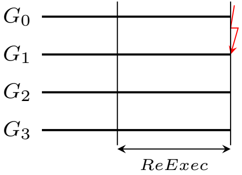
  $\rightarrow$ Hierarchical checkpointing protocols

Introduction

Hierarchical protocols:

☺ A subset of processes roll back

- Overlap recovery and normal execution
  - Tightly coupled applications:
    - ☹ Non-rolled back processes have to wait
- Execute another application during recovery
  - The failed group recovers
  - The non failed groups load and execute another application
    - ☺ Improve platform efficiency



Introduction

1. Application-oriented scenario:
   - Non failed processes wait for the recovering ones
   - The application is executed on $G + 1$ groups

2. Platform-oriented scenario:
   - The application is executed on G groups
   - A spare group is used for recovery
   - The G groups are used to execute another application while the spare group is recovering

Framework

- Periodic checkpointing policies (of period T)
- Independent and identically distributed failures
- Platform failure inter-arrival time: μ
- Tightly-coupled application:

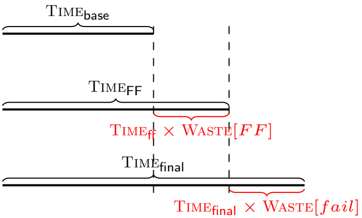
  progress $\Leftrightarrow$ all processors available
- First-order approximation: at most one failure within a period

Waste: fraction of time not spent for useful computations

1. Application waste: fraction of time the processes do not execute the application
2. Platform waste: fraction of time the resources are not used to perform useful work

Waste

- $\textsc{Time}_{\mathsf{base}}$: application base time
- $\textsc{Time}_{\mathsf{FF}}$: with periodic checkpoints but failure-free
- $\textsc{Time}_{\mathsf{final}}$: expectation of time with failures



$$(1 - \textsc{Waste}[FF])\textsc{Time}_{\mathsf{FF}} = \textsc{Time}_{\mathsf{base}}$$

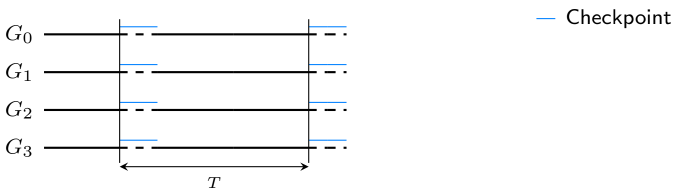
$$(1 - \textsc{Waste}[fail])\textsc{Time}_{\mathsf{final}} = \textsc{Time}_{\mathsf{FF}}$$

$$\textsc{Waste} = \frac{\textsc{Time}_{\mathsf{final}} - \textsc{Time}_{\mathsf{base}}}{\textsc{Time}_{\mathsf{final}}}$$

$$\textsc{Waste} = 1 - (1 - \textsc{Waste}[FF])(1 - \textsc{Waste}[fail])$$

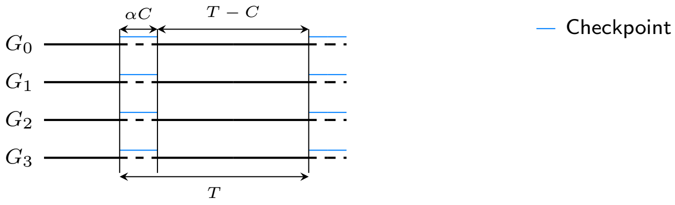
Outline

Checkpoint overhead



- During a checpkoint C, αC work is done $(0 \leq \alpha \leq 1)$

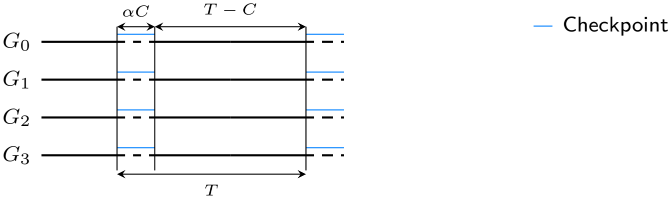
Checkpoint overhead



- During a checpkoint C, αC work is done $(0 \leq \alpha \leq 1)$
- The amount of computation executed in a Period T:

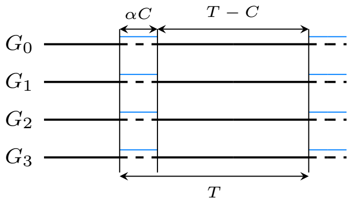
  $Work = \quad \alpha C + T - C$

Message logging overhead



- During a checpkoint C, αC work is done $(0 \leq \alpha \leq 1)$
- The amount of computation executed in a Period T:
  $Work = \quad \alpha C + T - C$

Message logging overhead



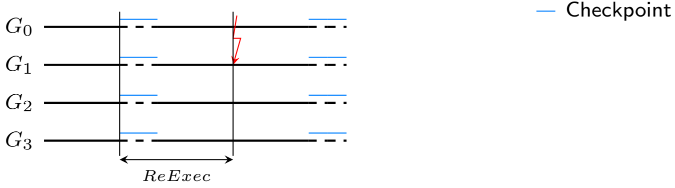
— Checkpoint

- During a checpkoint C, αC work is done ($0 \leq \alpha \leq 1$)
- The amount of computation executed in a Period T:
  $Work = \quad \alpha C + T - C$
- Message logging slows down the execution with a factor λ ($0 < \lambda < 1$):
  The amount of computation executed in a Period T: $\lambda Work$

Message logging overhead

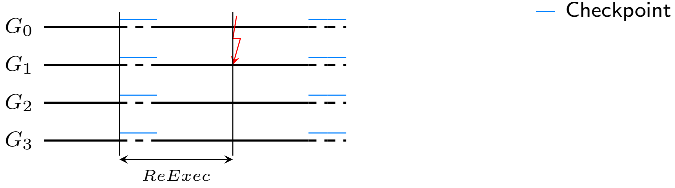
- During a checpkoint C, αC work is done $(0 \leq \alpha \leq 1)$
- The amount of computation executed in a Period T: $\lambda Work = \lambda(\alpha C + T - C)$

Message logging overhead



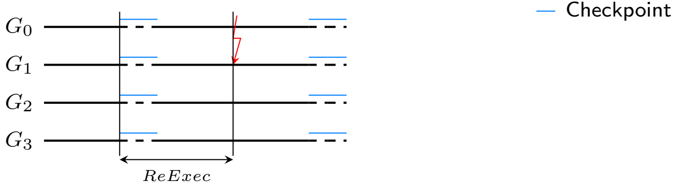
- During a checpkoint C, αC work is done $(0 \leq \alpha \leq 1)$
- The amount of computation executed in a Period T: $\lambda Work = \lambda(\alpha C + T - C)$

Message logging overhead



- During a checpkoint C, αC work is done $(0 \leq \alpha \leq 1)$
- The amount of computation executed in a Period T: $\lambda Work = \lambda(\alpha C + T - C)$
- The amount of work to re-execute is $ReExec$

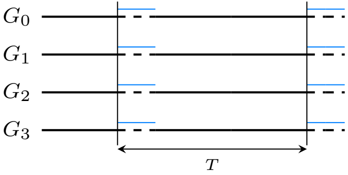
Message logging overhead



- During a checpkoint C, αC work is done $(0 \leq \alpha \leq 1)$
- The amount of computation executed in a Period T: $\lambda Work = \lambda(\alpha C + T - C)$
- The amount of work to re-execute is $ReExec$
- Message logging speeds up the re-execution with a factor ρ $(1 < \rho < 2)$: It takes $\dfrac{ReExec}{\rho}$ to execute $ReExec$

Outline

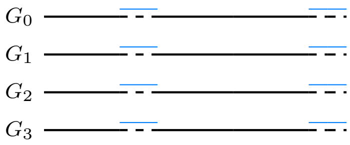
Waste in the absence of failures



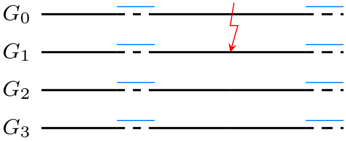
- T work should be done
- $\lambda Work = \lambda(T - C + \alpha C)$ is done

$$\textsc{Waste}[ff] = \frac{T - \lambda Work}{T}$$

Waste in case of failures

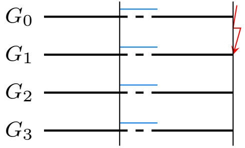


Waste in case of failures



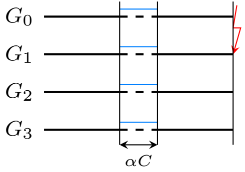
1. Failure during work:

Waste in case of failures



1. Failure during work:

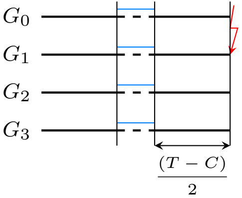
Waste in case of failures



1. Failure during work:

$$ReExec1 = \alpha C$$

Waste in case of failures

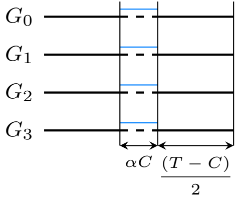


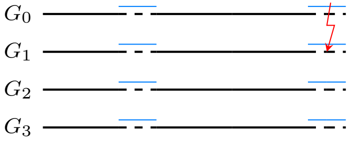
1. Failure during work:

$$ReExec1 = \quad \alpha C + \frac{T-C}{2}$$

Probability: $\dfrac{T-C}{T}$

Waste in case of failures





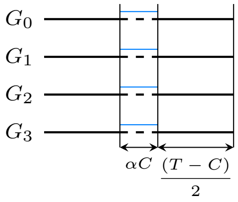
1. Failure during work:

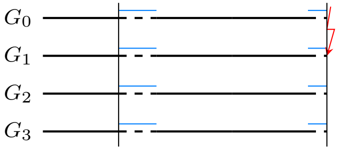
$$ReExec1 = \quad \alpha C + \frac{T-C}{2}$$

Probability: $\dfrac{T-C}{T}$

2. Failure during checkpoint:

Waste in case of failures





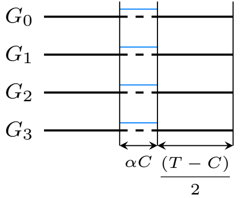
1. Failure during work:

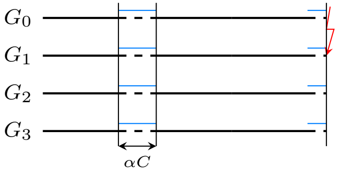
$$ReExec1 = \quad \alpha C + \frac{T-C}{2}$$

Probability: $\frac{T-C}{T}$

2. Failure during checkpoint:

Waste in case of failures





1. Failure during work:

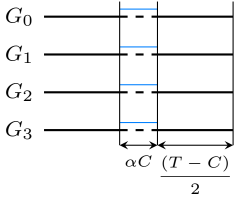
$$ReExec1 = \quad \alpha C + \frac{T-C}{2}$$

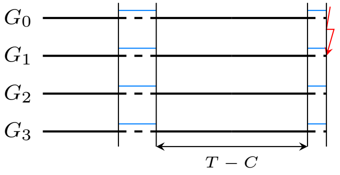
Probability: $\frac{T-C}{T}$

2. Failure during checkpoint:

$$ReExec2 = \quad \alpha C$$

Waste in case of failures





1. Failure during work:

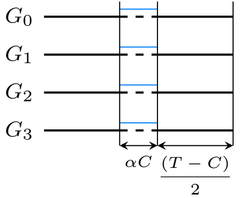
$$ReExec1 = \quad \alpha C + \frac{T-C}{2}$$

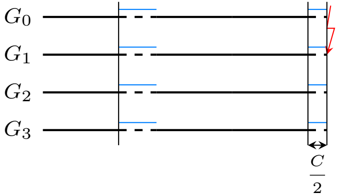
Probability: $\frac{T-C}{T}$

2. Failure during checkpoint:

$$ReExec2 = \quad \alpha C + T - C$$

Waste in case of failures





1. Failure during work:

$$ReExec1 = \quad \alpha C + \frac{T-C}{2}$$

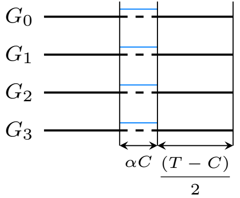
Probability: $\frac{T-C}{T}$

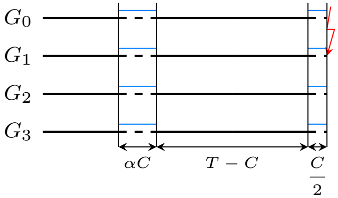
2. Failure during checkpoint:

$$ReExec2 = \quad \alpha C + T - C + \frac{C}{2}$$

Probability: $\frac{C}{T}$

Waste in case of failures





1. Failure during work:

$$ReExec1 = \alpha C + \frac{T-C}{2}$$

Probability: $\frac{T-C}{T}$

2. Failure during checkpoint:

$$ReExec2 = \alpha C + T - C + \frac{C}{2}$$

Probability: $\frac{C}{T}$

$$\textsc{Waste}[fail] = \frac{1}{\mu}[D + R + \frac{T-C}{T} \times \frac{ReExec1}{\rho} + \frac{T}{C} \times \frac{ReExec2}{\rho}]$$

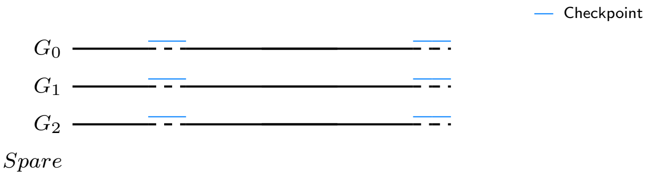
Outline

Framework

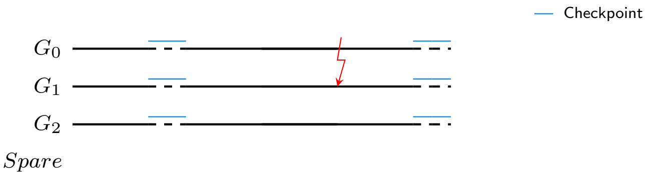
- $G+1$ available groups
- G groups used to run the application
- A spare group used for the rollback
  - Longer time needed to compute the same amount of work
  - Longer checkpoint duration (by a factor of $\frac{G+1}{G}$)

$$\textsc{Waste} = \frac{1}{G+1} + \frac{G}{G+1}(\textsc{Waste}[ff] + \textsc{Waste}[fail] - \textsc{Waste}[ff]\textsc{Waste}[fail])$$

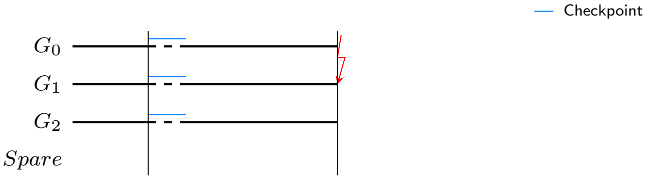
Waste in case of failures



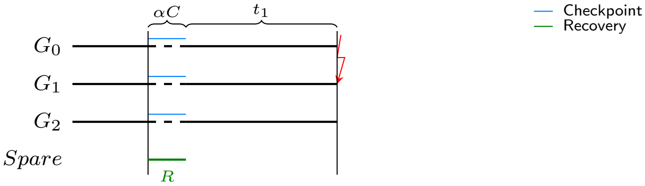
Waste in case of failures



Waste in case of failures

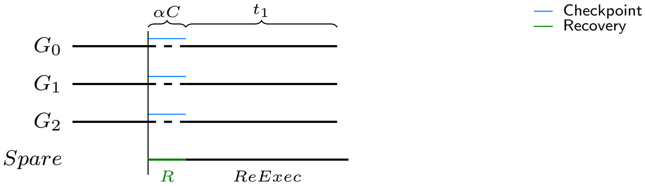


Waste in case of failures



- The spare group restarts from the checkpoint

Waste in case of failures



- The spare group restarts from the checkpoint
- $ReExec = \alpha C + t_1$

Waste in case of failures

- The non failed groups checkpoint

Waste in case of failures

- They load the other application from its checkpoint

Waste in case of failures

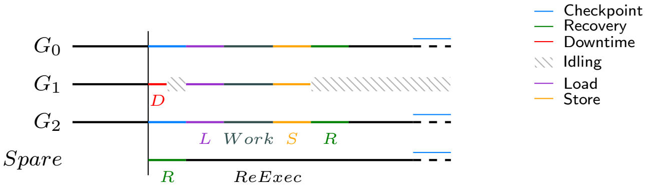
Waste in case of failures

- They store the work they did

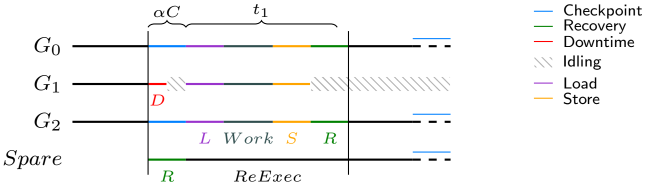
Waste in case of failures

- They reload the other application
- The faulty group becomes the spare group

Waste in case of failures

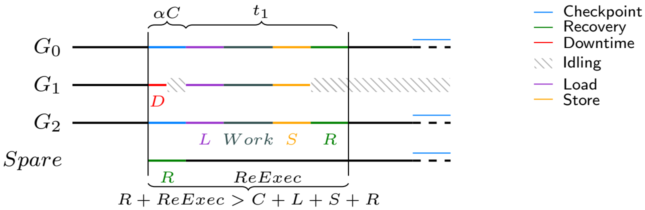


Platform scenario



$$ReExec = \frac{\alpha C + t_1}{\rho}$$

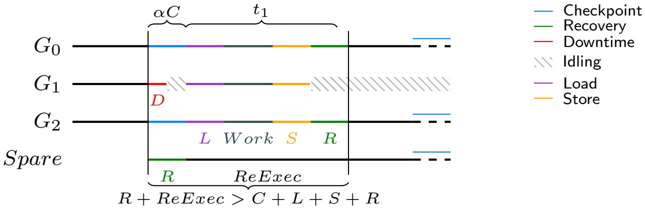
Platform scenario



$$ReExec = \frac{\alpha C + t_1}{\rho}$$

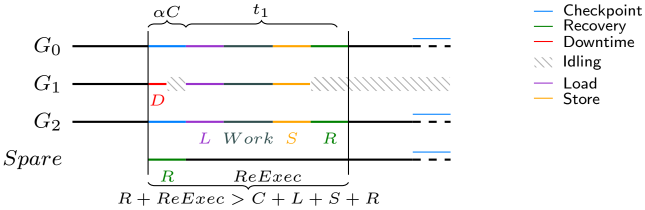
$$R + ReExec > C + L + S + R$$

Platform scenario



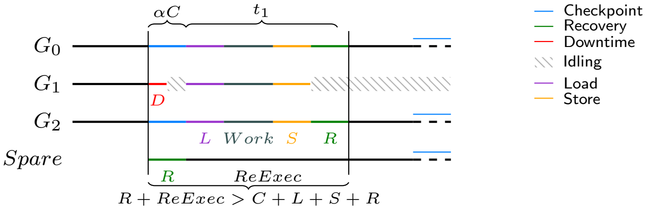
$$
\begin{aligned}
ReExec &= \frac{\alpha C + t_1}{\rho} \\
R + ReExec &> C + L + S + R \\
ReExec &> C + L + S
\end{aligned}
$$

Platform scenario



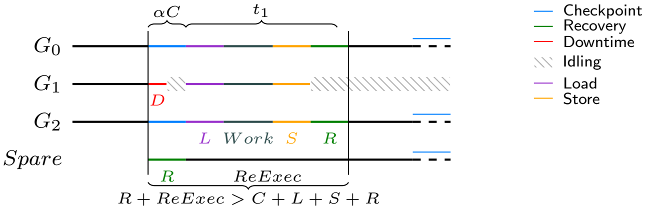
$$
\begin{aligned}
ReExec &= \frac{\alpha C + t_1}{\rho} \\
R + ReExec &> C + L + S + R \\
ReExec &> C + L + S \\
t_1 &> \rho(C + L + S) - \alpha C = Z
\end{aligned}
$$

Platform scenario



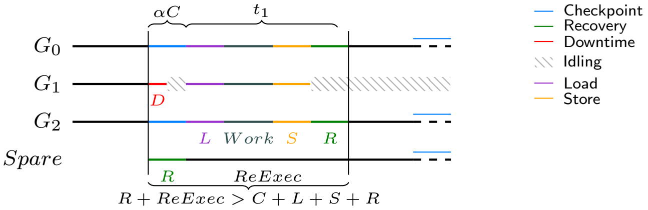
$$\begin{aligned}
ReExec &= \frac{\alpha C + t_1}{\rho} \\
R + ReExec &> C + L + S + R \\
ReExec &> C + L + S \\
t_1 &> \rho(C + L + S) - \alpha C = Z \\
t_1 &> Z
\end{aligned}$$

Platform scenario



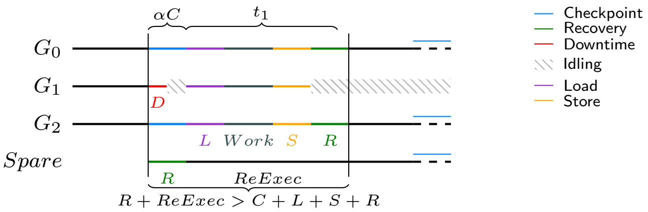
- $t_1 > Z$ with a probability $\dfrac{T - Z}{T}$
  The G groups loose $L + C + S + R = X$

Platform scenario



- $t_1 > Z$ with a probability $\dfrac{T-Z}{T}$
  The G groups loose $L + C + S + R = X$
- $t_1 <= Z$ with a probability $\dfrac{Z}{T}$:
  - The G groups loose $R + ReExec$
  - The expectation of t_1 is $\dfrac{Z}{2}$

Platform scenario



- $t_1 > Z$ with a probability $\dfrac{T-Z}{T}$

  The G groups loose $L + C + S + R = X$
- $t_1 <= Z$ with a probability $\dfrac{Z}{T}$:
  - The G groups loose $R + ReExec$
  - The expectation of t_1 is $\dfrac{Z}{2}$

$$\textsc{Waste}[fail] = \frac{1}{\mu}[\frac{T-Z}{T} \times X + \frac{Z}{T} \times (R + \frac{X-R}{2} + \frac{\alpha C}{2\rho}]$$

Outline

1. Protocol overhead
2. Application scenario
3. Platform scenario
4. Instantiating the model
5. Simulations

Model instantiation

1. Applications
   - 2D-stencil
   - Matrix product
2. Platforms
   - K-Computer
   - ExaScaleFat

Impact of message logging on checkpoint size

- Inter-groups messages logged continuously
- Checkpoint size increases with amount of work executed before a checkpoint
- C_0: Checkpoint size of a group without message logging

$$C = C_0(1 + \beta \textsc{Work}) \Leftrightarrow \beta = \frac{C - C_0}{C_0 \textsc{Work}}$$

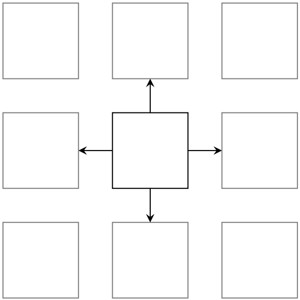
$$\textsc{Work} = \lambda(T - (1 - \alpha)GC)$$

$$C = \frac{C_0(1 + \beta\lambda T)}{1 + GC_0\beta\lambda(1 - \alpha)}$$

Computing β for Stencil-2D

$$C = C_0 + Logged\_Msg = C_0(1 + \beta\textsc{Work})$$

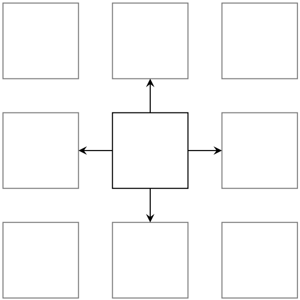
- $C_0 = \dfrac{Mem}{G}$
- Real matrix $n \times n$
- $Mem = 8n^2$



Computing β for Stencil-2D

$$C = C_0 + Logged\_Msg = C_0(1 + \beta \textsc{Work})$$

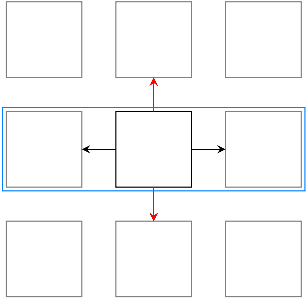
- $C_0 = \dfrac{Mem}{G}$
- s_p: speed of the process
- Block update: 9 floating points operations
- Each process holds a block of size b
- $Work = \dfrac{9b^2}{\mathsf{s}_p}$



Computing β for Stencil-2D

$$C = C_0 + Logged\_Msg = C_0(1 + \beta \textsc{Work})$$

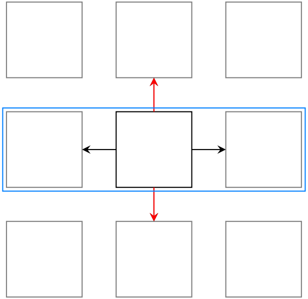
- $C_0 = \frac{Mem}{G}$
- $Work = \frac{9b^2}{s_p}$
- 1 group = 1 line
- Each process sends a block to its 4 neighbors
    - 2 out of the 4 messages are logged



Computing β for Stencil-2D

$$C = C_0 + Logged\_Msg = C_0(1 + \beta\textsc{Work})$$

- $C_0 = \dfrac{Mem}{G}$
- $Work = \dfrac{9b^2}{\mathsf{s}_p}$
    - 2 out of the 4 messages are logged
    - $\beta = \dfrac{2\mathsf{s}_p}{9b^3}$



Outline

Simulation parameters

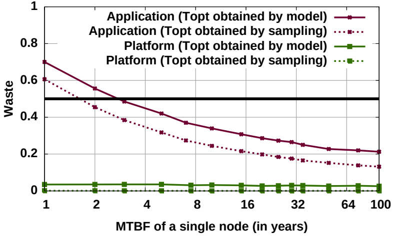
- Failure distribution: Weibull, $k = 0.7$
- Failure free execution on each process: 4 days
- Time-out: 1 year
- No assumption on failures
- $\alpha = 0.3$, $\rho = 1.5$, $\lambda = 0.98$
- Each point: average over 20 randomly generated instances
- Computed period and best period:

$\rightarrow$ Generate 480 periods in the neighborhood of the period from the model

$\rightarrow$ Numerically evaluate the best one through simulations

Waste comparison

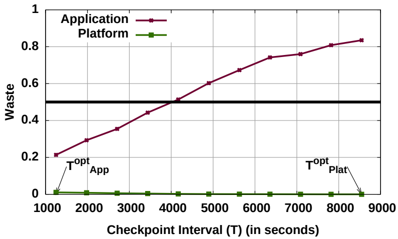
- Solid line: Computed period
- Dotted line: Best Period



Matrix-Product waste on K-Computer

Checkpointing period impact

- T^{opt}_{App}: Application-scenario optimal period
- T^{opt}_{Plat}: Platform-scenario optimal period



2D-Stencil waste on ExaScale Fat for an MTBF of 20 years

Outline

Conclusion and future work

- Overlap the idle time created by recovery periods
- Analytical model
  1. Application scenario:
     - No overlap
     - $G + 1$ groups
  2. Platform scenario:
     - Overlap with another application
     - G groups $+ 1$ spare group
- Improve platform efficiency
  - A spare group
  - Application optimal period
- Future work:
  - Energy efficiency