Unified Model for Assessing Checkpointing Protocols at Extreme-Scale

George Bosilca[1], Aurélien Bouteiller[1], Elisabeth Brunet[2], Franck Cappello[3], Jack Dongarra[1], Amina Guermouche[4], Thomas Hérault[1], Yves Robert[1,4], Frédéric Vivien[4], and Dounia Zaidouni[4]

1. University of Tennessee Knoxville, USA
2. Telecom SudParis, France
3. INRIA & University of Illinois at Urbana Champaign, USA
4. Ecole Normale Supérieure de Lyon & INRIA, France

May 30, 2012

Motivation

Framework

- **Very very** large number of processing elements (e.g., 2^{20})
- Failure-prone platform (like any realistic platform)
- Large application to be executed on the whole platform

$\Longrightarrow$ Failure(s) will certainly occur before completion!

- Resilience provided through checkpointing

Outline

Outline

Which checkpointing protocol to use?

Coordinated checkpointing

- ☺ No risk of cascading rollbacks
- ☺ No need to log messages
- ☹ All processors need to rollback
- ☹ *Rumor: does not scale to very large platforms*

Hierarchical checkpointing

- ☹ Need to log inter-groups messages
  - Slowdowns failure-free execution
  - Increases checkpoint size/time
- ☺ Only processors from failed group need to rollback
- ☺ Faster re-execution with logged messages
- ☺ *Rumor: scales well to very large platforms*

Framework

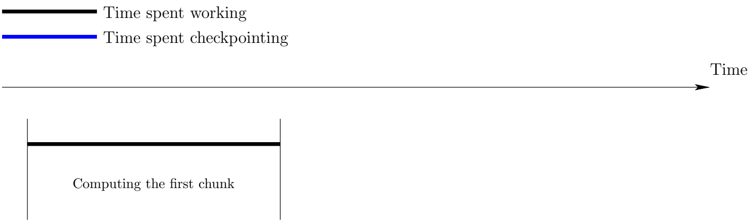
- Periodic checkpointing policies (of period T)
- Independent and identically distributed failures
- Platform failure inter-arrival time: μ
- Tightly-coupled application: progress $\Leftrightarrow$ all processors available
- First-order approximation: at most one failure within a period

Waste: fraction of time not spent for useful computations

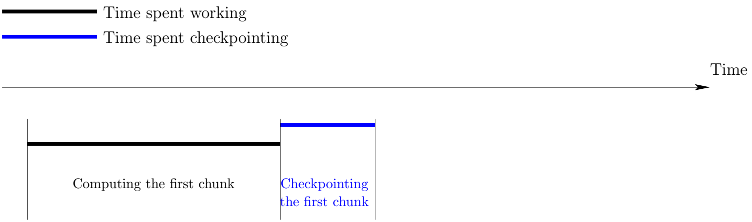
Checkpointing cost

Checkpointing cost

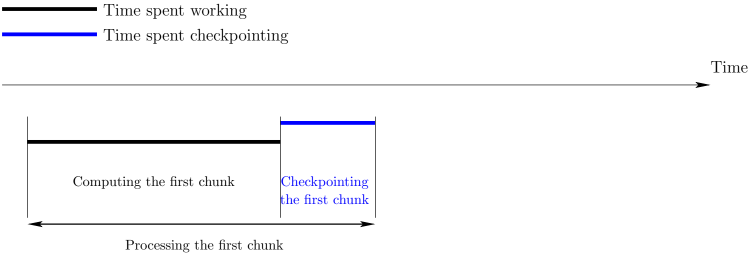
Checkpointing cost



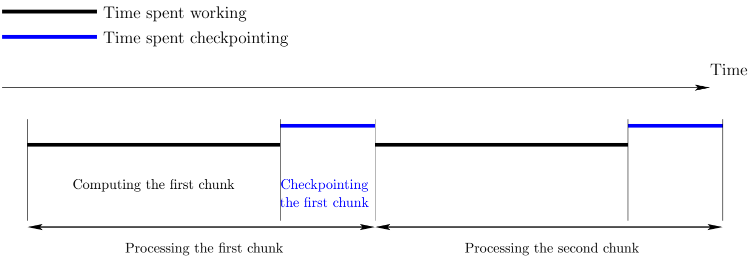
Checkpointing cost



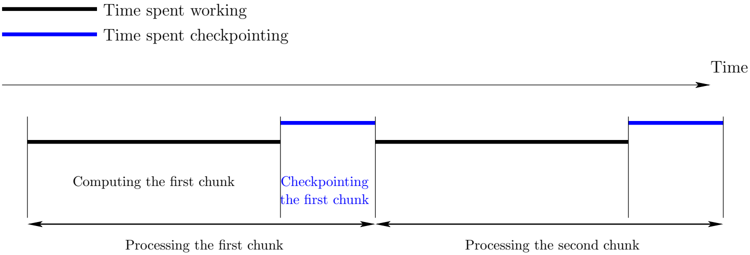
Checkpointing cost



Checkpointing cost

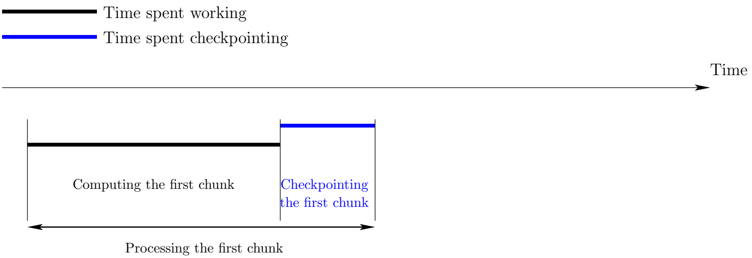


Checkpointing cost



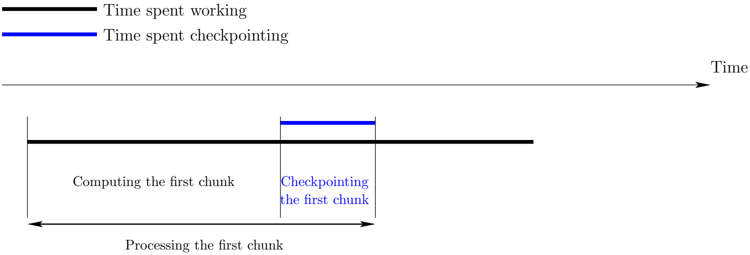
Blocking model: while a checkpoint is taken, no computation can be performed

Checkpointing cost



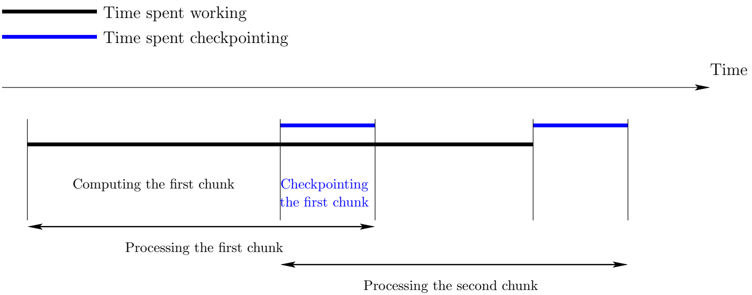
Non-blocking model: while a checkpoint is taken, computations are not impacted (e.g., first copy state to RAM, then copy RAM to disk)

Checkpointing cost



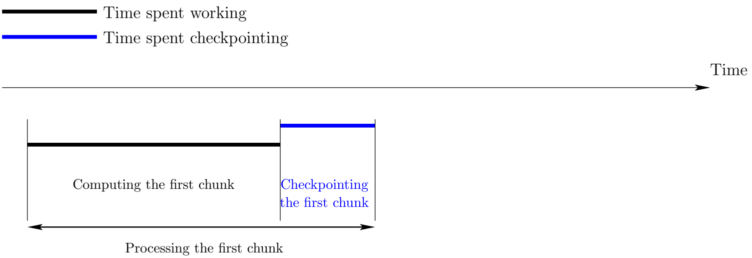
Non-blocking model: while a checkpoint is taken, computations are not impacted (e.g., first copy state to RAM, then copy RAM to disk)

Checkpointing cost



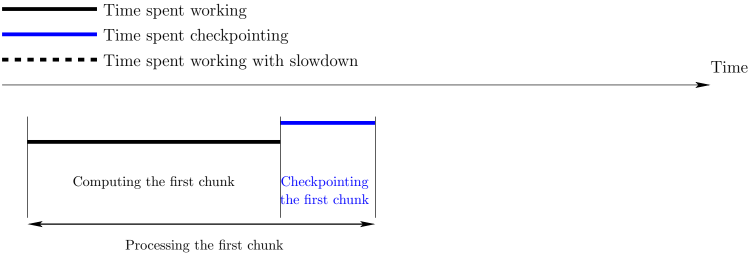
Non-blocking model: while a checkpoint is taken, computations are not impacted (e.g., first copy state to RAM, then copy RAM to disk)

Checkpointing cost



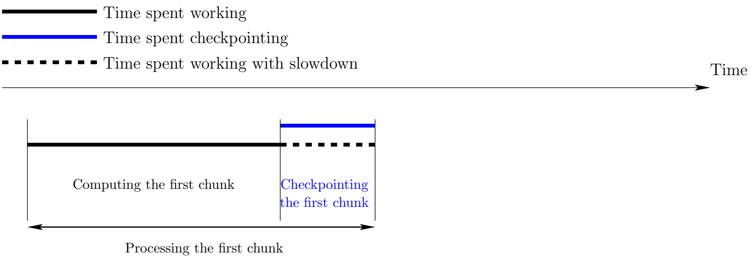
General model: while a checkpoint is taken, computations are slowed-down: during a checkpoint of duration C, the same amount of computation is done as during a time αC without checkpointing ($0 \leq \alpha \leq 1$).

Checkpointing cost



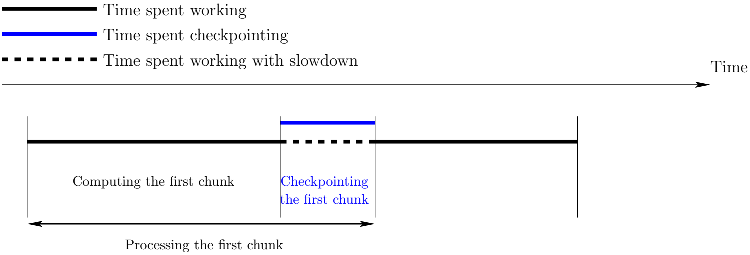
General model: while a checkpoint is taken, computations are slowed-down: during a checkpoint of duration C, the same amount of computation is done as during a time αC without checkpointing ($0 \leq \alpha \leq 1$).

Checkpointing cost



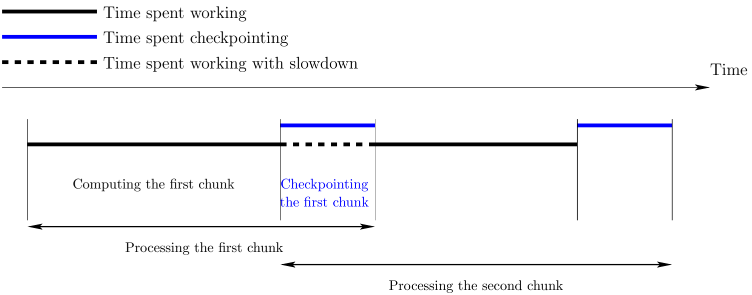
General model: while a checkpoint is taken, computations are slowed-down: during a checkpoint of duration C, the same amount of computation is done as during a time αC without checkpointing ($0 \leq \alpha \leq 1$).

Checkpointing cost



General model: while a checkpoint is taken, computations are slowed-down: during a checkpoint of duration C, the same amount of computation is done as during a time αC without checkpointing ($0 \leq \alpha \leq 1$).

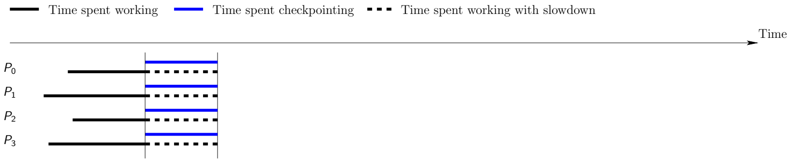
Checkpointing cost



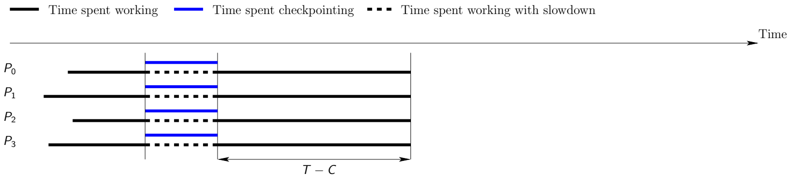
General model: while a checkpoint is taken, computations are slowed-down: during a checkpoint of duration C, the same amount of computation is done as during a time αC without checkpointing ($0 \leq \alpha \leq 1$).

Outline

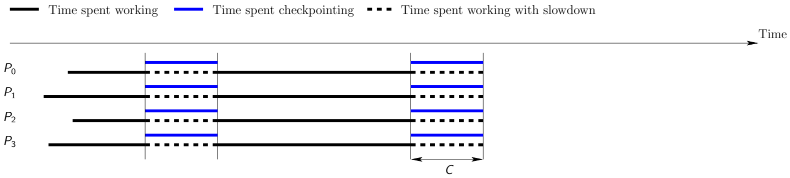
Waste in absence of failures



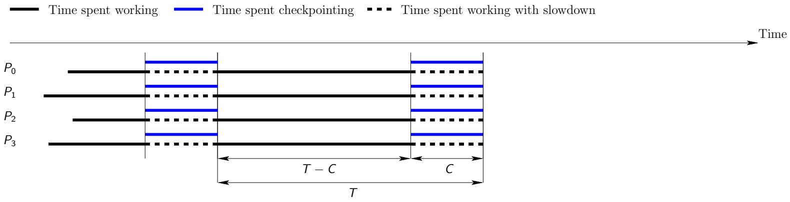
Waste in absence of failures



Waste in absence of failures



Waste in absence of failures

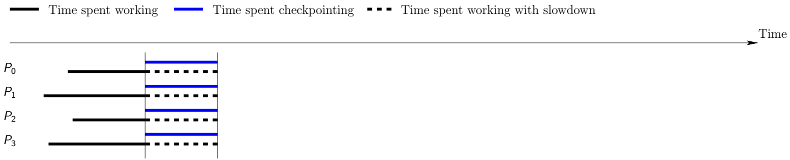


Time elapsed since last checkpoint: T

Amount of computation saved: $(T-C)+\alpha C$

$$\text{WASTE}_{coord-nofailure} = \frac{T-((T-C)+\alpha C)}{T} = \frac{(1-\alpha)C}{T}$$

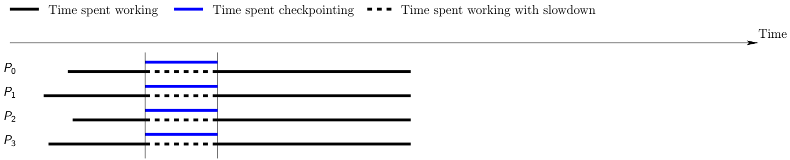
Waste due to failures



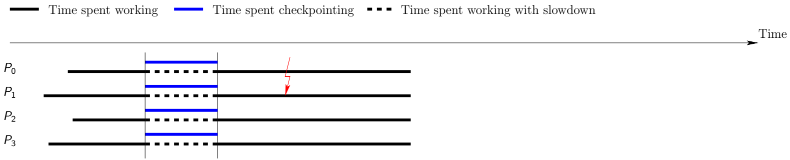
Failure can happen

1. During computation phase
2. During checkpointing phase

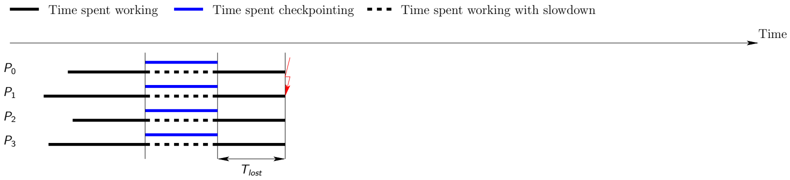
Waste due to failures in computation phase



Waste due to failures in computation phase

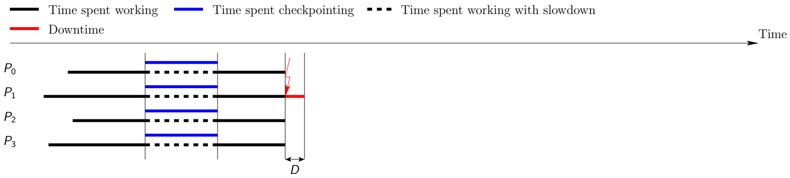


Waste due to failures in computation phase

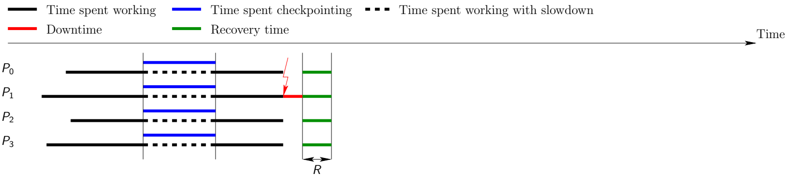


Tightly-coupled model: when one processor is victim of a failure, all processors lose their work and must roll-back to last checkpoint

Waste due to failures in computation phase

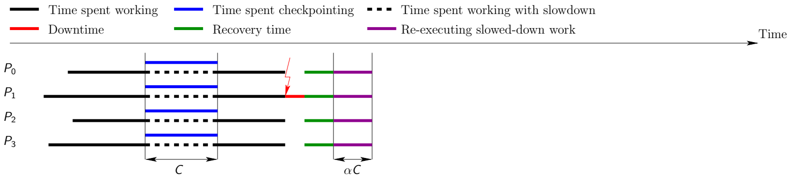


Waste due to failures in computation phase



Tightly-coupled model: All processors must recover from last checkpoint

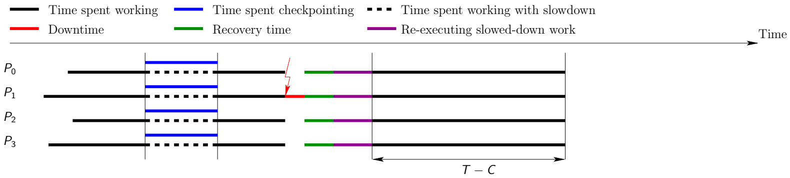
Waste due to failures in computation phase



Redo the work destroyed by the failure, that was done in the checkpointing phase before the computation phase

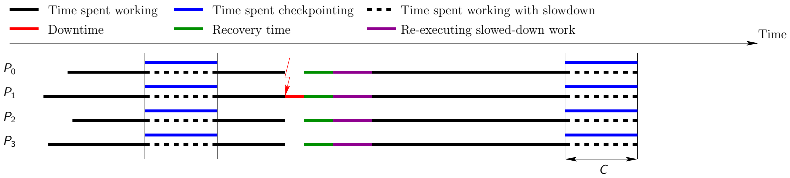
But no checkpoint is taken in parallel, hence this re-computation is faster than the original computation

Waste due to failures in computation phase



Re-execute the computation phase

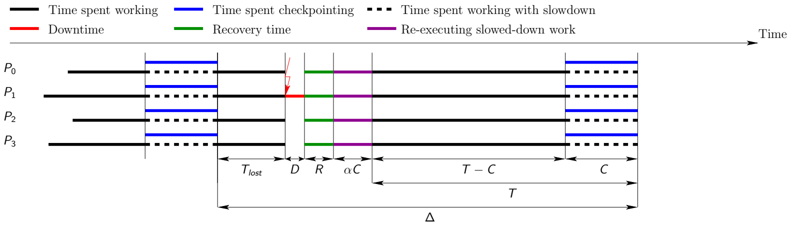
Waste due to failures in computation phase



Finally, the checkpointing phase is executed

First-order approximation: we assume that no other failure occurs during the re-execution

Waste due to failures in computation phase

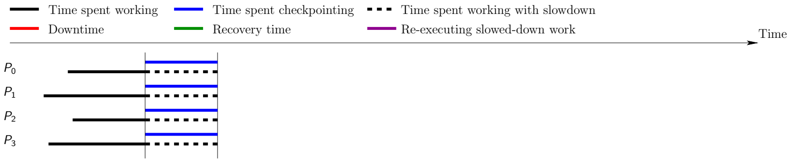


RE-EXEC: $\Delta - T = T_{lost} + D + R + \alpha C$

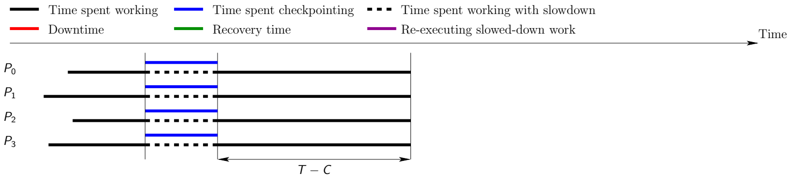
First-order: $T_{lost} = \frac{1}{2}(T - C)$

$$\text{RE-EXEC}_{coord-fail-in-work} = \frac{T-C}{2} + D + R + \alpha C$$

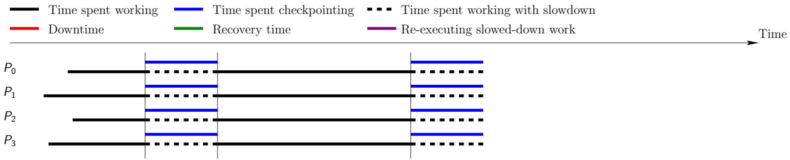
Waste due to failures in checkpointing phase



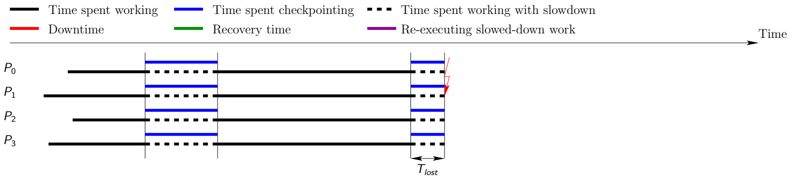
Waste due to failures in checkpointing phase



Waste due to failures in checkpointing phase

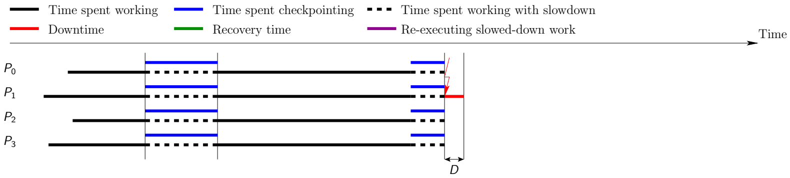


Waste due to failures in checkpointing phase

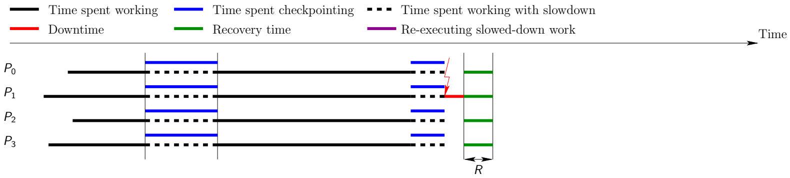


Tightly-coupled model: when one processor is victim of a failure, all processors lose their work and must roll-back to last checkpoint

Waste due to failures in checkpointing phase

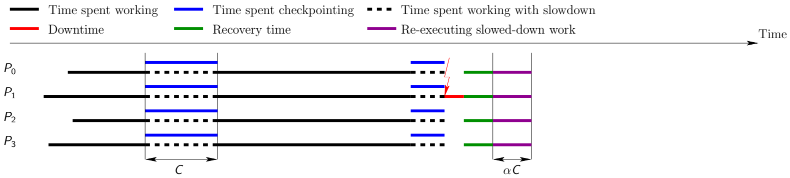


Waste due to failures in checkpointing phase



Tightly-coupled model: All processors must recover from last checkpoint

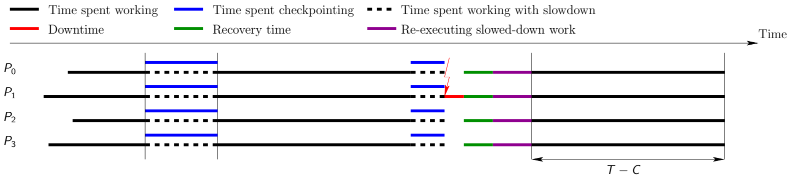
Waste due to failures in checkpointing phase



Redo the work destroyed by the failure, that was done in the checkpointing phase before the computation phase

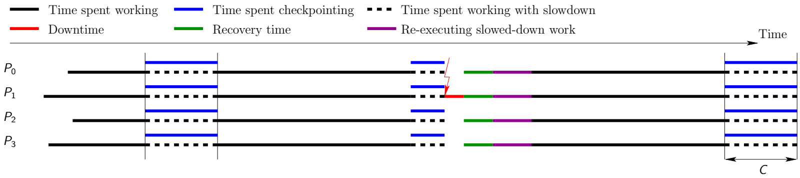
But no checkpoint is taken in parallel, hence this re-computation is faster than the original computation

Waste due to failures in checkpointing phase



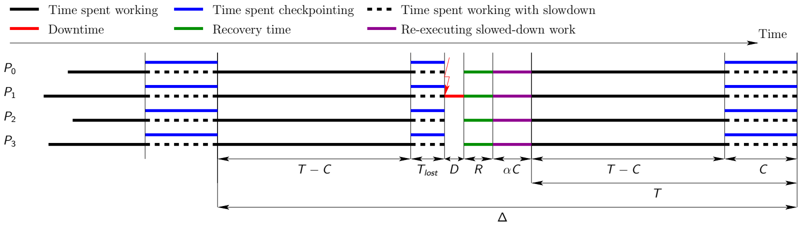
Re-execute the computation phase

Waste due to failures in checkpointing phase



Finally, the checkpointing phase is executed

Waste due to failures in checkpointing phase



Re-Exec: $\Delta - T = (T - C) + T_{lost} + D + R + \alpha C$

First-order approximation: $T_{lost} = \frac{1}{2} C$

$$\begin{aligned}\text{Re-Exec}_{coord-fail-in-checkpoint} &= (T - C) + \frac{C}{2} + D + R + \alpha C \\ &= T - \frac{C}{2} + D + R + \alpha C\end{aligned}$$

Waste due to failures

- Failure in the computation phase (probability: $\frac{T-C}{T}$)

$$\text{RE-EXEC}_{coord-fail-in-work} = \frac{T-C}{2} + D + R + \alpha C$$

- Failure in the checkpointing phase (probability: $\frac{C}{T}$)

$$\text{RE-EXEC}_{coord-fail-in-checkpoint} = T - \frac{C}{2} + D + R + \alpha C$$

$$\frac{T-C}{T}\left(\frac{T-C}{2} + D + R + \alpha C\right) + \frac{C}{T}\left(T - \frac{C}{2} + D + R + \alpha C\right)$$

$$= D + R + \alpha C + \frac{T}{2}$$

Overall waste

$$\textsc{Waste}_{coord} = \textsc{Waste}_{coord-nofailure} + \frac{1}{\mu}\textsc{Re-Exec}_{coord-failure}$$

$$= \frac{(1-\alpha)C}{T} + \frac{1}{\mu}\left(D + R + \alpha C + \frac{T}{2}\right)$$

Minimize $\textsc{Waste}_{coord}$ subject to:

1. $C \leq T$ (by construction)
2. $T \leq 0.1\mu$ ($\Rightarrow Proba(Poisson(\frac{T}{\mu}) \geq 2) \leq 0.05$)

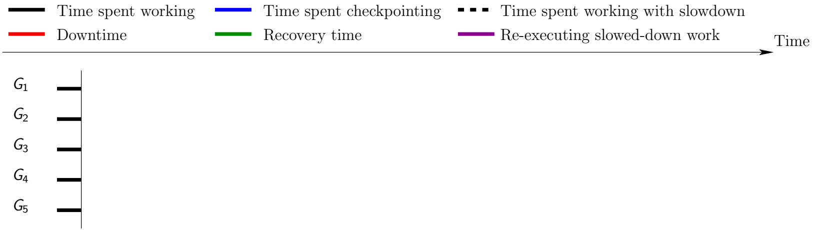
If μ large enough, optimal period is $\mathbb{T} = \sqrt{2\mu C(1-\alpha)}$
(remember Young's approximation)

Outline

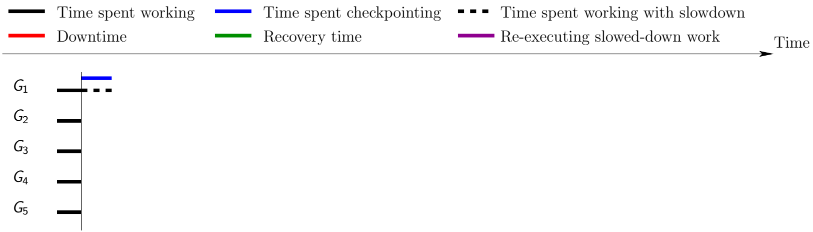
Hierarchical checkpointing

- Processors partitioned into G groups
- Each group includes q processors
- Inside each group: coordinated checkpointing in time $C(q)$
- Inter-group messages are logged

Impact of checkpointing

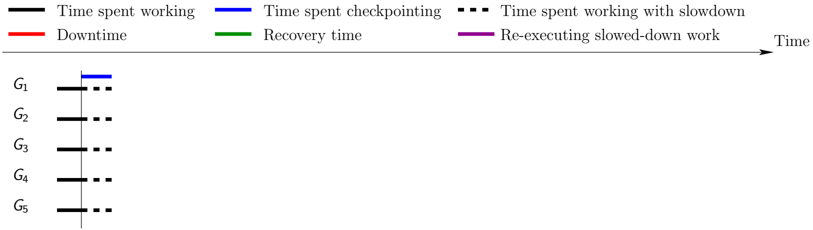


Impact of checkpointing



When a group checkpoints, its own computation speed is slowed-down

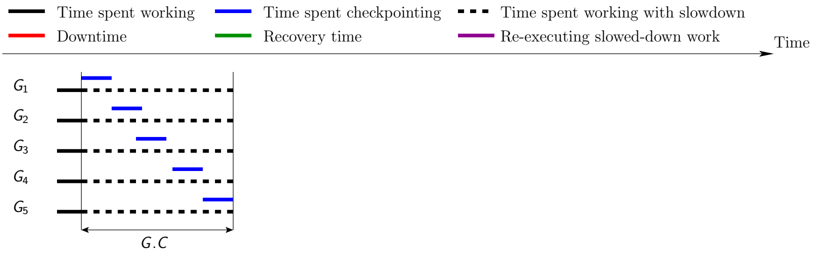
Impact of checkpointing



When a group checkpoints, its own computation speed is slowed-down

This holds for all groups because of the tightly-coupled assumption

Impact of checkpointing

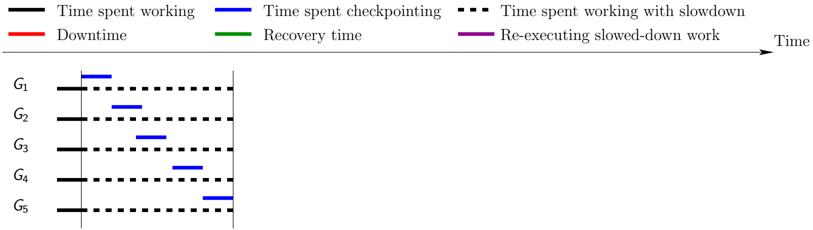


When a group checkpoints, its own computation speed is slowed-down

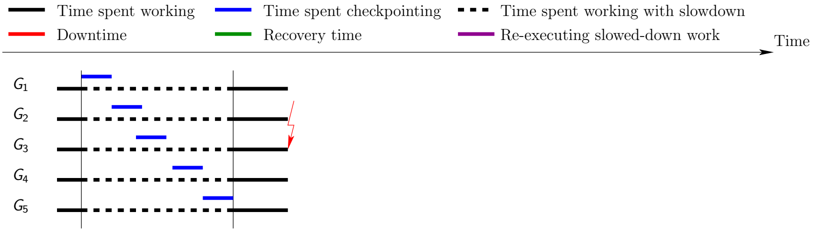
This holds for all groups because of the tightly-coupled assumption

$\textsc{Waste} = \frac{T - \textsc{Work}}{T}$ where $\textsc{Work} = T - (1-\alpha)GC(q)$

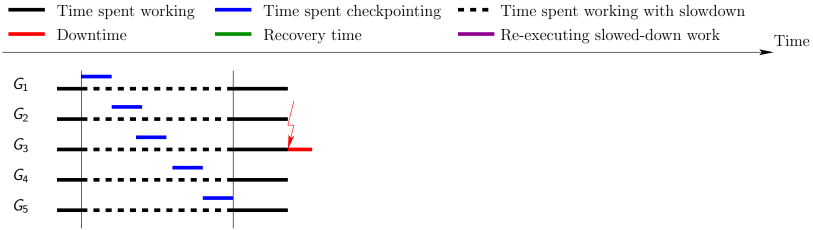
Failure during computation phase



Failure during computation phase

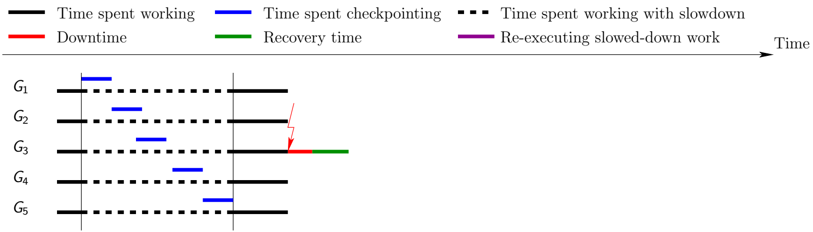


Failure during computation phase



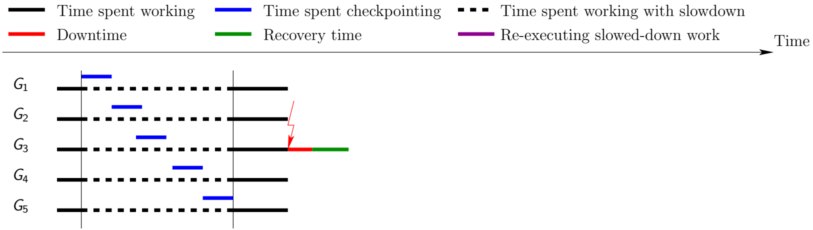
Tightly-coupled model: while one group is in downtime, none can work

Failure during computation phase



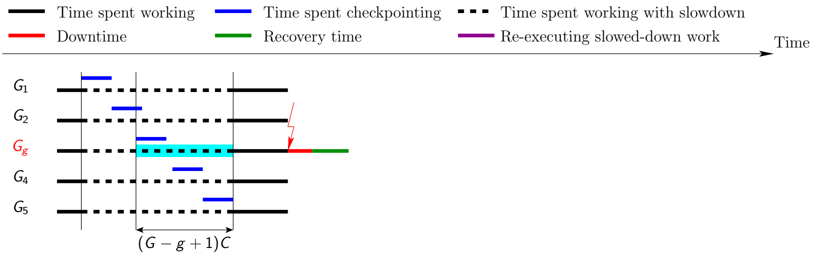
Tightly-coupled model: while one group is in recovery, none can work

Failure during computation phase



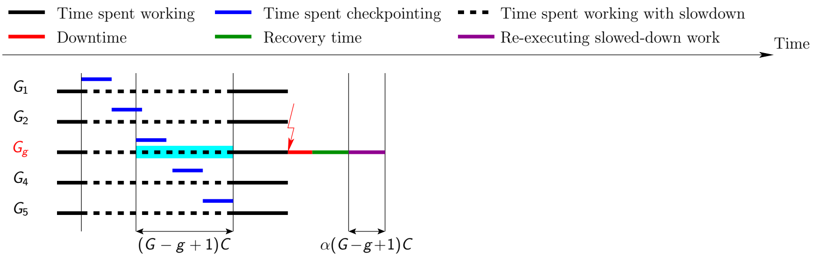
Groups must have completed the same amount of work in between two consecutive checkpoints, independently of the fact that a failure may or may not have happened on the platform in between these checkpoints. Hence, no checkpointing is possible during the rollback.

Failure during computation phase



Redo work done during previous checkpointing phase and that was destroyed by the failure

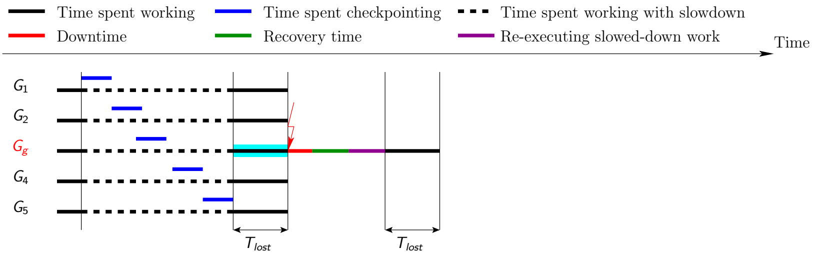
Failure during computation phase



Redo work done during previous checkpointing phase and that was destroyed by the failure

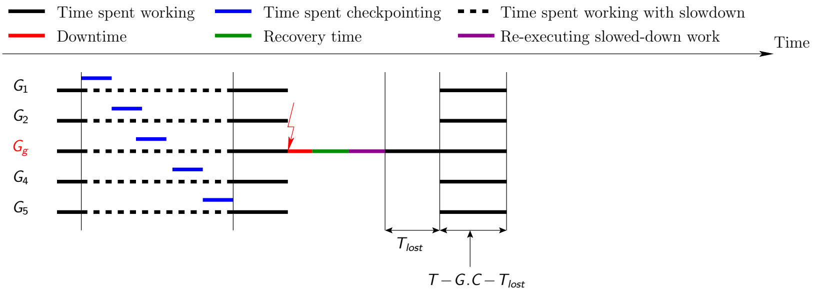
But no checkpoint is taken in parallel, hence this re-computation is faster than the original computation

Failure during computation phase



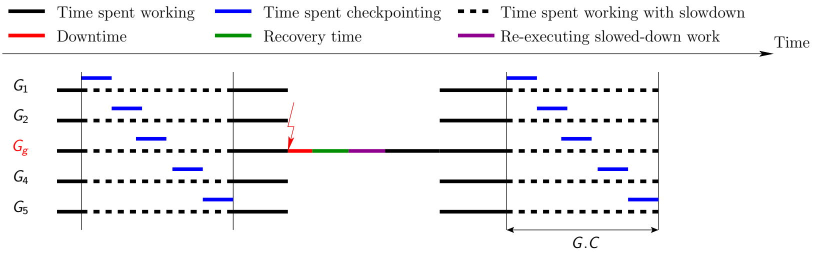
Redo work done in computation phase and that was destroyed by the failure

Failure during computation phase



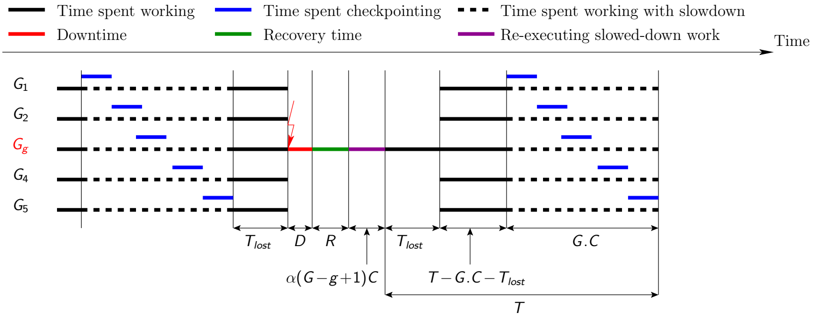
Failing group has reached the point where it previously failed, all groups now resume execution in parallel and complete the computation phase

Failure during computation phase



Finally, perform checkpointing phase

Failure during computation phase

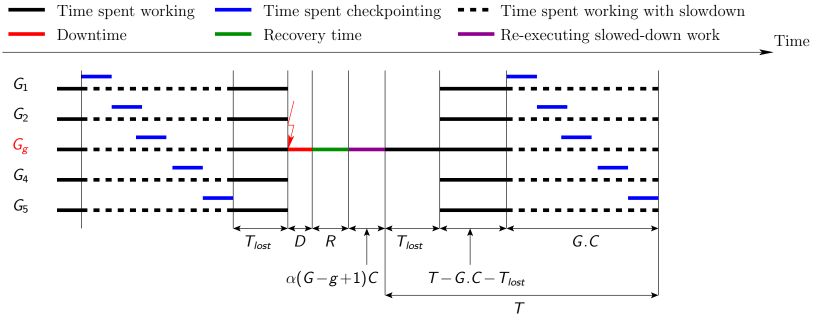


Re-Exec: $T_{lost} + D + R + \alpha(G - g + 1)C$

First-order: $T_{lost} = \frac{1}{2}(T - G.C)$

Approximated Re-Exec: $\frac{T-G.C}{2} + D + R + \alpha(G - g + 1)C$

Failure during computation phase



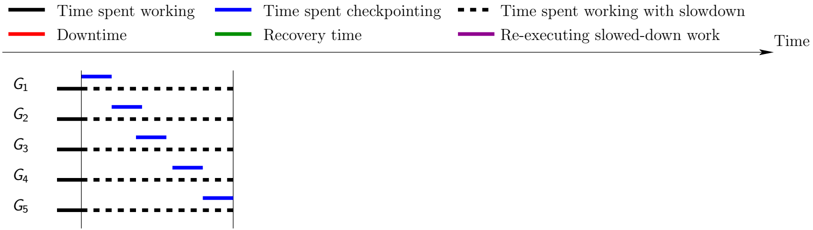
Approximated RE-EXEC: $\frac{T-G.C}{2} + D + R + \alpha(G-g+1)C$

Average approximated RE-EXEC:

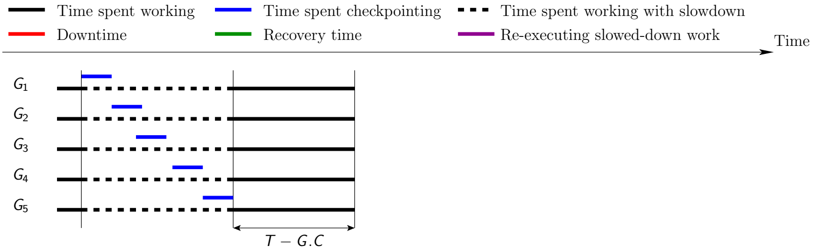
$$\frac{1}{G}\sum_{g=1}^{G}\left[\frac{T-G.C(q)}{2} + D(q) + R(q) + \alpha(G-g+1)C(q)\right]$$

$$= \frac{T-G.C(q)}{2} + D(q) + R(q) + \alpha\frac{G+1}{2}C$$

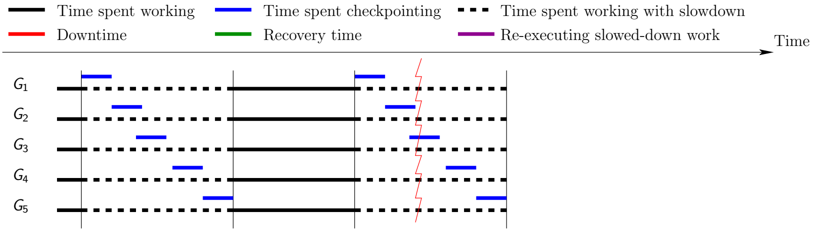
Failure during checkpointing phase



Failure during checkpointing phase



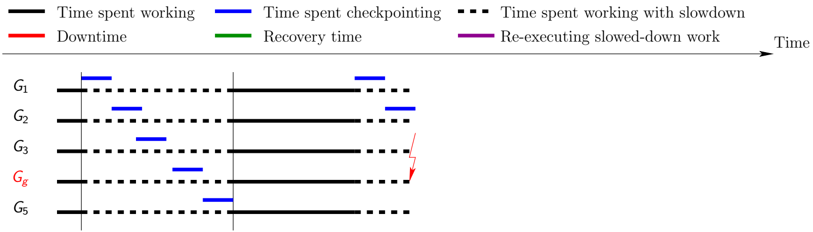
Failure during checkpointing phase



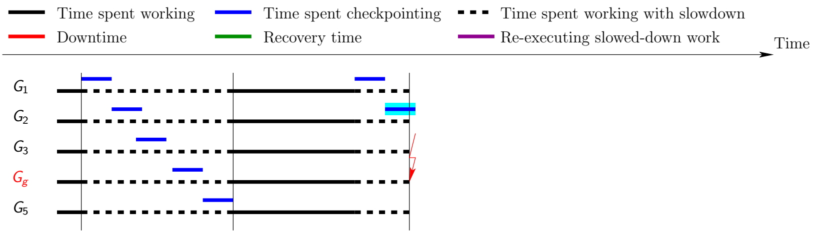
When does the failing group fail?

1. Before starting its own checkpoint
2. While taking its own checkpoint
3. After completing its own checkpoint

Failure during checkpointing phase: failure before checkpoint

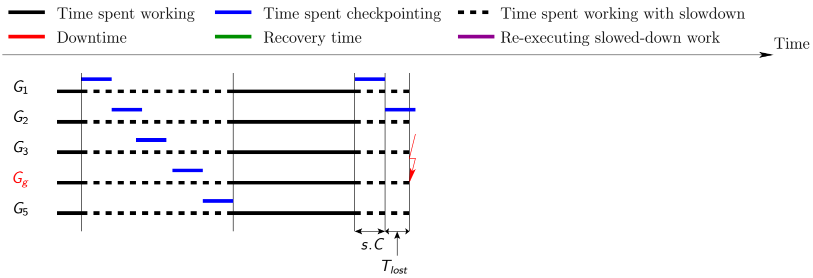


Failure during checkpointing phase: failure before checkpoint



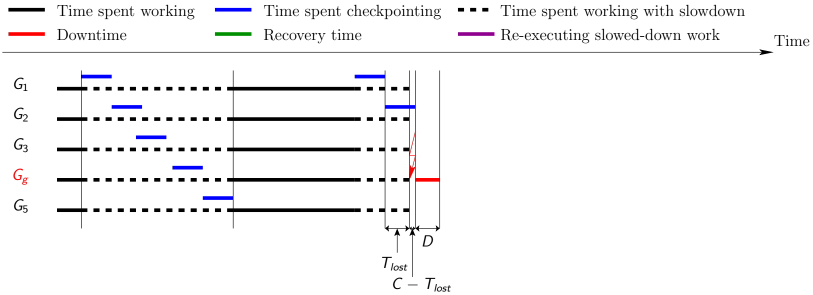
The checkpoint taken while the failure struck is that of another group; it is not affected and completes

Failure during checkpointing phase: failure before checkpoint



s: number of groups that have successfully completed their checkpoints before the failure

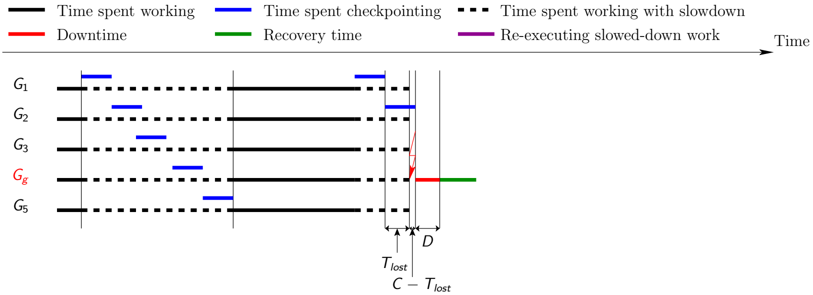
Failure during checkpointing phase: failure before checkpoint



Tightly-coupled model: while one group is in downtime, none can work

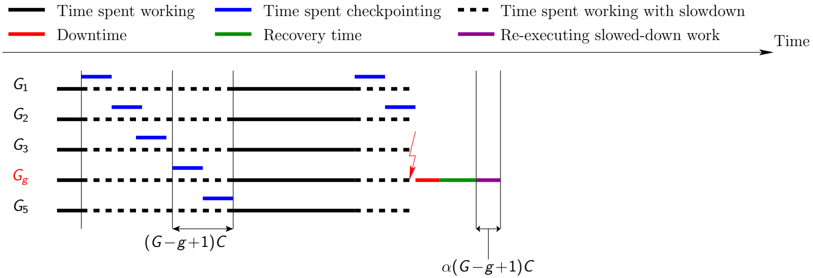
Groups must have completed the same amount of work in between two consecutive checkpoints, independently of the fact that a failure may or may not have happened on the platform in between these checkpoints. Hence, no checkpointing is possible during the rollback.

Failure during checkpointing phase: failure before checkpoint



Tightly-coupled model: while one group is in recovery, none can work

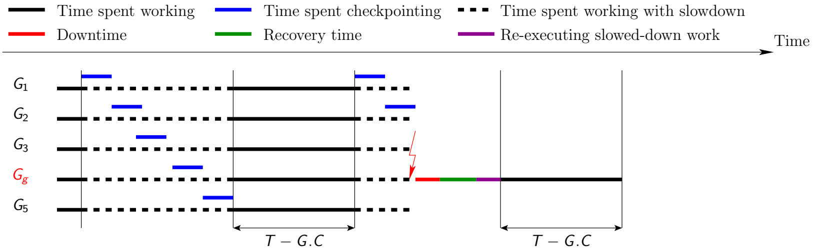
Failure during checkpointing phase: failure before checkpoint



Redo work done during previous checkpointing phase and that was destroyed by the failure

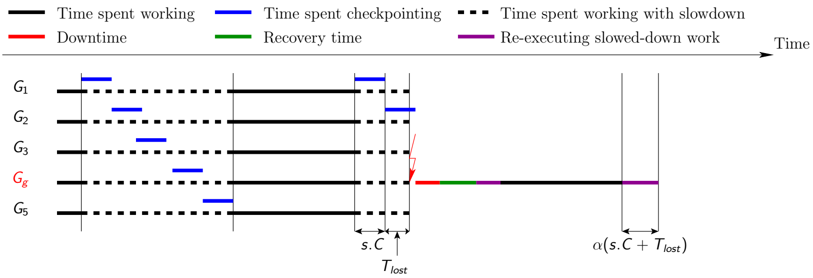
But no checkpoint is taken in parallel, hence this re-computation is faster than the original computation

Failure during checkpointing phase: failure before checkpoint



Redo work done in computation phase and that was destroyed by the failure

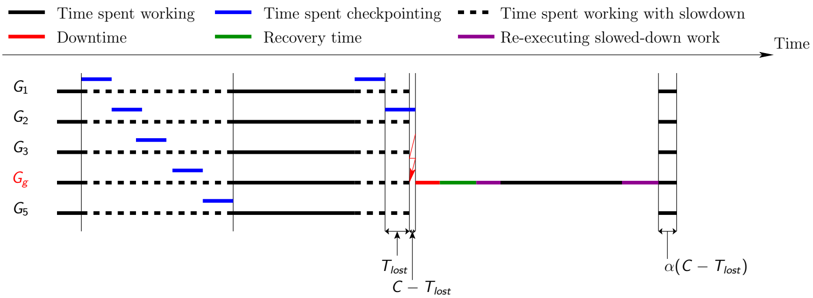
Failure during checkpointing phase: failure before checkpoint



Redo work done in checkpointing phase and that was destroyed by the failure

But no checkpoint is taken in parallel, hence this re-computation is faster than the original computation

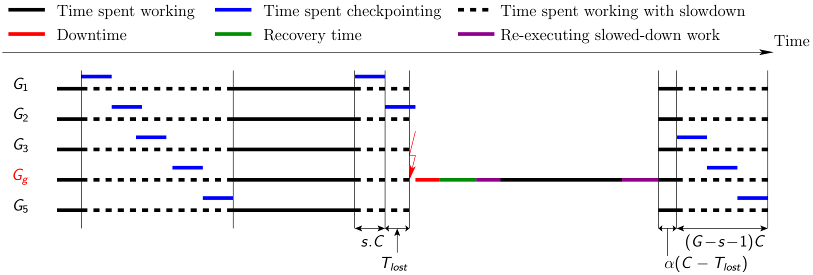
Failure during checkpointing phase: failure before checkpoint



Failing group has reached the point where it previously failed, all groups now resume execution in parallel and complete the computation phase

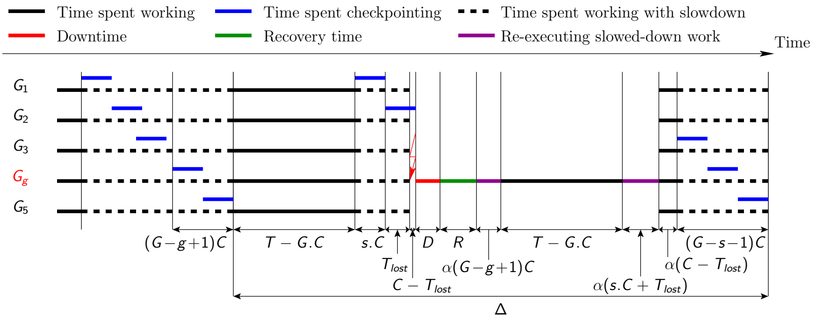
Groups first complete work that was to be done during the checkpoint during which the failure occurred

Failure during checkpointing phase: failure before checkpoint



Checkpointing phase completed

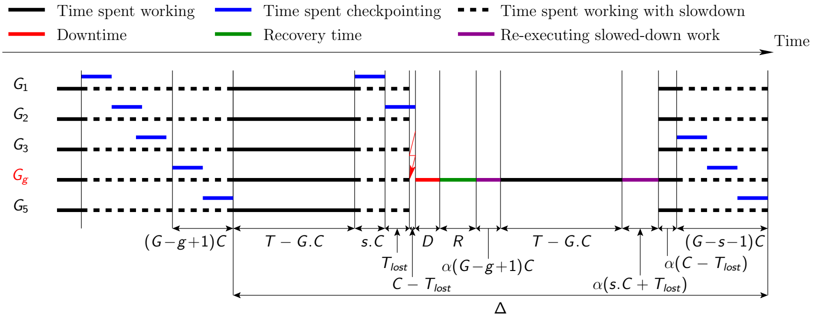
Failure during checkpointing phase: failure before checkpoint



RE-EXEC$= \Delta - T$

$$\begin{aligned}\Delta = \;& (T-G.C)+s.C+T_{lost}+C-T_{lost}+D+R+\alpha(G-g+1)C \\ & +(T-G.C)+\alpha(s.C+T_{lost})+\alpha(C-T_{lost})+(G-s-1).C \\ = \;& 2T-GC+D+R+\alpha(G-g+s+2)C\end{aligned}$$

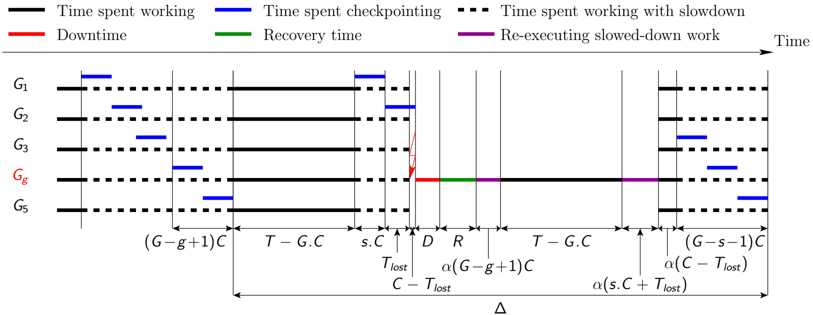
Failure during checkpointing phase: failure before checkpoint



$$\text{RE-EXEC} = \Delta - T$$

$$\Delta = \quad 2T - GC + D + R + \alpha(G-g+s+2)C$$

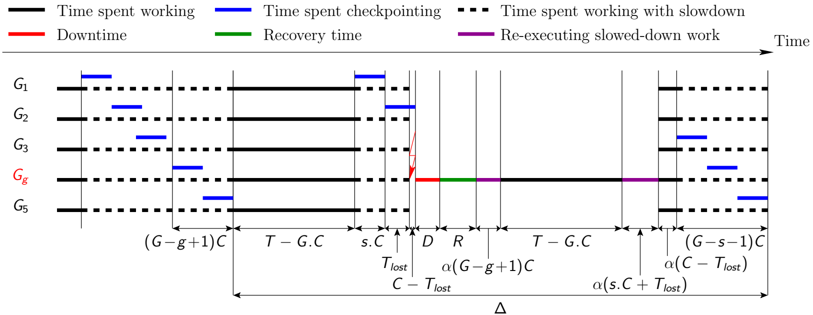
Failure during checkpointing phase: failure before checkpoint



RE-EXEC$= \Delta - T$

$$\Delta = 2T - GC + D + R + \alpha(G-g+s+2)C$$

Failure during checkpointing phase: failure before checkpoint

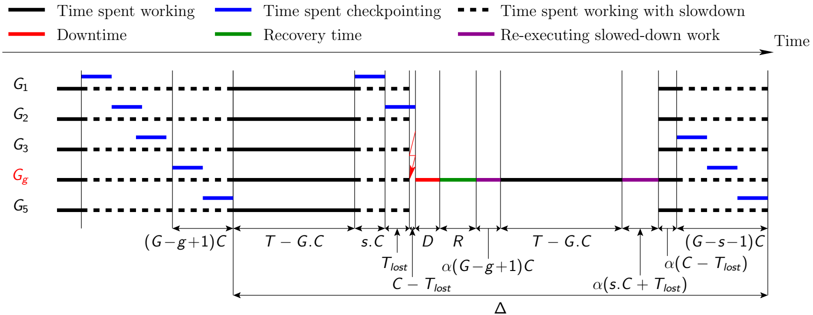


$$\text{RE-EXEC} = \Delta - T$$

$$\Delta = 2T - GC + D + R + \alpha(G-g+s+2)C$$

$$\text{RE-EXEC} = T+D+R+((\alpha-1)G + \alpha(-g+s+2)).C$$

Failure during checkpointing phase: failure before checkpoint



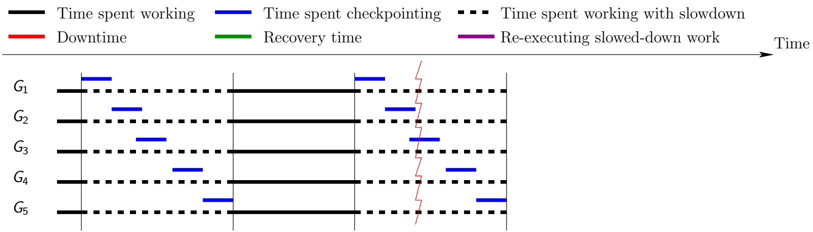
$$\text{RE-EXEC} = T+D+R+((\alpha-1)G+\alpha(-g+s+2)).C$$

Average RE-EXEC for group g (for $2 \leq g \leq G$):

$$\frac{1}{g-1}\sum_{s=0}^{g-2}(T+D(q)+R(q)+((\alpha-1)G+\alpha(-g+s+2)).C(q))$$

$$= T+D(q)+R(q)+\left((\alpha-1)G-\alpha\frac{g-2}{2}\right).C(q)$$

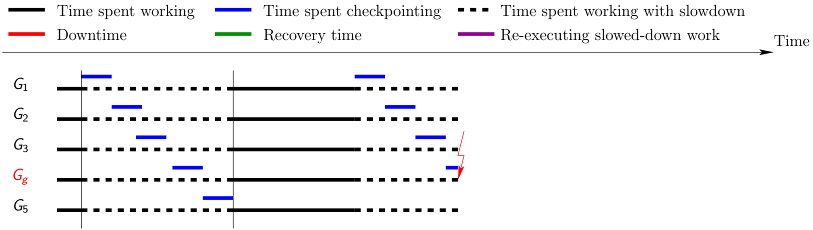
Failure during checkpointing phase



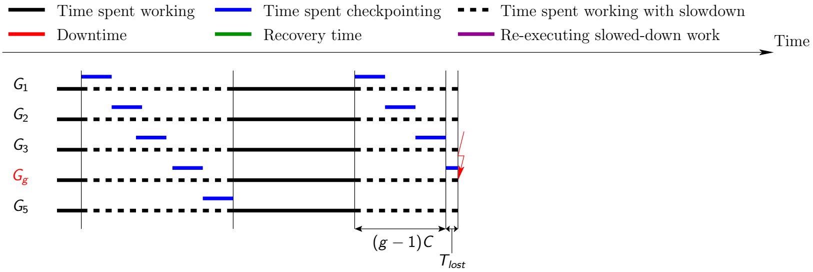
When does the failing group fail?

1. Before starting its own checkpoint
2. While taking its own checkpoint
3. After completing its own checkpoint

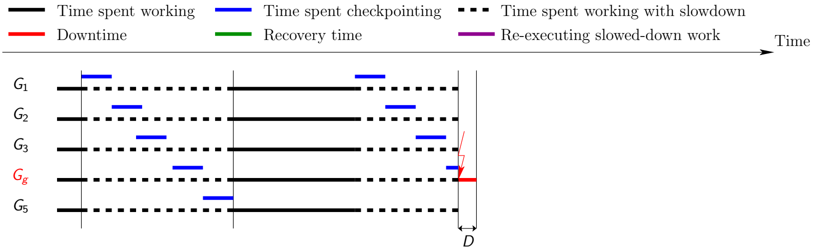
Failure during checkpointing phase: failure during checkpoint



Failure during checkpointing phase: failure during checkpoint



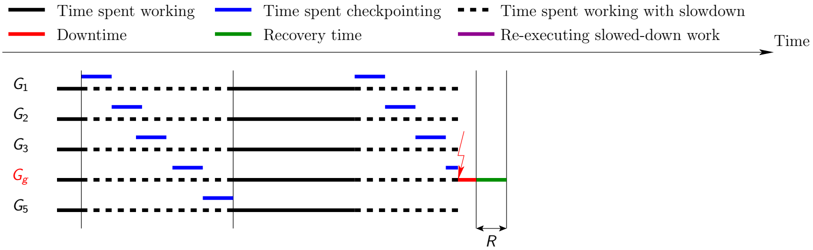
Failure during checkpointing phase: failure during checkpoint



Tightly-coupled model: while one group is in downtime, none can work

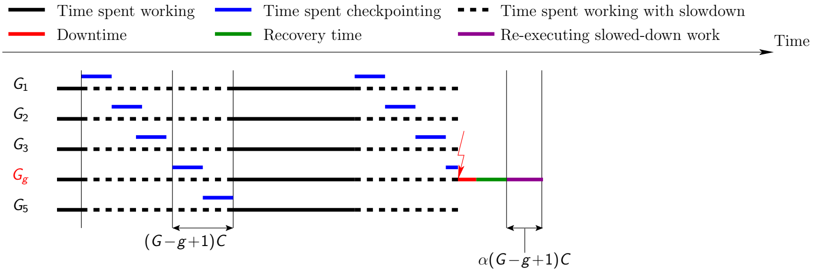
Groups must have completed the same amount of work in between two consecutive checkpoints, independently of the fact that a failure may or may not have happened on the platform in between these checkpoints. Hence, no checkpointing is possible during the rollback.

Failure during checkpointing phase: failure during checkpoint



Tightly-coupled model: while one group is in recovery, none can work

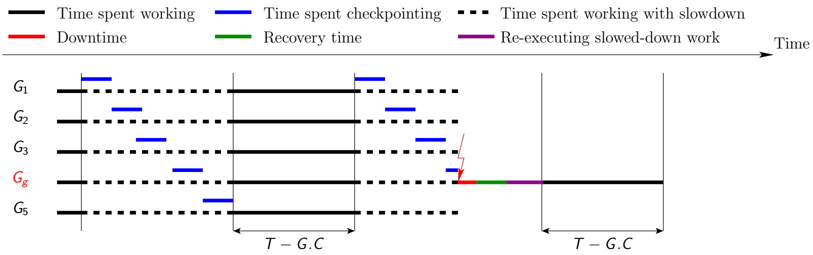
Failure during checkpointing phase: failure during checkpoint



Redo work done during previous checkpointing phase and that was destroyed by the failure

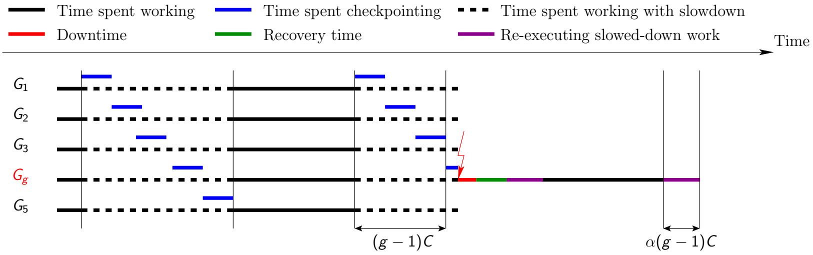
But no checkpoint is taken in parallel, hence this re-computation is faster than the original computation

Failure during checkpointing phase: failure during checkpoint



Redo work done in computation phase and that was destroyed by the failure

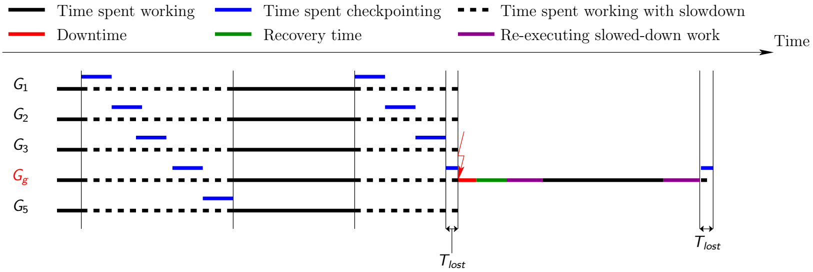
Failure during checkpointing phase: failure during checkpoint



Redo work done in checkpointing phase that was destroyed by the failure and that preceded the beginning of the killed checkpoint

But no checkpoint is taken in parallel, hence this re-computation is faster than the original computation

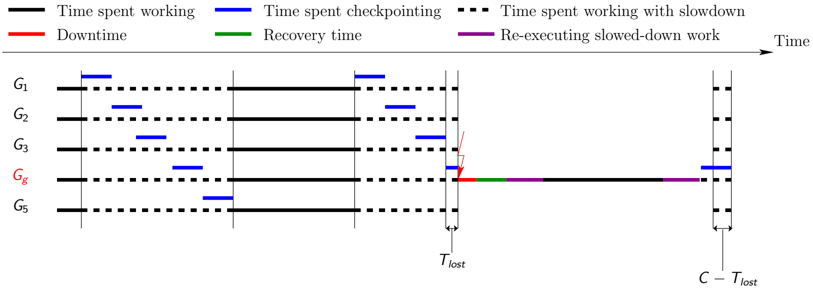
Failure during checkpointing phase: failure during checkpoint



The failing group has now reached the point where it can retry taking its checkpoint

Redo work done during the checkpoint and that was destroyed by the failure

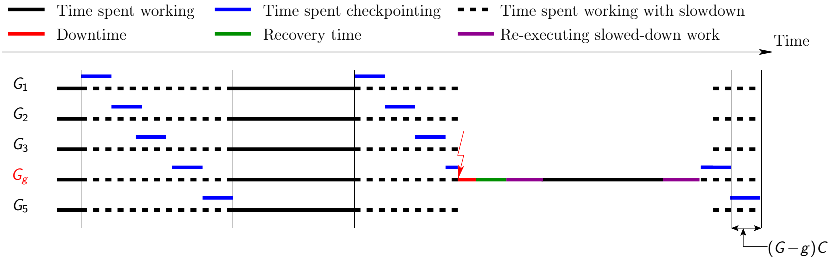
Failure during checkpointing phase: failure during checkpoint



Failing group has reached the point where it previously failed, all groups now resume execution in parallel and complete the computation phase

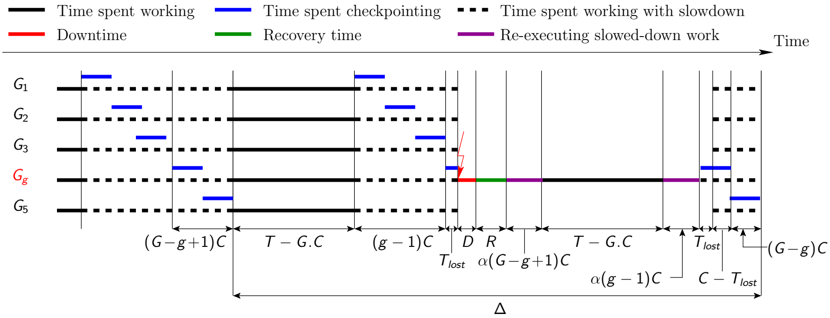
Failing group completes its checkpoint

Failure during checkpointing phase: failure during checkpoint



Checkpointing phase completed

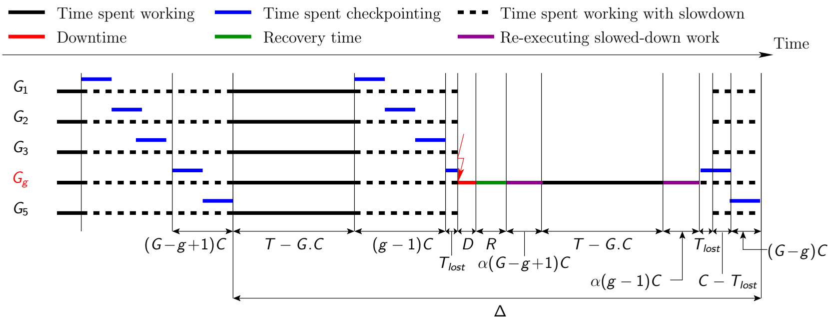
Failure during checkpointing phase: failure during checkpoint



RE-EXEC$= \Delta - T$

$$
\begin{aligned}
\Delta = \; & (T - G.C) + (g-1)C + T_{lost} + D + R + \alpha(G-g+1)C \\
& + (T - G.C) + \alpha(g-1)C + T_{lost} + (C - T_{lost}) + (G-g)C
\end{aligned}
$$

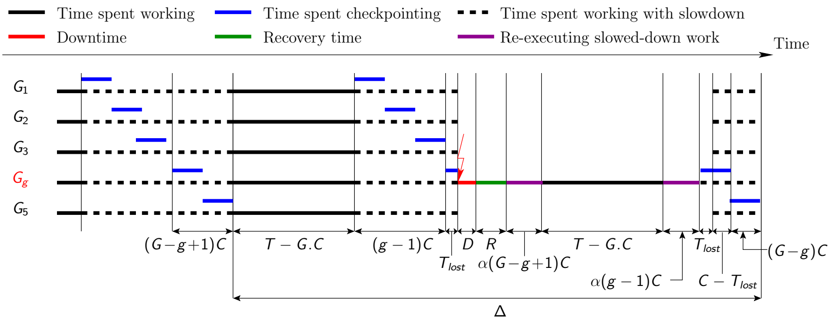
Failure during checkpointing phase: failure during checkpoint



$$\text{RE-EXEC} = \Delta - T$$

$$\begin{aligned}\Delta = \; & (T - G.C) + (g-1)C + T_{lost} + D + R + \alpha(G-g+1)C \\ & +(T - G.C) + \alpha(g-1)C + T_{lost} + (C - T_{lost}) + (G-g)C \\ = \; & T + (\alpha - 1)G.C + T_{lost} + D + R + T\end{aligned}$$

Failure during checkpointing phase: failure during checkpoint

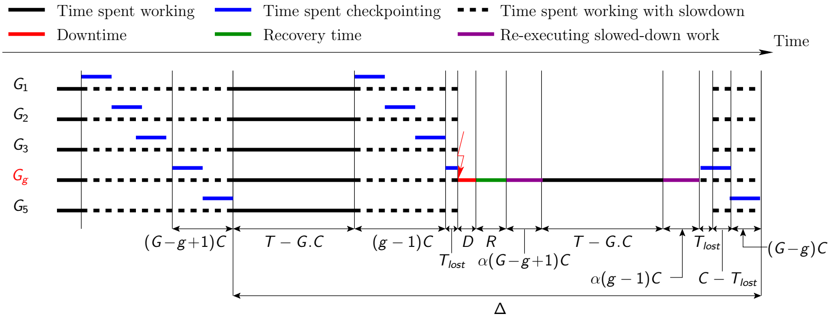


RE-EXEC$= \Delta - T$

$$\begin{aligned}\Delta = \; & (T - G.C) + (g-1)C + T_{lost} + D + R + \alpha(G-g+1)C \\ & +(T-G.C) + \alpha(g-1)C + T_{lost} + (C - T_{lost}) + (G-g)C \\ = \; & T + (\alpha - 1)G.C + T_{lost} + D + R + T\end{aligned}$$

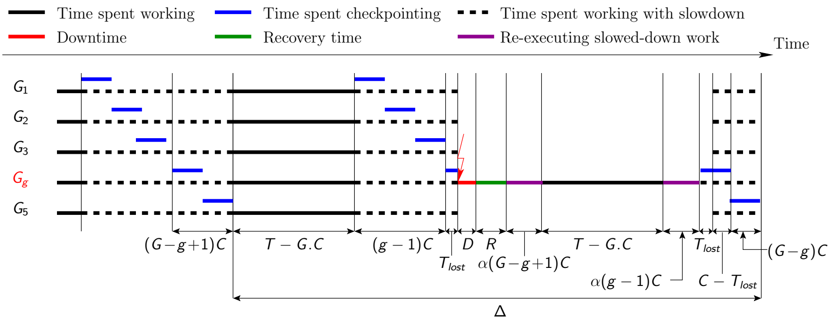
RE-EXEC$= T + (\alpha - 1)G.C + T_{lost} + D + R$

Failure during checkpointing phase: failure during checkpoint



$$\text{Re-Exec} = T + (\alpha - 1)G.C + T_{lost} + D + R$$

Failure during checkpointing phase: failure during checkpoint



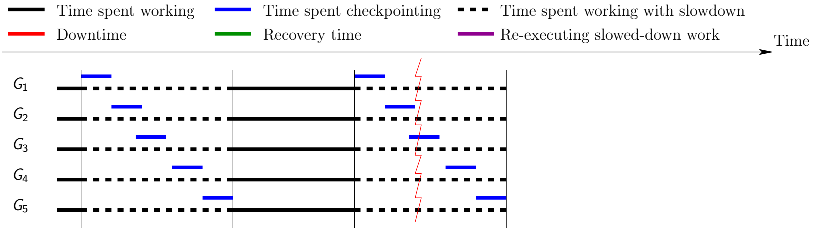
$$\text{RE-EXEC} = T + (\alpha - 1)G.C + T_{lost} + D + R$$

Approximation: $T_{lost} = \frac{C}{2}$

Approximated RE-EXEC

$$T + (\alpha - 1)G.C(q) + \frac{C(q)}{2} + D(q) + R(q)$$

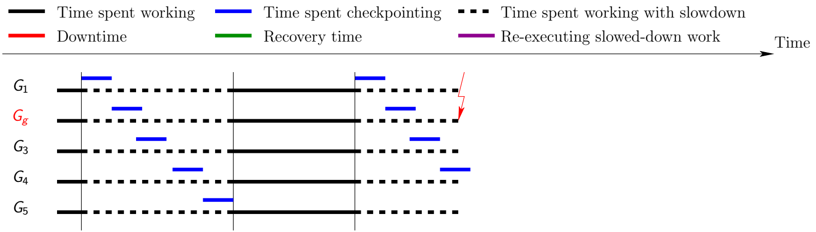
Failure during checkpointing phase



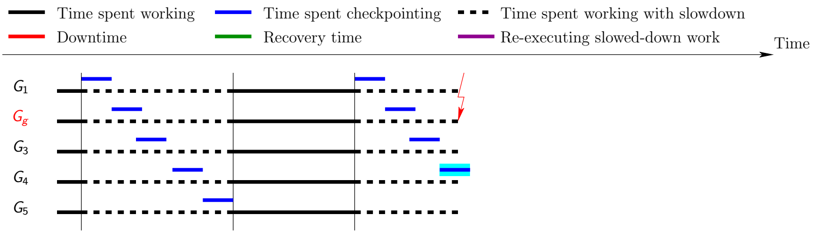
When does the failing group fail?

1. Before starting its own checkpoint
2. While taking its own checkpoint
3. After completing its own checkpoint

Failure during checkpointing phase: failure after checkpoint

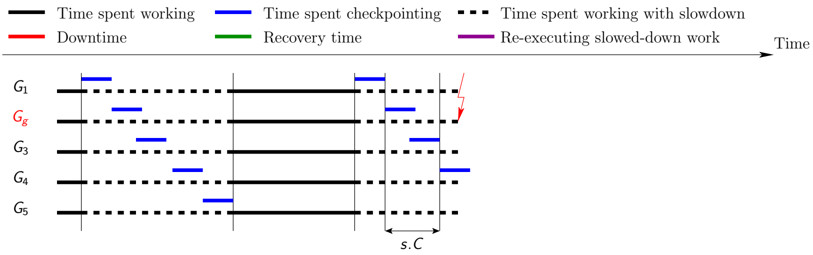


Failure during checkpointing phase: failure after checkpoint



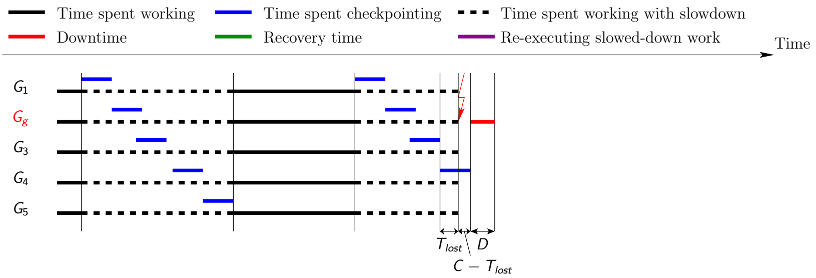
The checkpoint taken while the failure struck is that of another group; it is not affected and completes

Failure during checkpointing phase: failure after checkpoint



s: number of groups that have successfully completed their checkpoints before the failure, among groups that are after the failing group (including the failing group)

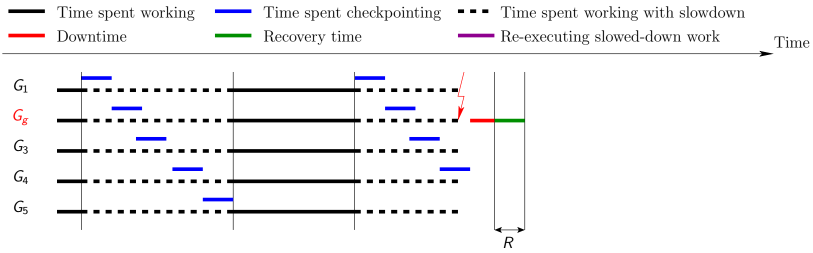
Failure during checkpointing phase: failure after checkpoint



Tightly-coupled model: while one group is in downtime, none can work

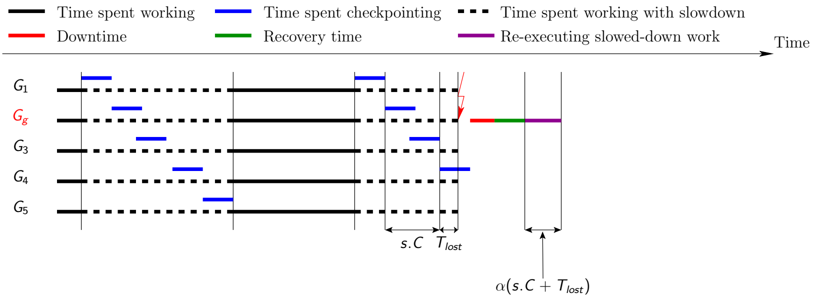
Groups must have completed the same amount of work in between two consecutive checkpoints, independently of the fact that a failure may or may not have happened on the platform in between these checkpoints. Hence, no checkpointing is possible during the rollback.

Failure during checkpointing phase: failure after checkpoint



Tightly-coupled model: while one group is in recovery, none can work

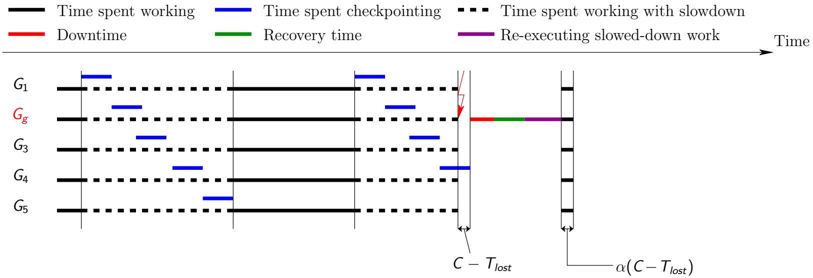
Failure during checkpointing phase: failure after checkpoint



Redo work done in checkpointing phase and that was destroyed by the failure

But no checkpoint is taken in parallel, hence this re-computation is faster than the original computation

Failure during checkpointing phase: failure after checkpoint

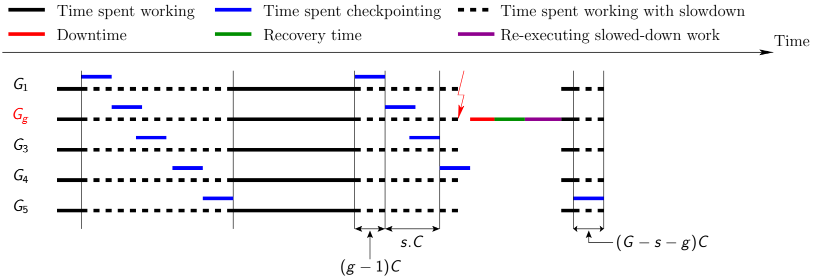


Failing group has reached the point where it previously failed, all groups now resume execution in parallel

Groups first complete work that was to be done during the checkpoint during which the failure occurred

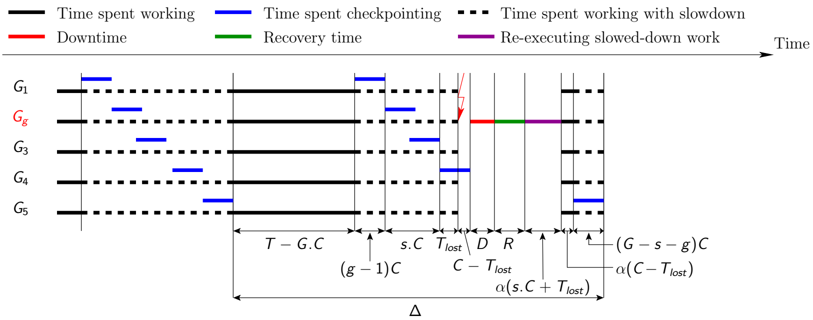
But no checkpoint is taken in parallel, hence this re-computation is faster than the original computation

Failure during checkpointing phase: failure after checkpoint



Checkpointing phase completed

Failure during checkpointing phase: failure after checkpoint

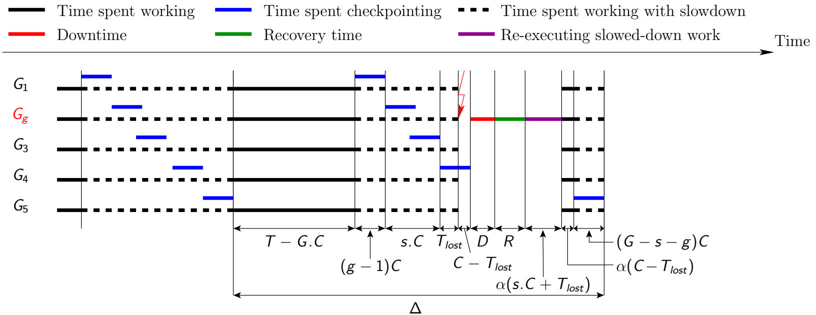


RE-EXEC $= \Delta - T$

$$
\begin{aligned}
\Delta = \;& (T - G.C) + (g-1)C + s.C + T_{lost} + C - T_{lost} + D \\
& + R + \alpha(s.C + T_{lost}) + \alpha(C - T_{lost}) + (G - s - g)C \\
= \;& T + D + R + \alpha(s+1).C
\end{aligned}
$$

RE-EXEC $= D + R + \alpha(s+1)C$

Failure during checkpointing phase: failure after checkpoint



Re-Exec$= D + R + \alpha(s+1)C$

Average Re-Exec for group g (for $1 \leq g \leq G-1$):

$$\frac{1}{G-g}\sum_{s=1}^{G-g}(D(q)+R(q)+\alpha(s+1)C(q))$$

$$= D(q) + R(q) + \alpha\frac{G-g+3}{2}C(q)$$

Average waste for failures during checkpointing phase

Average RE-EXEC when the failing-group g fails

1. Before its checkpoint (for $2 \leq g \leq G$):

$$\text{RE-EXEC}_{before\_ckpt} = T + D(q) + R(q) + ((\alpha - 1)G - \alpha \frac{g-2}{2}).C(q)$$

2. During its checkpoint

$$\text{RE-EXEC}_{during\_ckpt} = T + (\alpha - 1)G.C(q) + \frac{C(q)}{2} + D(q) + R(q)$$

3. After its checkpoint (for $1 \leq g \leq G - 1$):

$$\text{RE-EXEC}_{after\_ckpt} = D(q) + R(q) + \alpha \frac{G - g + 3}{2} C(q)$$

Overall average RE-EXEC: $\text{RE-EXEC}_{ckpt} =$

$$\frac{1}{G}((g-1).\text{RE-EXEC}_{before\_ckpt} + 1.\text{RE-EXEC}_{during\_ckpt} + (G-g).\text{RE-EXEC}_{after\_ckpt})$$

Average waste for failures during checkpointing phase

Average $\textsc{Re-Exec}$ when the failing-group g fails
Overall average $\textsc{Re-Exec}$: $\textsc{Re-Exec}_{ckpt} =$

$$\frac{1}{G}((g-1).\textsc{Re-Exec}_{before\_ckpt} + 1.\textsc{Re-Exec}_{during\_ckpt} + (G-g).\textsc{Re-Exec}_{after\_ckpt})$$

Average over all groups:

$$\textsc{avg\_Re-Exec}_{ckpt} =$$

$$D(q)+R(q)+\frac{G+1}{2G}T+\frac{\alpha C(q)(G+3)}{2}+\frac{C(q)(1-2\alpha)}{2G}-\frac{C(q)(G+1)}{2}$$

Average waste

$$\textsc{Waste}_{hierach} = \frac{T - \textsc{Work}}{T} + \frac{1}{\mu_p}\Big(D(q) + R(q) + \textsc{Re-Exec}\Big)$$

$$= \frac{1}{2\mu_p T} \times \left(\begin{array}{l} T^2 \\ +GC(q)\big[(1-\alpha)(2\mu_p - T) + (2\alpha - 1)C(q)\big] \\ +T\big[2(D(q) + R(q)) + (\alpha + 1)C(q)\big] \\ +(1-2\alpha)C(q)^2 \end{array} \right)$$

Minimize $\textsc{Waste}_{hierarch}$ subject to:

1. $GC(q) \leq T$ (by construction)
2. $T \leq 0.1\mu$ ($\Rightarrow Proba(Poisson(\frac{T}{\mu}) \geq 2) \leq 0.05$)

Outline

Impact on work

- ☹ Logging messages slows down execution:
  $\Rightarrow$ $\textsc{Work}$ becomes $\lambda\textsc{Work}$, where $0 < \lambda < 1$
  Typical value: $\lambda \approx 0.98$
- ☺ Re-execution after a failure is faster:
  $\Rightarrow$ $\textsc{Re-Exec}$ becomes $\frac{\textsc{Re-Exec}}{\rho}$, where $\rho \in [1..2]$
  Typical value: $\rho \approx 1.5$

$$\textsc{Waste}_{\text{hierarch}} = \frac{T - \lambda\textsc{Work}}{T} + \frac{1}{\mu_p}\left(D(q) + R(q) + \frac{\textsc{Re-Exec}}{\rho}\right)$$

Impact on checkpoint size

- Inter-groups messages logged continuously
- Checkpoint size increases with amount of work executed before a checkpoint

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

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

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

- Constraint $GC(q) \leq T$ translates into

$$GC_0(q)\beta\lambda\alpha \leq 1 \text{ and } T \geq \frac{GC_0(q)}{1 - GC_0(q)\beta\lambda\alpha}$$

Outline

1. Checkpointing protocols
2. Coordinated checkpointing
3. Hierarchical checkpointing
4. Accounting for message logging
5. Instanciating the model

Three case studies

Coord-IO

Coordinated approach: $C = C_{\text{Mem}} = \frac{\text{Mem}}{\text{b}_{io}}$
where Mem is the memory footprint of the application

Hierarch-IO

Several (large) groups, *I/O-saturated*
$\Rightarrow$ groups checkpoint sequentially

$$C_0(q) = \frac{C_{\text{Mem}}}{G} = \frac{\text{Mem}}{G\text{b}_{io}}$$

Hierarch-Port

Very large number of smaller groups, *port-saturated*
$\Rightarrow$ some groups checkpoint in parallel
q_{min} as the smallest value such that $\text{q}_{\text{min}}\text{b}_{port} \geq \text{b}_{io}$
Groups of q_{min} processors

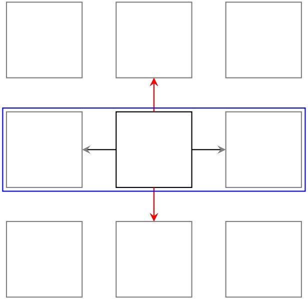
Three applications: 1) 2D-stencil

- Real matrix of size $n \times n$ partitioned across a $p \times p$ processor grid
- Each processor holds a matrix block of size $b = n/p$
- At each iteration:
  - average each matrix element with its 8 closest neighbors
  - exchange rows and columns that lie at partition boundary
  - each processor sends four messages of size b
- (Parallel) work for one iteration is $\textsc{Work} = \frac{9b^2}{s_p}$

Computing β for 2D-Stencil

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

Real $n \times n$ matrix and $p \times p$ grid
$Work = \frac{9b^2}{s_p}$, $b = n/p$
Each process sends a block to its 4 neighbors



HIERARCH-IO:

- 1 group = 1 grid row
- 2 out of the 4 messages are logged
- $\beta = \frac{Logged\_Msg}{C_0(q)\text{Work}} = \frac{2pb}{pb^2(9b^2/s_p)} = \frac{2s_p}{9b^3}$

HIERARCH-PORT:

- β doubles

Three applications: 2) 3D-stencil

- Real matrix of size $n \times n \times n$ partitioned across a $p \times p \times p$ processor grid
- Each processor holds a cube of size $b = n/p$
- At each iteration:
  - average each matrix element with its 27 closest neighbors
  - exchange the six faces of its cube
- (Parallel) work for one iteration is $\textsc{Work} = \frac{27b^3}{s_p}$

Three hierarchical variants

1. Hierarch-IO-Plane: group = horizontal plane of size p^2: $\beta = \frac{2s_p}{27b^3}$
2. Hierarch-IO-Line: group = horizontal line of size p: $\beta = \frac{4s_p}{27b^3}$
3. Hierarch-Port: groups of size q_{min} : $\beta = \frac{6s_p}{27b^3}$

Three applications: 3) Matrix product

- 3 real matrices of size $n \times n$ partitioned across a $p \times p$ processor grid
- Mem $= 24n^2$ (in bytes)
- Each processor holds three matrix blocks of size $b = n/p$
- At each iteration (Cannon's algorithm):
  - shift one block vertically and one horizontally
  - perform a matrix product
- (Parallel) work for one iteration is $\textsc{Work} = \frac{2b^3}{s_p}$

1. $\textsc{Hierarch-IO}$: one group per grid row: $\beta = \frac{s_p}{6b^3}$
2. $\textsc{Hierarch-Port}$: groups of size q_{min}: $\beta = \frac{s_p}{3b^3}$

Four platforms: basic characteristics

Name	Number of cores	Number of processors p_{total}	Number of cores per processor	Memory per processor	I/O Network Bandwidth (b_{io}) Read	Write	I/O Bandwidth (b_{port}) Read/Write per processor
Titan	299,008	16,688	16	32GB	300GB/s	300GB/s	20GB/s
K-Computer	705,024	88,128	8	16GB	150GB/s	96GB/s	20GB/s
Exascale-Slim	1,000,000,000	1,000,000	1,000	64GB	1TB/s	1TB/s	200GB/s
Exascale-Fat	1,000,000,000	100,000	10,000	640GB	1TB/s	1TB/s	400GB/s

Name	Scenario	G ($C(q)$)	β for 2D-STENCIL	β for MATRIX-PRODUCT
Titan	COORD-IO	1 (2,048s)	/	/
	HIERARCH-IO	136 (15s)	0.0001098	0.0004280
	HIERARCH-PORT	1,246 (1.6s)	0.0002196	0.0008561
K-Computer	COORD-IO	1 (14,688s)	/	/
	HIERARCH-IO	296 (50s)	0.0002858	0.001113
	HIERARCH-PORT	17,626 (0.83s)	0.0005716	0.002227
Exascale-Slim	COORD-IO	1 (64,000s)	/	/
	HIERARCH-IO	1,000 (64s)	0.0002599	0.001013
	HIERARCH-PORT	200,0000 (0.32s)	0.0005199	0.002026
Exascale-Fat	COORD-IO	1 (64,000s)	/	/
	HIERARCH-IO	316 (217s)	0.00008220	0.0003203
	HIERARCH-PORT	33,3333 (1.92s)	0.00016440	0.0006407

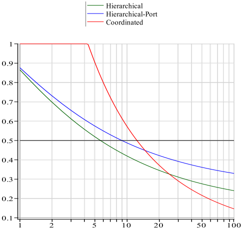
Checkpoint time

Name	C
K-Computer	14,688s
Exascale-Slim	64,000
Exascale-Fat	64,000

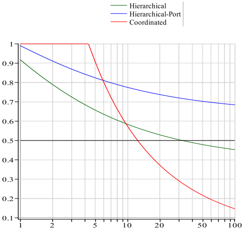
- Large time to dump the memory
- Using $1\% C$
- Comparing with $0.1\% C$ for exascale platforms
- $\alpha = 0.3$, $\lambda = 0.98$ and $\rho = 1.5$

Plotting formulas – Platform: Titan

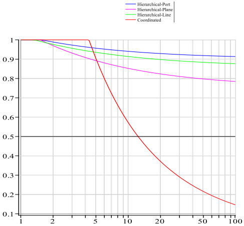
Stencil 2D



Matrix product



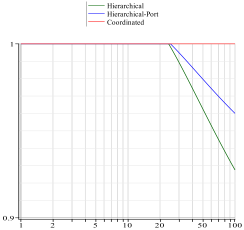
Stencil 3D



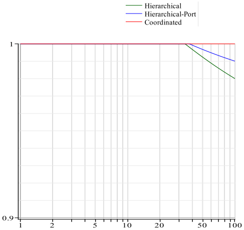
Waste as a function of processor MTBF μ_{ind}

Platform: K-Computer

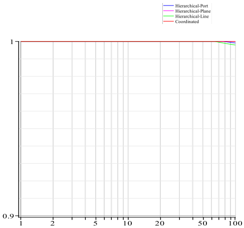
Stencil 2D



Matrix product



Stencil 3D



Waste as a function of processor MTBF μ_{ind}

Plotting formulas – Platform: Exascale

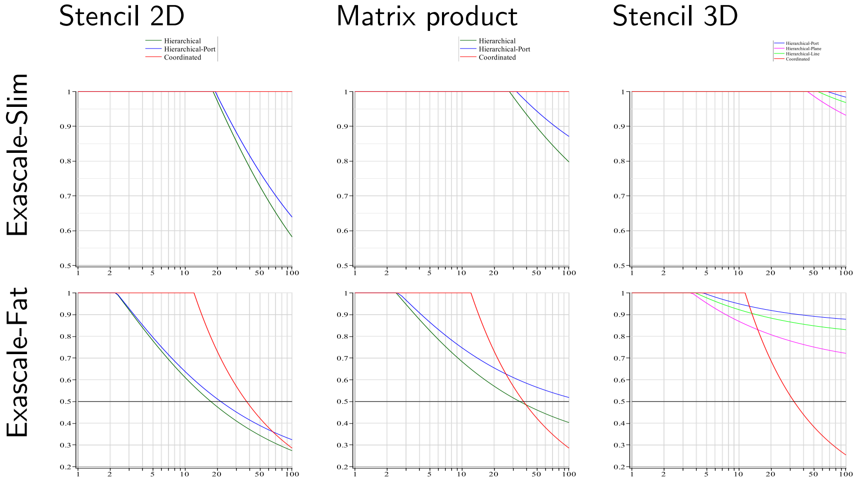
$\textsc{Waste} = 1$ for all scenarios!!!

Plotting formulas – Platform: Exascale

WASTE = 1 for all scenarios!!!

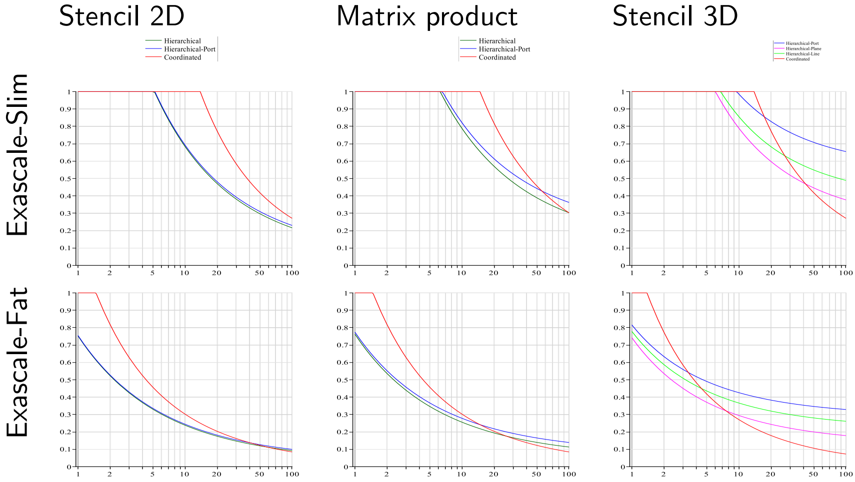
Goodbye Exascale?!

Plotting formulas – Platform: Exascale with $C = 1,000$



Waste as a function of processor MTBF μ_{ind}, $C = 1,000$

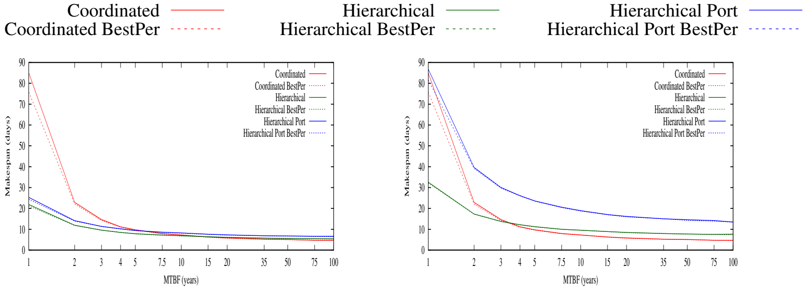
Plotting formulas – Platform: Exascale with $C = 100$



Waste as a function of processor MTBF μ_{ind}, $C = 100$

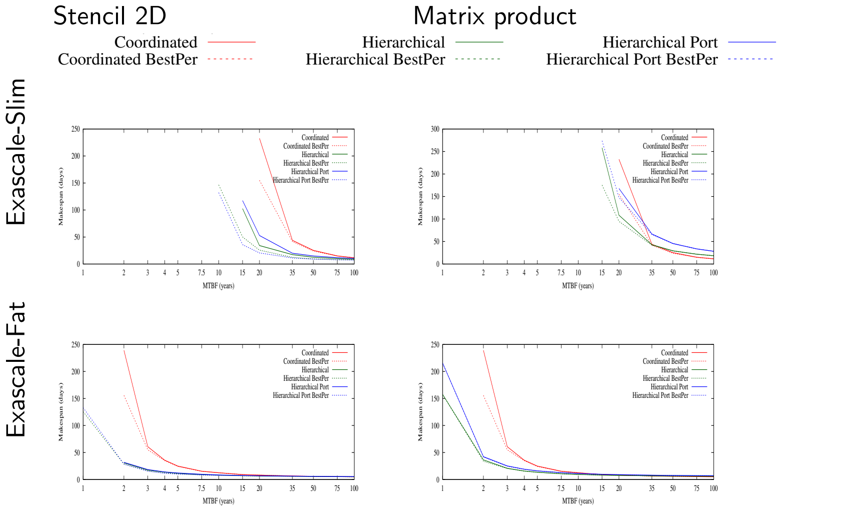
Simulations – Platform: Titan

Stencil 2D Matrix product



Makespan (in days) as a function of processor MTBF μ_{ind}

Simulations – Platform: Exascale with $C = 1,000$

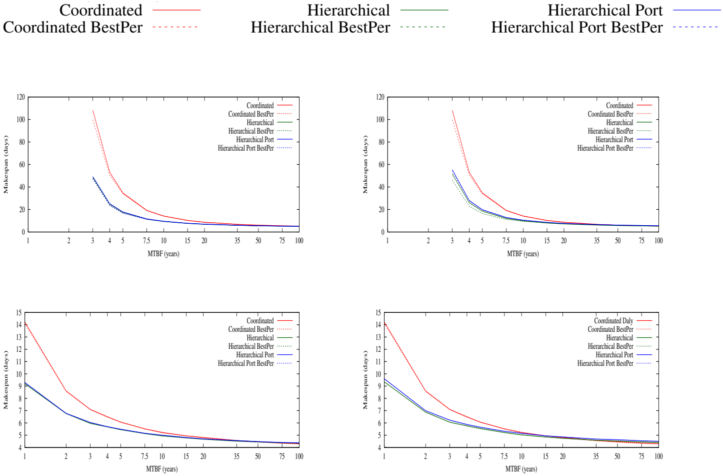


Makespan (in days) as a function of processor MTBF μ_{ind}, $C = 1,000$

Simulations – Platform: Exascale with $C = 100$

Stencil 2D | Matrix product

Exascale-Slim | Exascale-Fat



Makespan (in days) as a function of processor MTBF μ_{ind}, $C = 100$