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

June 13, 2012

Motivation

Framework

- **Very very** large number of processing elements (e.g., 2^{20})
  - The probability of failures increases
- Large application to be executed on the whole platform

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

- Resilience provided through checkpointing
  1. Coordinated protocols
  2. Hierarchical protocols

Which checkpointing protocol to use?

Coordinated checkpointing

- ☺ No risk of cascading rollbacks
- ☺ No need to log messages
- ☹ All processors need to roll back
- ☹ May 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 roll back
- ☺ Faster re-execution with logged messages
- ☺ Should scale to very large platforms

Outline

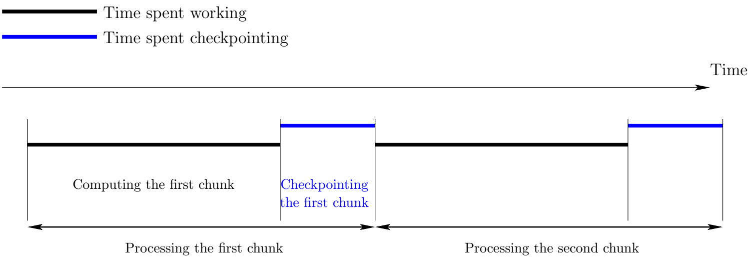
Outline

Framework

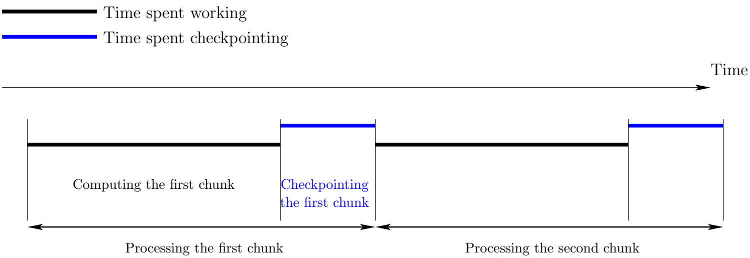
- Periodic checkpointing policies (of period T)
- Independent and identically distributed failures
- Platform failure inter-arrival time: μ
- Tightly-coupled application: progress $\Leftrightarrow$ all processors available
- First-order approximation: at most one failure within a period

Waste: fraction of time not spent for useful computations

Checkpointing cost

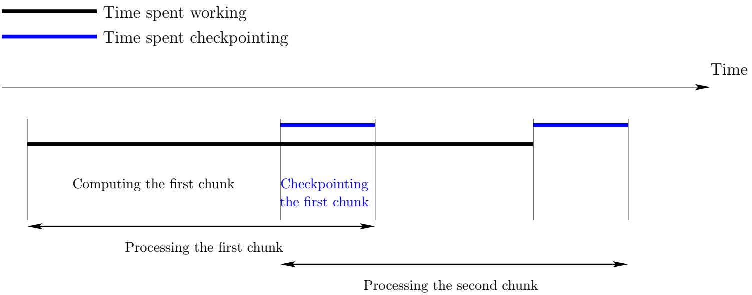


Checkpointing cost



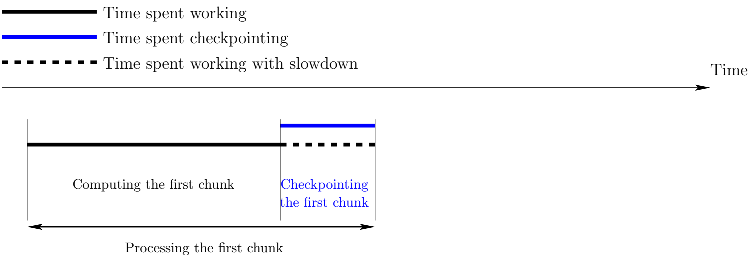
Blocking model: while a checkpoint is taken, no computation can be performed

Checkpointing cost



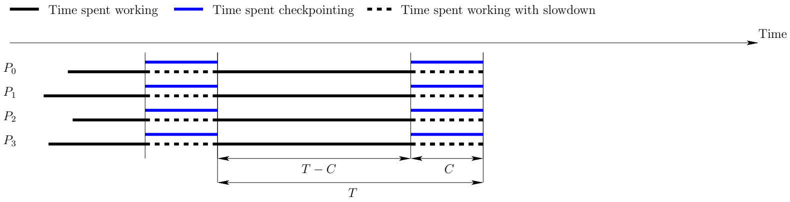
Non-blocking model: while a checkpoint is taken, computations are not impacted (e.g., first copy state to RAM, then copy RAM to disk)

Checkpointing cost



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 $(0 \leq \alpha \leq 1)$.

Waste in absence of failures

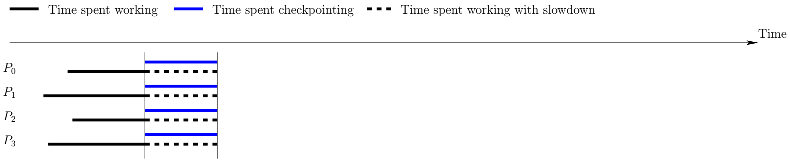


Time elapsed since last checkpoint: T

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

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

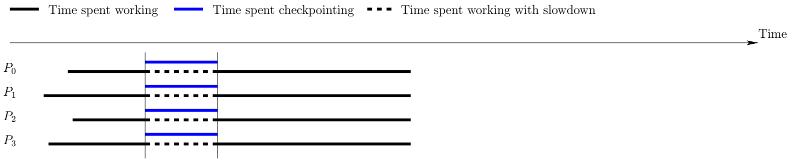
Waste due to failures



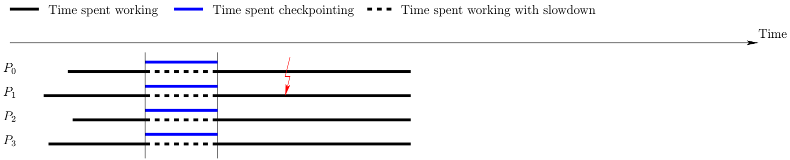
Failure can happen

1. During computation phase
2. During checkpointing phase

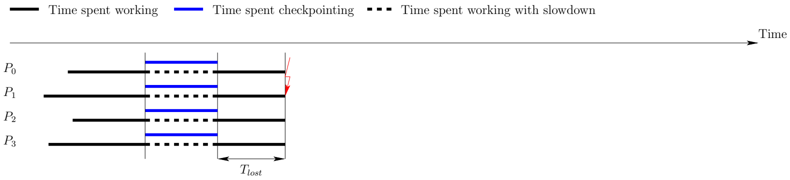
Waste due to failures



Waste due to failures

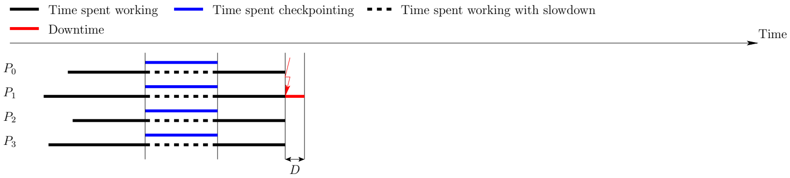


Waste due to failures

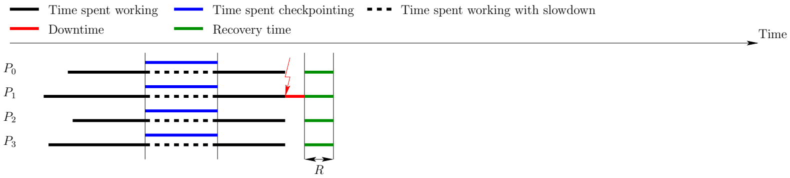


Coordinated checkpointing protocol: when one processor is victim of a failure, all processors lose their work and must roll back to last checkpoint

Waste due to failures in computation phase

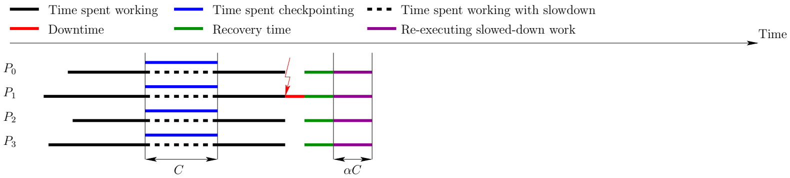


Waste due to failures in computation phase



Coordinated checkpointing protocol: All processors must recover from last checkpoint

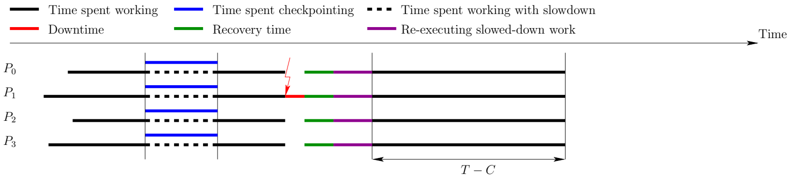
Waste due to failures in computation phase



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

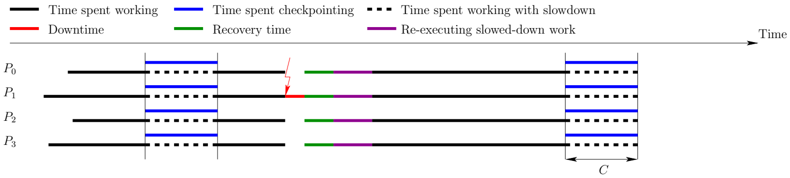
But no checkpoint is taken in parallel, hence this re-computation is faster than the original computation

Waste due to failures in computation phase



Re-execute the computation phase

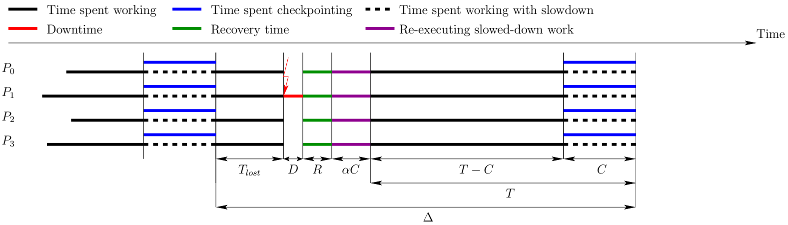
Waste due to failures in computation phase



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: $\Delta - T = T_{lost} + \alpha C$

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

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

Waste due to failures

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

$$\textsc{Re-Exec}_{coord-fail-in-work} = \frac{T-C}{2} + \alpha C$$

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

$$\textsc{Re-Exec}_{coord-fail-in-checkpoint} = T - \frac{C}{2} + \alpha C$$

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

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

Overall waste

$$\text{WASTE}_{coord} = \text{WASTE}_{coord-nofailure} + \frac{1}{\mu}(D + R + \text{RE-EXEC}_{coord})$$

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

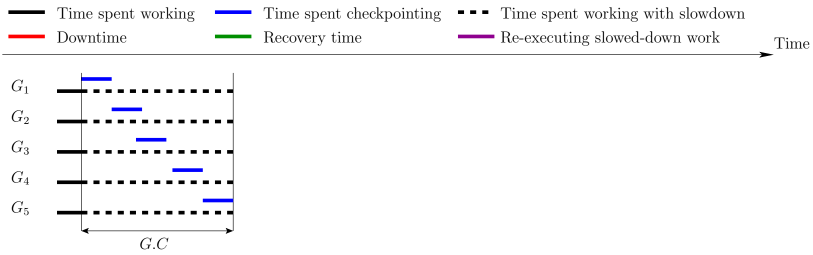
Minimize WASTE_{coord} subject to:

- $C \leq T$ (by construction)

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

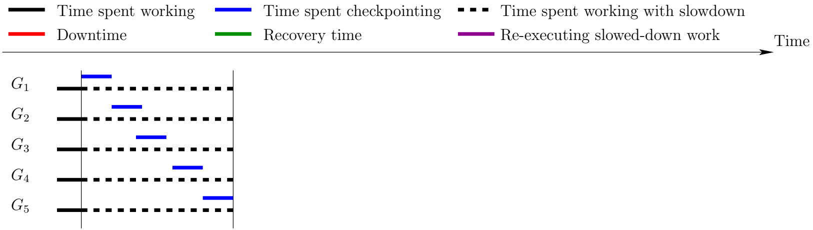


When a group checkpoints, its own computation speed is slowed-down

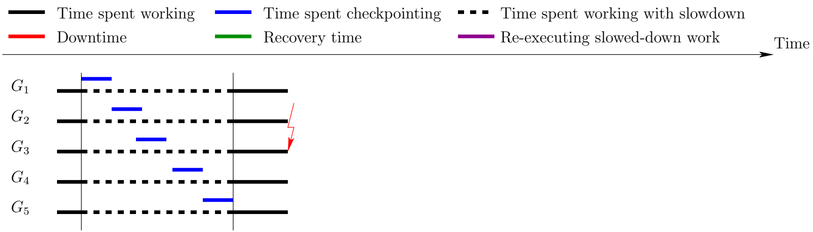
This holds for all groups because of the tightly-coupled assumption

$$\text{WASTE} = \frac{T - \text{WORK}}{T} \text{ where } \text{WORK} = T - (1-\alpha)GC(q)$$

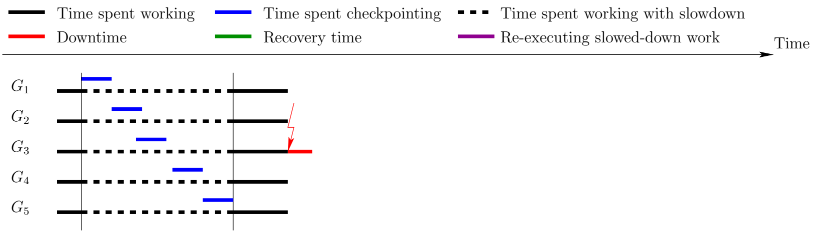
Impact of checkpointing



Impact of checkpointing

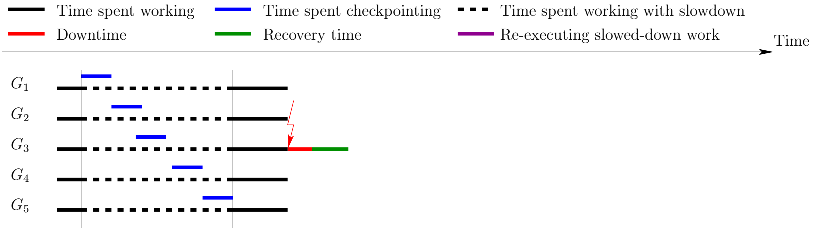


Impact of checkpointing



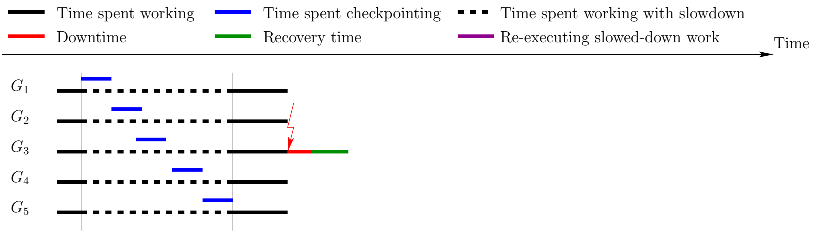
Tightly-coupled model: while one group is in downtime, none can work

Failure during computation phase



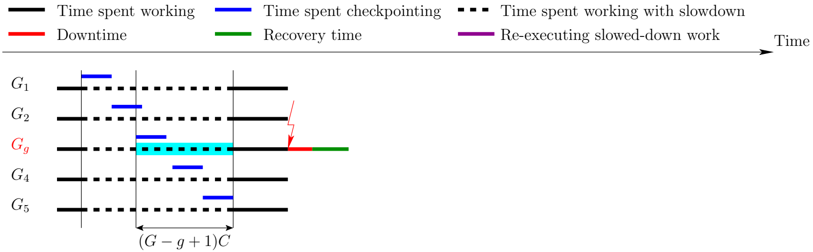
Tightly-coupled model: while one group is in recovery, none can work

Failure during computation phase



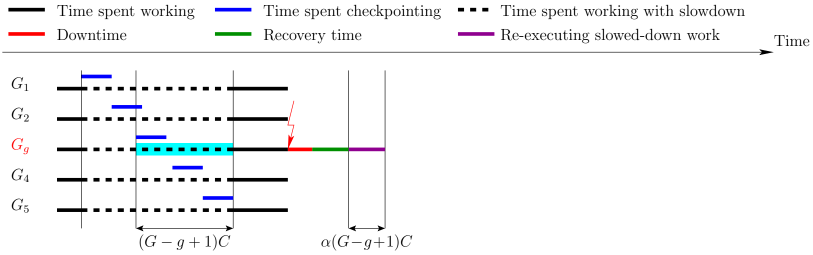
Groups must have completed the same amount of work in between two consecutive checkpoints, independently of the fact that a failure may or may not have happened on the platform in between these checkpoints. Hence, no checkpointing is possible during the rollback.

Failure during computation phase



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

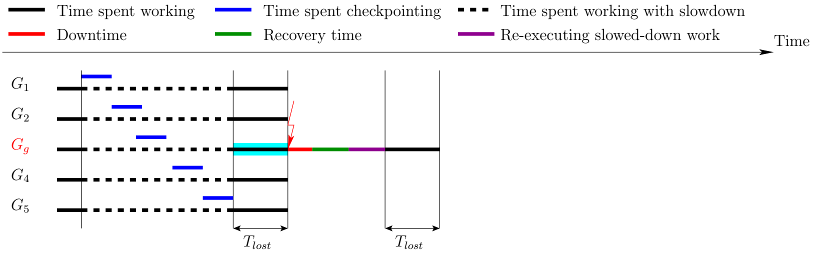
Failure during computation phase



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

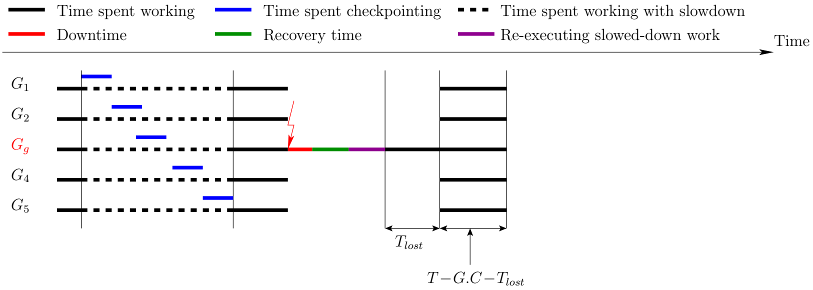
But no checkpoint is taken in parallel, hence this re-computation is faster than the original computation

Failure during computation phase



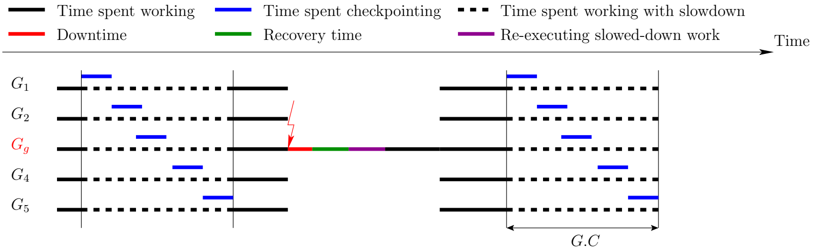
Redo work done in computation phase and that was destroyed by the failure

Failure during computation phase



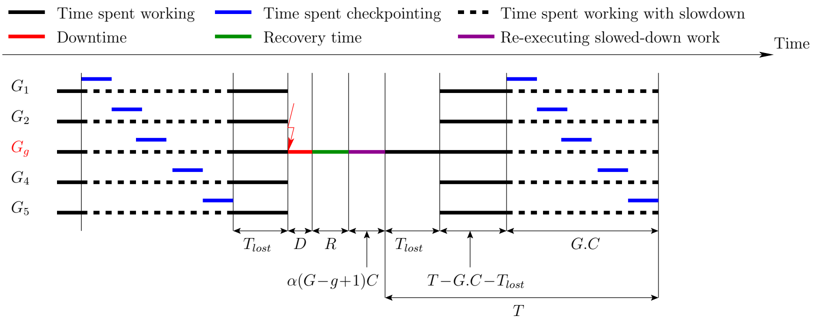
Failing group has reached the point where it previously failed, all groups now resume execution in parallel and complete the computation phase

Failure during computation phase



Finally, perform checkpointing phase

Failure during computation phase

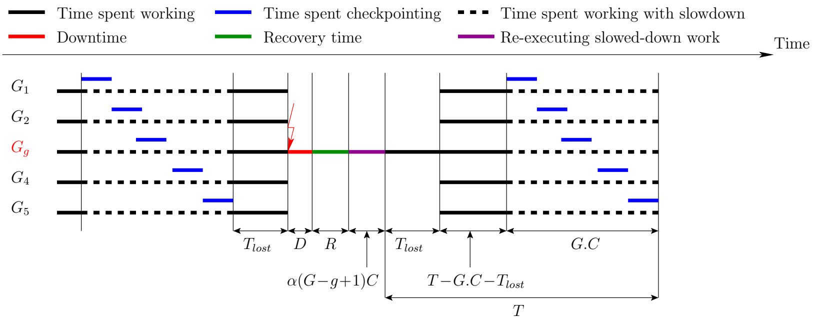


RE-EXEC: $T_{lost} + \alpha(G-g+1)C$

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

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

Failure during computation phase

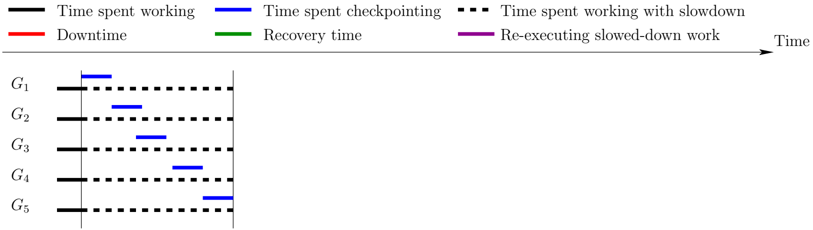


Approximated RE-EXEC: $\dfrac{T-G.C}{2}+\alpha(G-g+1)C$

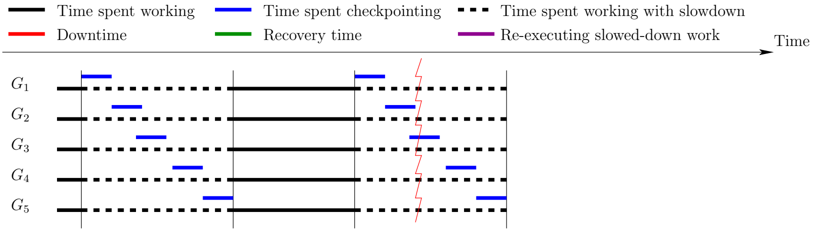
Average approximated RE-EXEC:

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

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

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:

$$\textsc{avg\_Re-Exec}_{ckpt} = \frac{G+1}{2G}T + \frac{\alpha C(q)(G+3)}{2} + \frac{C(q)(1-2\alpha)}{2G} - \frac{C(q)(G+1)}{2}$$

Average waste

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

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

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

- $GC(q) \leq T$ (by construction)

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 $\dfrac{\textsc{Re-Exec}}{\rho}$, where $\rho \in [1..2]$
  Typical value: $\rho \approx 1.5$

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

Impact on checkpoint size

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

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

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

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

- 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

Two 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

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

1. Protocols Cost
   - Coordinated checkpointing
   - Hierarchical checkpointing

2. Accounting for message logging

3. Instanciating the model
   - Applications
   - Platforms

4. Plotting the formulas

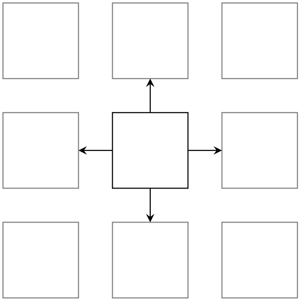
Three applications

1. 2D-stencil
2. 3D-Stencil
   - Plane
   - Line
3. Matrix product

Computing β for Stencil-2D

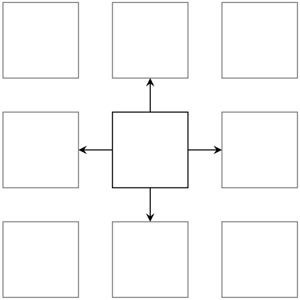
$$C(q) = C_0(q) + Logged\_Msg = C_0(q)(1 + \beta \textsc{Work})$$

- Real matrix $n \times n$
- $Mem = 8n^2$



Computing β for Stencil-2D

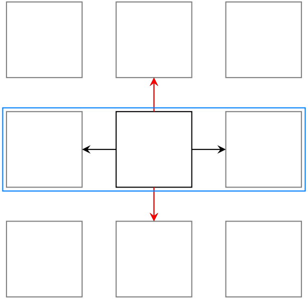
$$C(q) = C_0(q) + Logged\_Msg = C_0(q)(1 + \beta \text{WORK})$$



- $Mem = 8n^2$
- s_p: speed of the process
- b: block size
- Block update: 9 floating points operations
- Each process holds a block of size b^2
- $Work = \frac{9b^2}{\mathsf{s}_p}$

Computing β for Stencil-2D

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

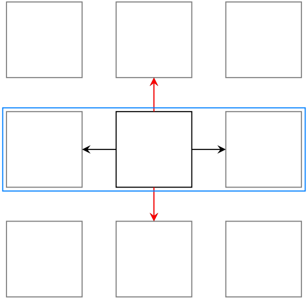


- $Mem = 8n^2$
- $Work = \frac{9b^2}{\mathsf{s}_p}$
- Each process sends a block to its 4 neighbors 1 group = 1 line
    - 2 out of the 4 messages are logged

Computing β for Stencil-2D

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

- $Mem = 8n^2$
- $Work = \frac{9b^2}{\mathsf{s}_p}$
  - 2 out of the 4 messages are logged
  - $\beta = \frac{2\mathsf{s}_p}{9b^3}$



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	I/O Network Bandwidth (b_{io}) Write
Titan	299,008	16,688	16	32GB	300GB/s	300GB/s
K-Computer	705,024	88,128	8	16GB	150GB/s	96GB/s
Exascale-Slim	1,000,000,000	1,000,000	1,000	64GB	1TB/s	1TB/s
Exascale-Fat	1,000,000,000	100,000	10,000	640GB	1TB/s	1TB/s

Four platforms: 2D-Stencil and Matrix-Product

Name	Scenario	$G\ (C(q))$	β for 2D-Stencil	β for Matrix-Product
Titan	Coord-IO Hierarch-IO	1 (2,048s) 136 (15s)	/ 0.0001098	/ 0.0004280
K-Computer	Coord-IO Hierarch-IO	1 (14,688s) 296 (50s)	/ 0.0002858	/ 0.001113
Exascale-Slim	Coord-IO Hierarch-IO	1 (64,000s) 1,000 (64s)	/ 0.0002599	/ 0.001013
Exascale-Fat	Coord-IO Hierarch-IO	1 (64,000s) 316 (217s)	/ 0.00008220	/ 0.0003203

Four platforms: 3D-STENCIL

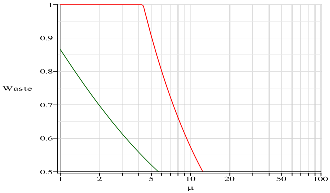
Name	Scenario	G	β for 3D-STENCIL
Titan	COORD-IO	1	/
	HIERARCH-IO-PLANE	26	0.001476
	HIERARCH-IO-LINE	675	0.002952
K-Computer	COORD-IO	1	/
	HIERARCH-IO-PLANE	44	0.003422
	HIERARCH-IO-LINE	1,936	0.006844
Exascale-Slim	COORD-IO	1	/
	HIERARCH-IO-PLANE	100	0.003952
	HIERARCH-IO-LINE	10,000	0.007904
Exascale-Fat	COORD-IO	1	/
	HIERARCH-IO-PLANE	46	0.001834
	HIERARCH-IO-LINE	2,116	0.003668

Outline

Platform Titan

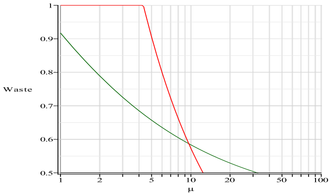
2D-STENCIL

Hierarchical
Coordinated



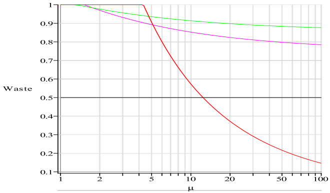
MATRIX-PRODUCT

Hierarchical
Coordinated



3D-STENCIL

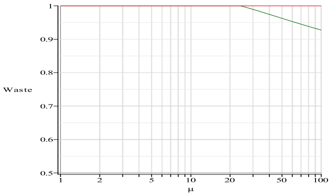
Hierarchical-Plane
Hierarchical-Line
Coordinated



Platform K-computer

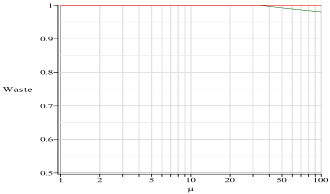
2D-Stencil

Hierarchical
Coordinated



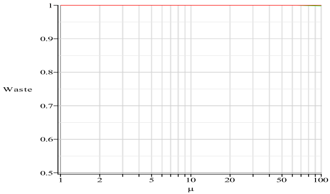
Matrix-Product

Hierarchical
Coordinated



3D-Stencil

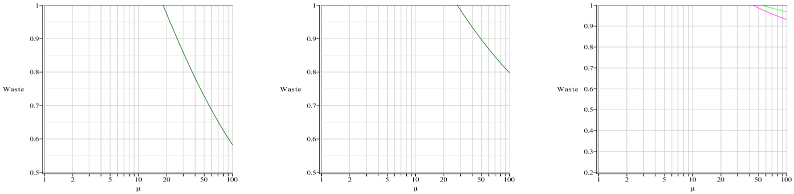
Hierarchical-Plane
Hierarchical-Line
Coordinated



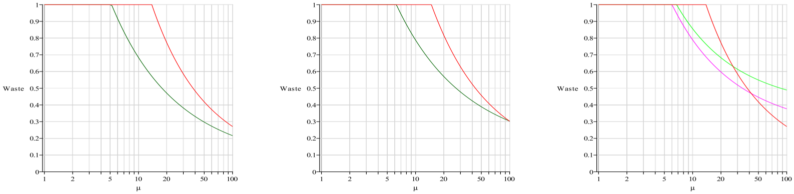
Platform Exascale-Slim

- Coordinated checkpoint: $C = 64,000$
- No progress can be made
    - $C = 1,000$ s

2D-Stencil | Matrix-Product | 3D-Stencil



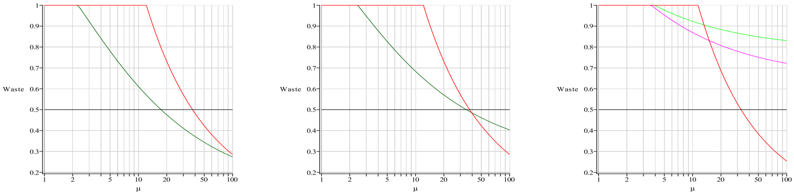
- $C = 100$ s



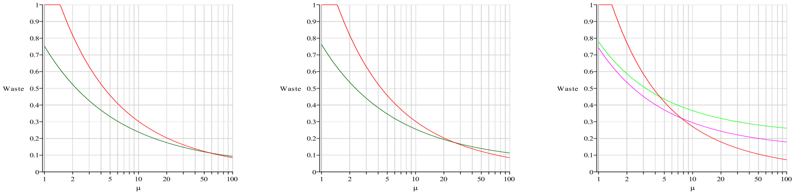
Platform Exascale-Fat

- Coordinated checkpoint: $C = 64,000$
- No progress can be made
  - $C = 1,000$ s

2D-Stencil Matrix-Product 3D-Stencil



  - $C = 100$ s



Conclusion and future work

1. Conclusion
    - First attempt at analytical comparison of coordinated and hierarchical checkpointing protocols
        - Message logging impact
        - Checkpointing impact
2. Current work
    - Simulation analysis
3. Future work
    - Model extension: Energy

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

June 13, 2012