Checkpointing strategies: Towards exascale

ROMA project-team

January 13, 2015

Failures happen often... but how do we cope with them?

(Not so) Secret data

- Tsubame 2: 962 failures during last 18 months so $\mu = 13$ hrs
- Blue Waters: 2-3 node failures per day
- Titan: a few failures per day
- Tianhe 2: wouldn't say

The question is: Given an application and a platform, which tolerance solutions is the best? How should it be used?

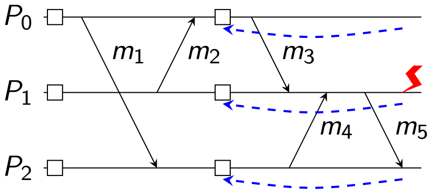
Many proposed fault-tolerance solutions but...

- Experiments are impossible
  - Experiments on petascale machines are too expensive
  - Exascale platforms does not exist yet
  - We do not know (exactly) what exascale platforms will be

Need for modelization, analysis, and simulation

Background: coordinated checkpointing protocols

- Coordinated checkpoints over all processes
- Global restart after a failure



Fault Prediction and Coordinated Checkpointing

Fault predictor

- Imperfect predictor
- $Recall = \frac{Predicted\ faults}{Faults}$: percentage of faults predicted
- $Precision = \frac{Predicted\ faults}{Predictions}$: percentage of predictions corresponding to faults
- Predicted "time" of failure: either exact date or time interval

Questions

- Should predictions always be trusted?
- How do predictions impact checkpointing policies?
- Is it always beneficial to use a predictor?

Fault Prediction and Coordinated Checkpointing

Predicted “time” of failure = exact date

- First-order analysis
- Optimal algorithms to decide whether and when to take predictions into account
- Optimal value of the checkpointing period
- Recall is more important than precision

Predicted “time” of failure = time interval

- New approach with two periodic modes: one outside prediction windows, and one inside prediction windows
- Optimal checkpointing periods
- Results of the analytical study are corroborated by simulations (validity of model and accuracy of approach)

Silent Error Detection and Coordinated Checkpointing

Context

- No immediate detection of silent errors
- Necessity of a detection mechanism
- Two models
  - Errors detected (by an oracle) after a delay
  - Errors detected through a user-initiated verification mechanism

Questions

- First model: impact of detection latency on checkpointing policy?
- Second model: when to invoke the verification mechanism?

Silent Error Detection and Coordinated Checkpointing

Errors detected (by an oracle) after a delay

- Exponential failure and latency distributions: no impact of latency distribution on optimal checkpointing strategy
- Finite memory: lower bound on period to guarantee that at least one valid checkpoint is live (within a risk threshold)

Errors detected through a user-initiated verification mechanism

- Either k checkpoints for one verification or
  k verifications for one checkpoint
- Analytical formula for the waste
- Optimal checkpointing and verification periods

Detailed result

A unified model for assessing checkpointing protocols at extreme-scale

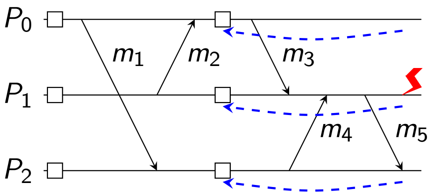
by George Bosilca, Aurélien Bouteiller, Élisabeth Brunet, Franck Cappello, Jack Dongarra, Amina Guermouche, Thomas Hérault, Yves Robert, Frédéric Vivien, Dounia Zaidouni

Journal of Concurrency and Computation: Practice and Experience, Wiley InterScience, 2013, DOI: 10.1002/cpe.3173

`https://hal.inria.fr/hal-00908447`

Background: coordinated checkpointing protocols

- Coordinated checkpoints over all processes
- Global restart after a failure



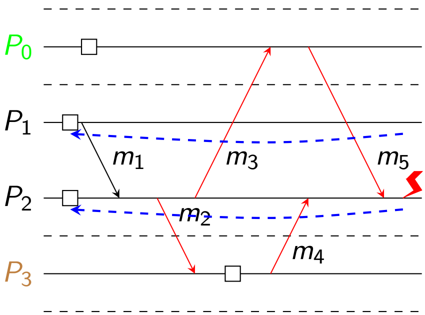
☺ No risk of cascading rollbacks

☺ No need to log messages

☹ All processors need to roll back

Background: hierarchical protocols

- Clusters of processes
- Coordinated checkpointing protocol within clusters
- Message logging protocols between clusters
- Only processors from failed group need to roll back



☹ Need to log inter-groups messages
- Slowdowns failure-free execution
- Increases checkpoint size/time

☺ Faster re-execution with logged messages

Which checkpointing protocol to use?

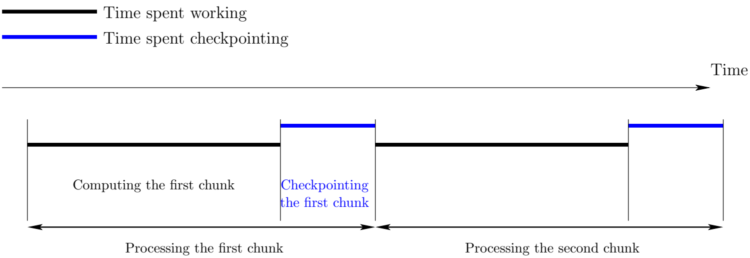
Coordinated checkpointing

- ☺ No risk of cascading rollbacks
- ☺ No need to log messages
- ☹ All processors need to roll back
- ☹ Rumor: May not scale to very large platforms

Hierarchical checkpointing

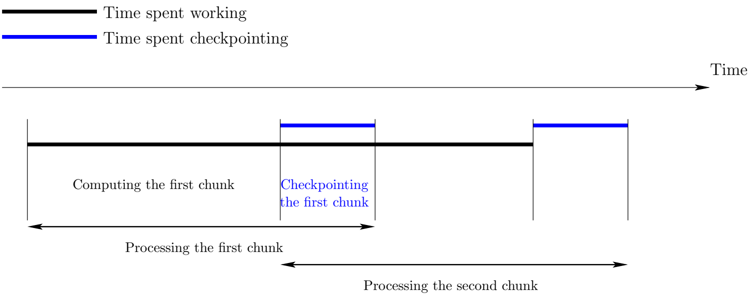
- ☹ Need to log inter-groups messages
  - Slowdowns failure-free execution
  - Increases checkpoint size/time
- ☺ Only processors from failed group need to roll back
- ☺ Faster re-execution with logged messages
- ☺ Rumor: Should scale to very large platforms

Coordinated checkpointing



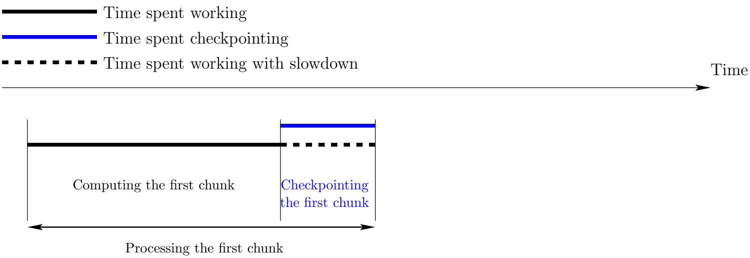
Blocking model: checkpointing blocks all computations

Coordinated checkpointing



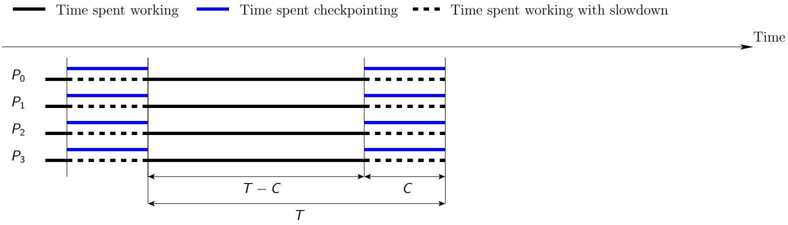
Non-blocking model: checkpointing has no impact on computations (e.g., first copy state to RAM, then copy RAM to disk)

Coordinated checkpointing



General model: checkpointing slows computations 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 fault-free execution

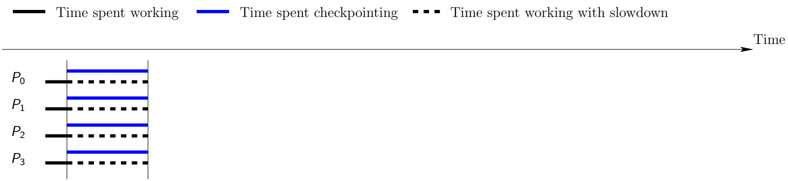


Time elapsed since last checkpoint: T

Amount of computations executed: $\textsc{Work} = (T - C) + \alpha C$

$\textsc{Waste}[FF] = \frac{T - \textsc{Work}}{T} = \frac{(1-\alpha)}{T}$

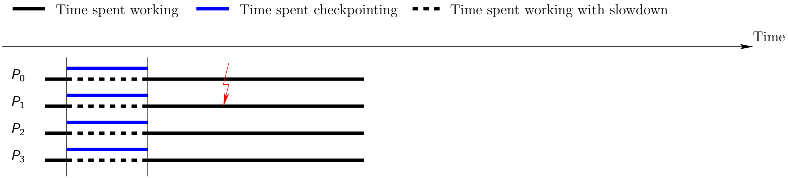
Waste due to failures



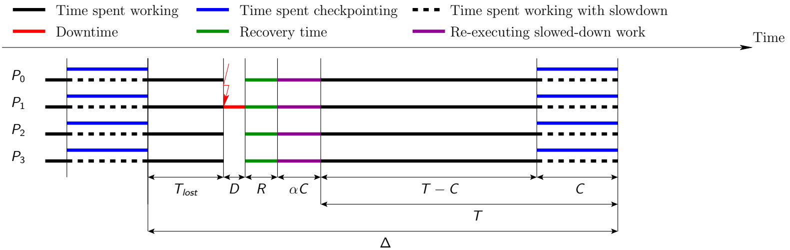
Failure can happen

1. During computation phase
2. During checkpointing phase

Waste due to failures in computation phase



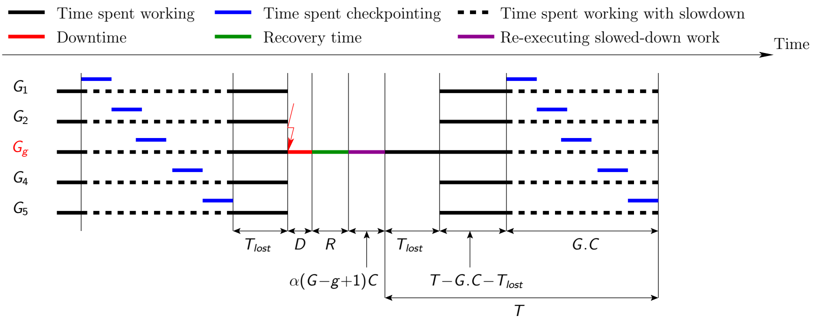
Total waste



$$\text{WASTE}[fail] = \frac{1}{\mu}\left(D + R + \alpha C + \frac{T}{2}\right)$$

Optimal period $T_{\text{opt}} = \sqrt{2(1-\alpha)(\mu-(D+R+\alpha C))C}$

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

Accounting for message logging: 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: $\rho \approx 1.5$

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

$$\textsc{Waste}[fail] = \frac{1}{\mu}\left(D(q) + R(q) + \frac{\textsc{Re-Exec}}{\rho}\right)$$

Accounting for message logging: 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

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

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

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

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
Groups of q_{min} processors, where $\text{q}_{\text{min}}\text{b}_{port} \geq \text{b}_{io}$

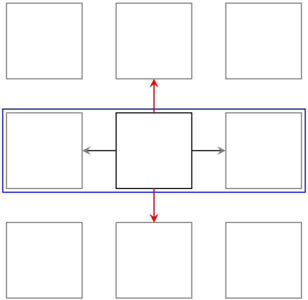
Three applications

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

Computing β for 2D-Stencil

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

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
- $\beta = \frac{Logged\_Msg}{C_0(q)\textsc{Work}} = \frac{2pb}{pb^2(9b^2/s_p)} = \frac{2s_p}{9b^3}$

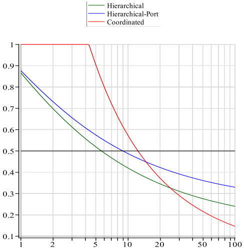
Four platforms: basic characteristics

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	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

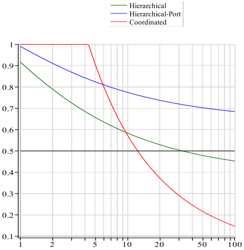
Name	Scenario	G ($C(q)$)	β for 2D-STENCIL	β for MATRIX-PRODUCT
Titan	COORD-IO	1 (2,048s)	/	/
	HIERARCH-IO	136 (15s)	0.0001098	0.0004280
	HIERARCH-PORT	1,246 (1.6s)	0.0002196	0.0008561
K-Computer	COORD-IO	1 (14,688s)	/	/
	HIERARCH-IO	296 (50s)	0.0002858	0.001113
	HIERARCH-PORT	17,626 (0.83s)	0.0005716	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

Plotting formulas – Platform: Titan

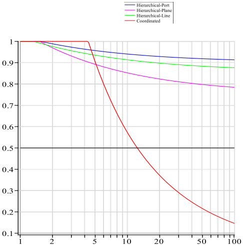
Stencil 2D



Matrix product



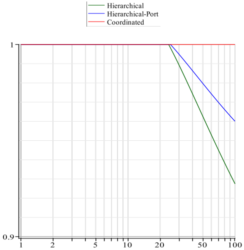
Stencil 3D



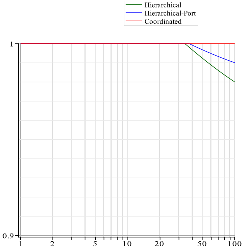
Waste as a function of processor MTBF μ_{ind}

Platform: K-Computer

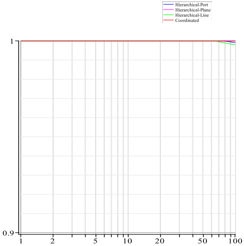
Stencil 2D



Matrix product



Stencil 3D



Waste as a function of processor MTBF μ_{ind}

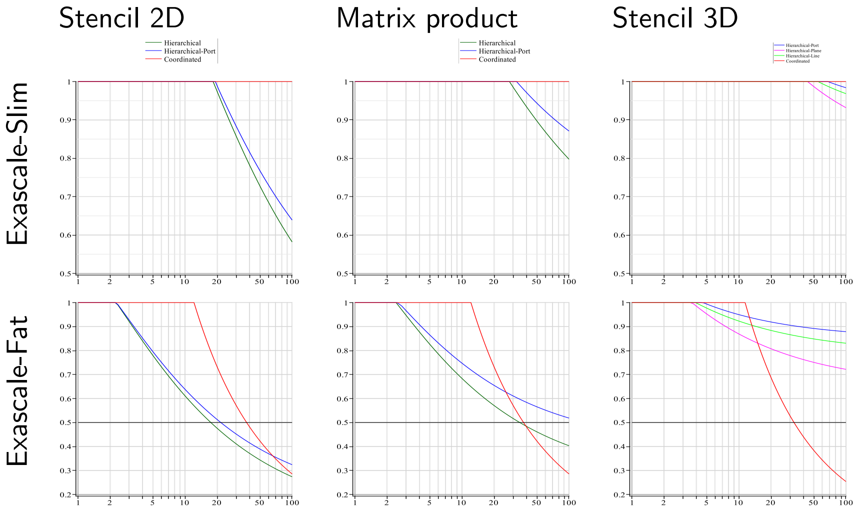
Plotting formulas – Platform: Exascale

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

WASTE = 1 for all scenarios!!!

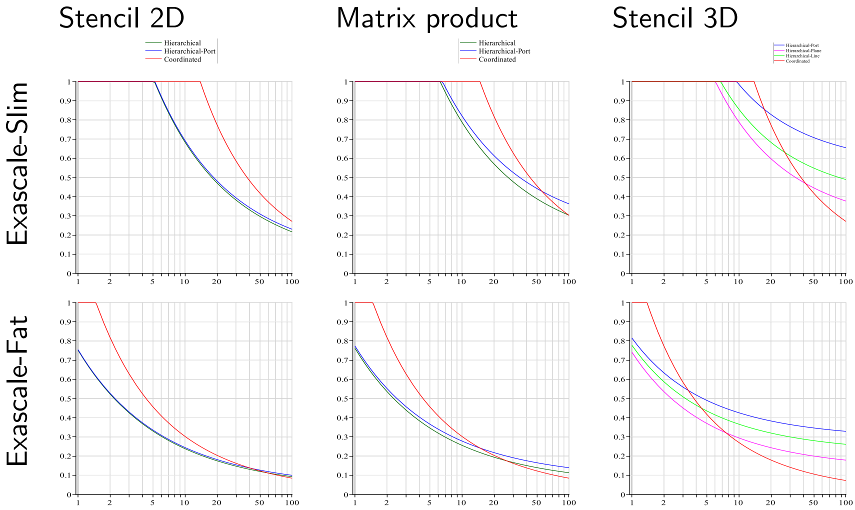
Goodbye Exascale?!

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



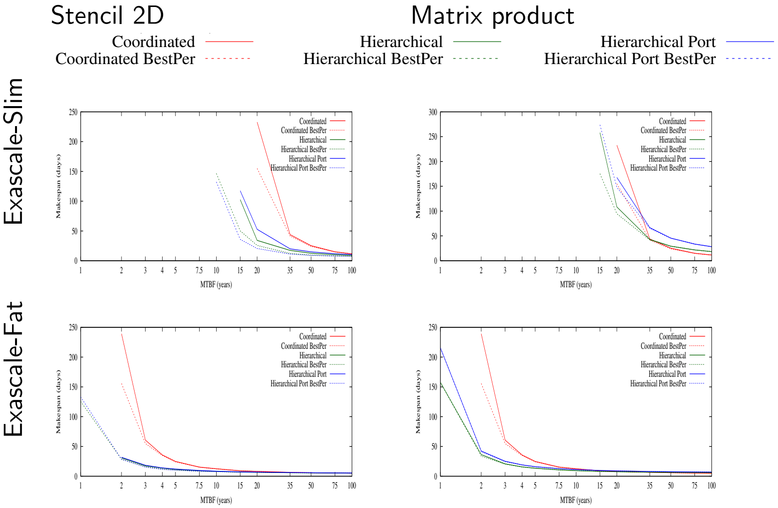
Waste as a function of processor MTBF μ_{ind}, $C = 1,000$

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



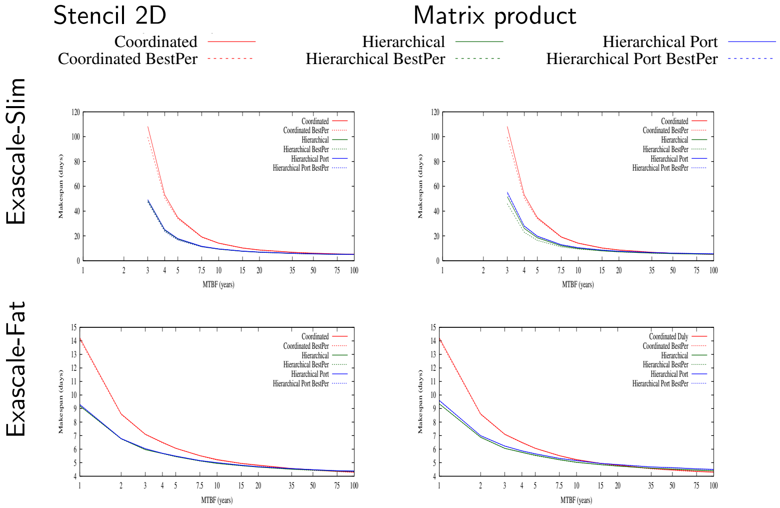
Waste as a function of processor MTBF μ_{ind}, $C = 100$

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



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

Simulations – Platform: Exascale with $C = 100$



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

Subjects addressed

- Combining silent error detection and checkpointing
- Checkpointing algorithms and fault prediction
- A unified model for assessing checkpointing protocols at extreme-scale
- Multi-criteria checkpointing strategies: Optimizing response-time versus resource utilization
- Optimal checkpointing period: Time vs. energy
- Revisiting the double checkpointing algorithm
- Using group replication for resilience on exascale systems
- Assessing the Impact of ABFT and Checkpoint Composite Strategies