Fault-tolerant numerical iterative algorithms at scale

Alix Tremodeux

June 2024
ENS de Lyon

Supervisors: Anne Benoit, Yves Robert
Emmanuel Agullo, Luc Giraud, Thomas Herault

Plan

Introduction

Initially: We have an iterative algorithm (i.e., an initialization and then a repeated code, one repetition is an iteration) that is executed on a large-scale platform.

Issue: Errors of different types can strike during the execution.

Aim: Protect the execution of the iterative algorithm in an optimized way (better in terms of cost than to check for error at each iteration).

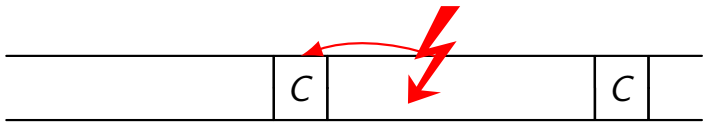
Types of errors

We consider different types of errors:

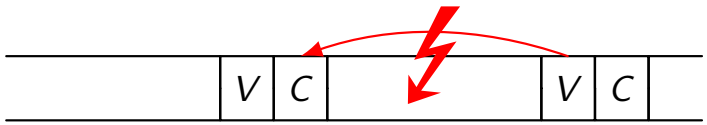
- **Fail-stop error**: When it strikes, the system crashes (immediately)
- **Silent error**: If not corrected, Wrong output. Two different types:
  - Computation error: For $a + b$, the output is c' instead of c ($a + b = c$)
  - Memory error: Bit-flip in memory (affects static data)

Mitigation techniques

- Fail-stop error: Global checkpoint



- Silent error: Local checkpoint and verifications:



Aim of this work

Protect the iterative algorithm against errors:

- A model with verifications, checkpoints and rollbacks
- Not specific to an iterative algorithm
- But easy to optimize and to execute

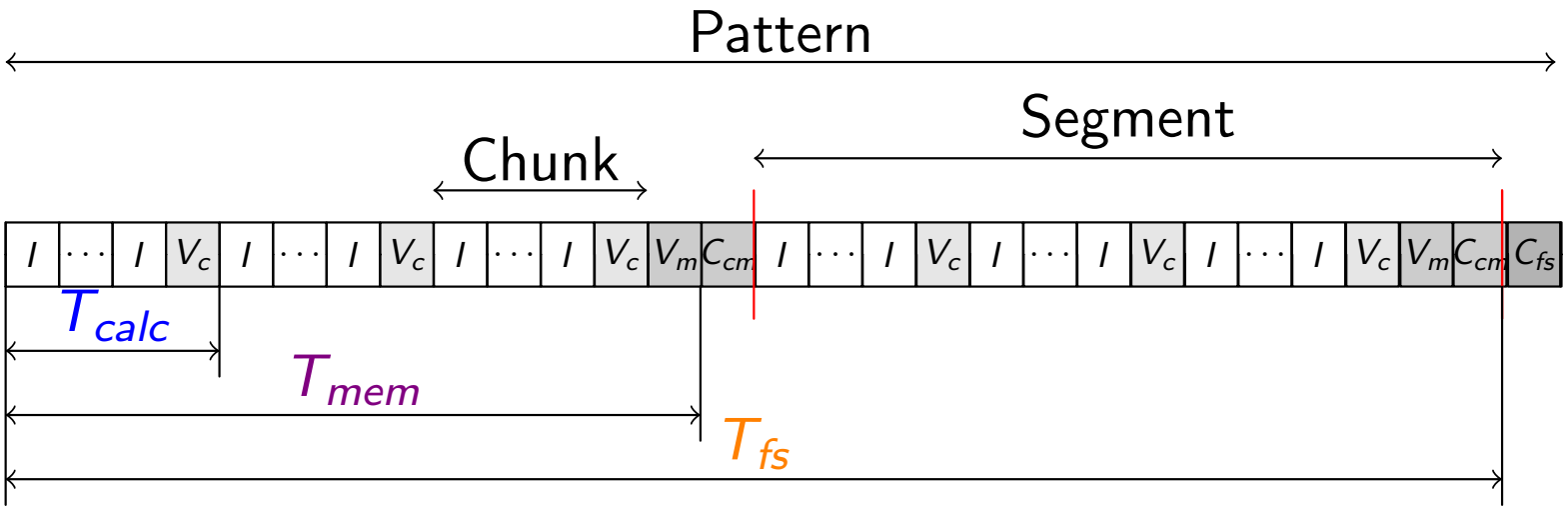
Hence:

- The model is a periodic pattern (easy to implement and not bound to the iterative algorithm)
- Optimize the expected time of the pattern

Often, when we protect an iterative algorithm, verifications and checkpoints (at least for silent errors) are done at each iteration of the algorithm. We consider this approach as the naive approach.

Plan

The pattern



$T_{calc} = n_{vc} I + V_c$
$T_{mem} = n_{cm}(T_{calc}) + V_m$
$T_{fs} = n_{fs}(T_{mem} + C_{cm})$

n_{vc}: number of iteration in a chunk
n_{cm}: number of chunks
n_{fs}: number of segments

Hypotheses

We need some hypotheses to create the model:

- All the iterations have the same size
- Recall and precision of verification is 1 (no false positive or negative)
- No silent error strikes during checkpoint and verification
- No fail-stop error strikes during the fail-stop checkpoint

Expected time of the pattern

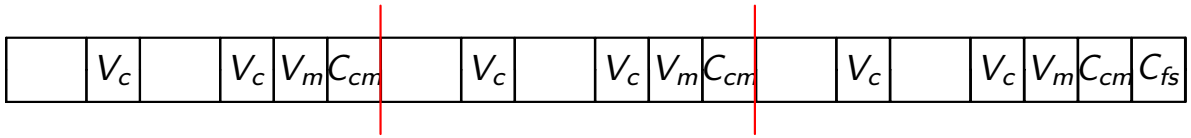
Let E be the expected time of the pattern, it depends on n_{fs}, n_{cm}, n_{vc} (resp. number of iterations in a chunk, number of chunks, number of segments in the pattern).

Useful work in the pattern (execution of the iterative algorithm): $W = n_{fs} n_{cm} n_{vc} I$.

Goal: find expected time E and then minimize the overhead

$$\min \frac{E(n_{fs}, n_{cm}, n_{vc})}{n_{fs} n_{cm} n_{vc} I}$$

Computation of the expected time

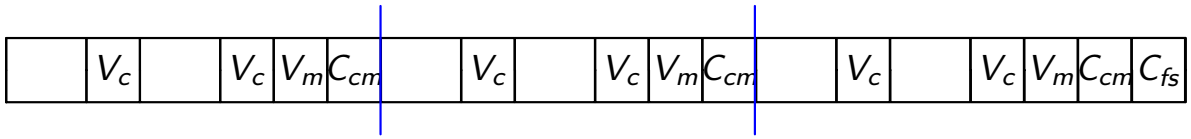


$$E = \sum_{k=0}^{n_{fs}-1} E_k + C_{fs}$$

E_k is the expected time of the kth segment

Expected time of the segment

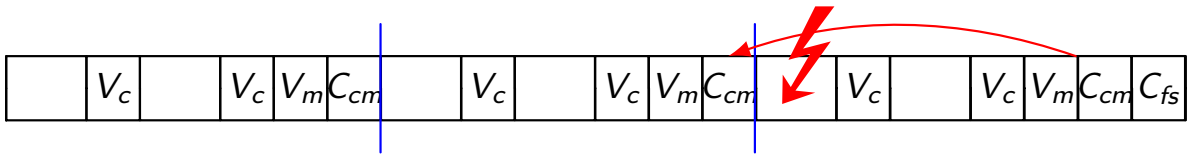
Case 1: No errors



$$\Rightarrow P_{succ}^{fs,mem,calc}[T_{mem} + C_{cm}]$$

Expected time of the segment

Case 2: The first error to be detected is a memory error

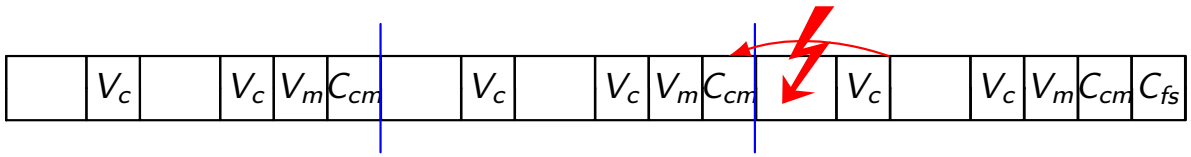


- No computation error strikes during the execution of the segment (otherwise it would have been detected)
- No fail-stop error strikes during the execution of the segment(it would have been detected)

$$\Rightarrow P_{fail}^{mem} P_{succ}^{calc} P_{succ}^{fs} [T_{mem} + R_{cm} + E_k]$$

Expected time of the segment

Case 3: The first error to be detected is a computation error

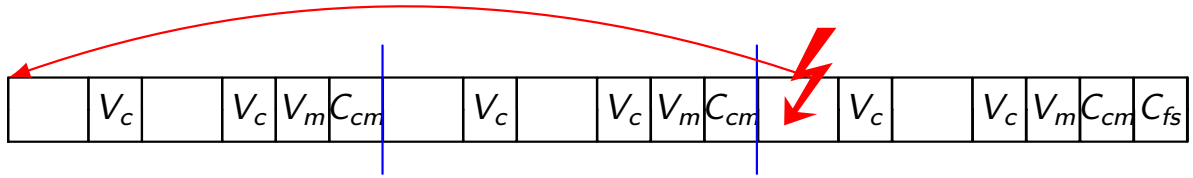


- No fail-stop error strikes before the computation error is detected
- Memory errors can strike (the memory verification is after the computation ones)

$$\Rightarrow \sum_{i=1}^{n_{cm}} P_{fail}^{calc}(i) P_{succ}^{fs}(i)[iT_{calc} + R_{cm} + E_k]$$

Expected time of the segment

Case 4: The first error to be detected is a fail-stop error



- No computation error strikes in the previous chunks
- Memory errors can strike

$\Rightarrow (1 - P_{no\_fs})[E_{lost} + R_{fs} + \sum_{i=0}^{k} E_i]$

The formula for the mean time

$$E = \sum_{k=0}^{n_{fs}-1} E_k + C_{fs}$$

$$E = \frac{M}{1 - P_{no\_fs}} \left[\left(\frac{1 - P_{no\_fs}}{P_{succ}^{fs,mem,calc}} + 1 \right)^{n_{fs}} - 1 \right] + C_{fs}$$

Minimization ...

$$
\begin{aligned}
M = {} & \mathrm{e}^{-(n_{cm}(T_{calc})+V_m)(\lambda_{fs}+\lambda_{mem})-C_{cm}\lambda_{fs}}(f_{calc})^{n_{vc}n_{cm}}[n_{cm}(T_{calc})+V_m+C_{cm}] \\
& + (1-\mathrm{e}^{-\lambda_{mem}(n_{cm}(T_{calc})+V_m)})\,\mathrm{e}^{-\lambda_{fs}(n_{cm}(T_{calc})+V_m)}(f_{calc})^{n_{vc}n_{cm}}[n_{cm}(T_{calc})+V_m+R_{cm}] \\
& + (1-f_{calc}^{n_{vc}})\,\mathrm{e}^{-\lambda_{fs}T_{calc}}\left(\frac{(f_{calc}^{n_{vc}}\,\mathrm{e}^{-\lambda_{fs}T_{calc}})^{n_{cm}}-1}{f_{calc}^{n_{vc}}\,\mathrm{e}^{-\lambda_{fs}T_{calc}}-1}\right)[R_{cm}] \\
& + (1-f_{calc}^{n_{vc}})\,\mathrm{e}^{-\lambda_{fs}T_{calc}}\left(\frac{(f_{calc}^{n_{vc}}\,\mathrm{e}^{-\lambda_{fs}T_{calc}})^{n_{cm}}(n_{cm}(f_{calc}^{n_{vc}}\,\mathrm{e}^{-\lambda_{fs}T_{calc}}-1)-1)+1}{(f_{calc}^{n_{vc}}\,\mathrm{e}^{-\lambda_{fs}T_{calc}}-1)^2}\right)[(n_{vc}I+V_c)] \\
& + [1-[\mathrm{e}^{-(n_{cm}(T_{calc})+V_m)(\lambda_{fs}+\lambda_{mem})-C_{cm}\lambda_{fs}}(f_{calc})^{n_{vc}n_{cm}} \\
& \quad + (1-\mathrm{e}^{-\lambda_{mem}(n_{cm}(T_{calc})+V_m)})\,\mathrm{e}^{-\lambda_{fs}(n_{cm}(T_{calc})+V_m)}(f_{calc})^{n_{vc}n_{cm}} \\
& \quad + (1-f_{calc}^{n_{vc}})\,\mathrm{e}^{-\lambda_{fs}T_{calc}}\left(\frac{(f_{calc}^{n_{vc}}\,\mathrm{e}^{-\lambda_{fs}T_{calc}})^{n_{cm}}-1}{f_{calc}^{n_{vc}}\,\mathrm{e}^{-\lambda_{fs}T_{calc}}-1}\right)]][\frac{1}{\lambda_{fs}}-\frac{n_{cm}(T_{calc})+V_m+C_{cm}}{\mathrm{e}^{\lambda_{fs}(n_{cm}(T_{calc})+V_m+C_{cm})}-1}+R_{fs}]
\end{aligned}
$$

$\rightarrow$ *Difficult to minimize analytically ...* ***(and this only for an exponential distribution)***

Plan

Preconditioned Conjugated Gradient (PCG)

For the simulation, we choose the Preconditioned Conjugated Gradient (PCG) for the iterative algorithm. PCG was the main motivation for this work.
PCG solves a system of linear equations when the matrix is positive semi-definite:

$$Ax = b$$

We use it because there are previous works to protect PCG from silent errors[1] (the contents of the V_c verification).

[1] E. Agullo, S. Cools, E. F. Yetkin, L. Giraud, N. Schenkels, and W. Vanroose, "On soft errors in the conjugate gradient method: Sensitivity and robust numerical detection," 2020

Simulation setting

- The simulation depends on the probabilities of errors: the variable x
- Error distributions: exponential distribution. Mean time between faults (MTBF $= \frac{1}{\lambda}$):
  - MTBF(fail-stop error) $= x$ (between 1h and 8h)
  - MTBF(memory error) $= \frac{x}{2}$
  - MTBF(computation error) $= \frac{x}{20}$

- For the checkpoints, the whole state is stored in the fail-stop checkpoint, whereas the memory checkpoint stores only the updated variables
- The memory verification is done through checksums

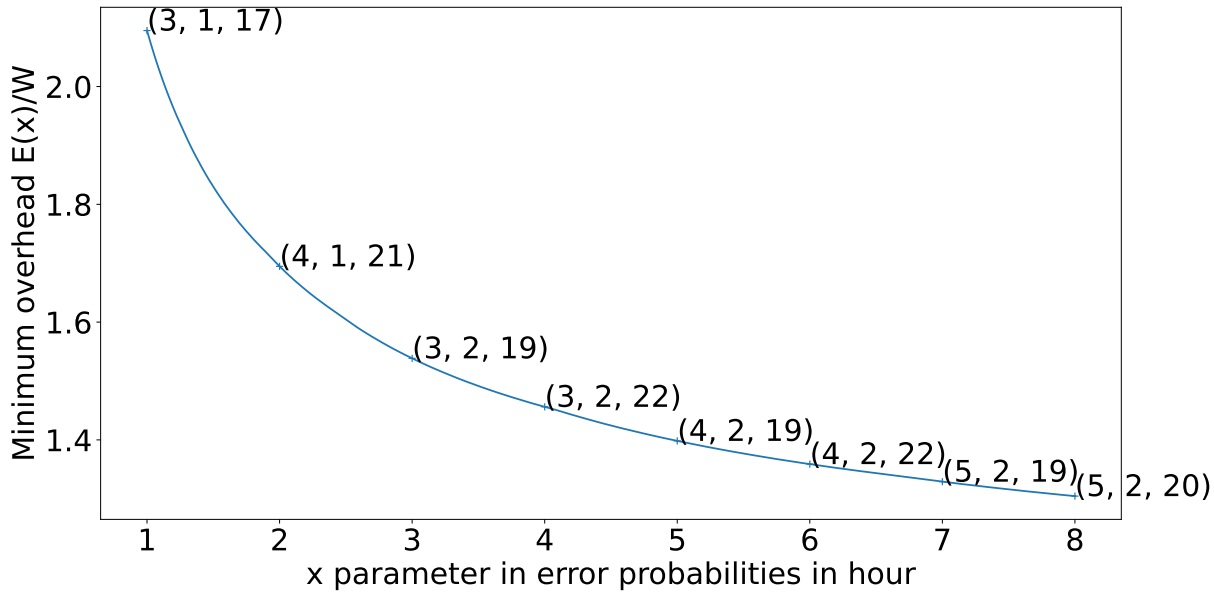
Scenarios: extrapolations from data

Scenario	Scenario 2^2	Scenario 3^3
Size of the problem	3.6×10^{12}	1.1×10^{13}
Number of nodes	3×10^3	2×10^4
I	13	110
V_c	2	17
V_m	6	3
C_{cm}	0.5	0.25
R_{cm}	0.5	0.25
C_{fs}	180	540
R_{fs}	180	540
	second	second

[2] N. Kohl and U. Rüde, "Textbook efficiency: Massively parallel matrix-free multigrid for the stokes system," 2022

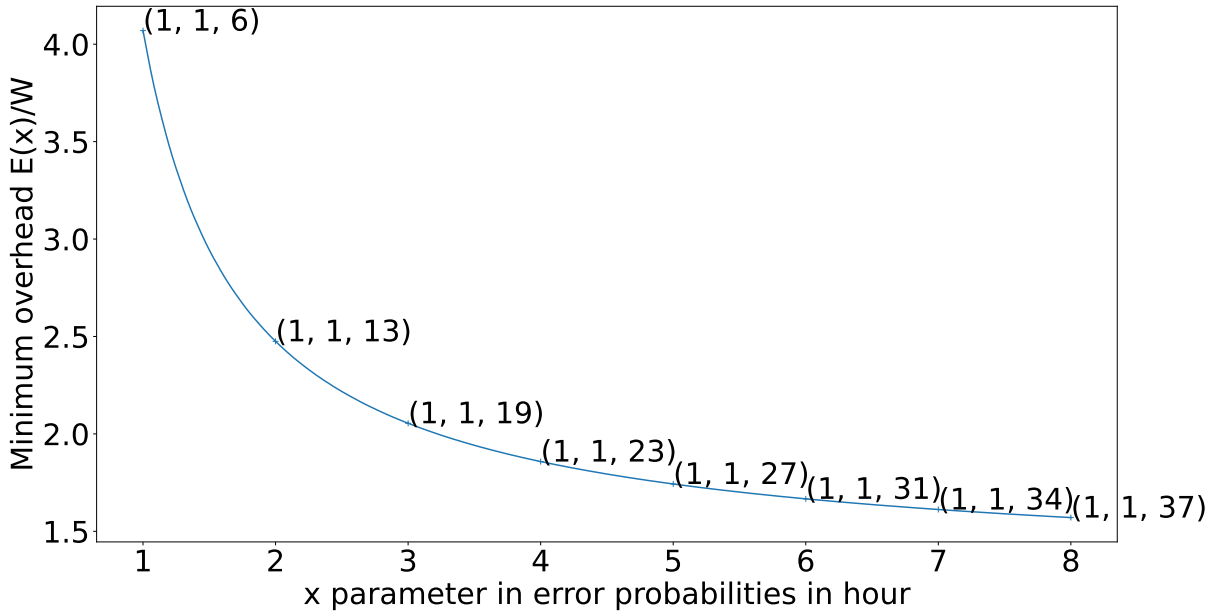
[3] M. Mohr, U. Rüde, B. Wohlmuth, and H.-P. Bunge, "Challenges for Mantle Convection Simulations at the ExaScale: Numerics, Algorithmics and Software", 2023

Simulation results: Scenario 2



For x=4 hours, (3,2,22) corresponds to n_{vc}=3 iterations in a chunk, n_{cm}=2 chunks in a segment and n_{fs}=22 segments in a pattern
Overhead $min\frac{E(4hours,3,2,22)}{I\times3\times2\times22}$: approximately 1.5 compared to the naive approach $E(4hours, 1, 1, 1)$ which is 15

Simulation results: Scenario 3



This scenario's result resembles the naive approach. This happens because the cost of a fail-stop checkpoint is massive and the cost of one iteration really exceeds the other costs.

Plan

Conclusion

Contributions:

- New model considering three types of failures in numerical iterative applications
- Computation of expected time
- Instantiation of the model on various realistic scenarios and simulations: Better than naive approach in terms of expected time

Future works: Some hypotheses are classical in the literature but not really realistic. Is it possible to remove these hypotheses?
For instance, what if the recall or precision of verification is not 1, what are the consequence on the model and expected time?