Resilience Algorithms to Cope with Fail-Stop and Silent Errors

Hongyang Sun

ENS Lyon

hongyang.sun@ens-lyon.fr

http://perso.ens-lyon.fr/hongyang.sun/

HPC Days in Lyon
7 April, 2016

Acknowledgment

Joint work with

- Anne Benoit, Aurélien Cavelan, Yves Robert (*ENS Lyon & Inria, France*)
- Leonardo Bautista-Gomez (*Argonne National Laboratory, USA*)
- Saurabh K. Raina (*Jaypee Institute of Information Technology, India*)

Computing at Exascale

Exascale platform

- Larger node count: 10^5 or 10^6 nodes, each with 10^2 or 10^3 cores
- Shorter Mean Time Between Failures (MTBF) μ

Theorem: $\mu_p = \frac{\mu_{\text{ind}}}{p}$ for arbitrary distributions

MTBF (individual node)	1 year	10 years	100 years
MTBF (platform of 10^6 nodes)	30 secs	5 mins	50 mins

- Multiple failure sources: fail-stop error, silent data corruption, etc.

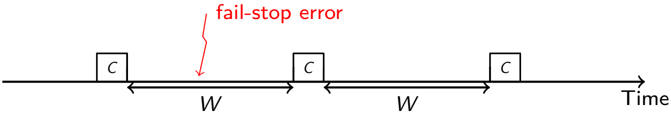
Need more reliable components!
Need more scalable algorithms!
Need more resilient techniques!

Coping with Fail-Stop Errors

Fail-stop errors: e.g., resource crash, node failure

- Instantaneous error detection

Standard approach: periodic checkpointing, rollback and recovery



Well-known first-order approximation formula to compute optimal checkpointing interval [*Young 1973, Daly 2006*]:

$$W^* = \sqrt{2\mu C}$$

μ: Platform MTBF

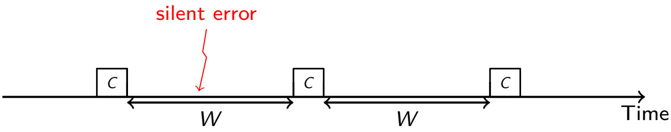
C: Checkpointing time

Coping with Silent Errors

Silent errors (or silent data corruptions): e.g., soft faults in L1 cache, ALU, double bit flip, due to cosmic radiation, packaging pollution, etc.

- Arbitrary detection latency

Same approach?

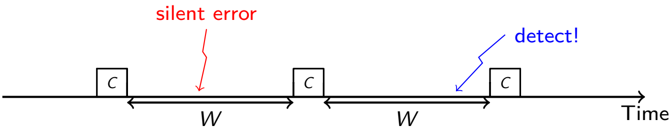


Coping with Silent Errors

Silent errors (or silent data corruptions): e.g., soft faults in L1 cache, ALU, double bit flip, due to cosmic radiation, packaging pollution, etc.

- Arbitrary detection latency

Same approach?

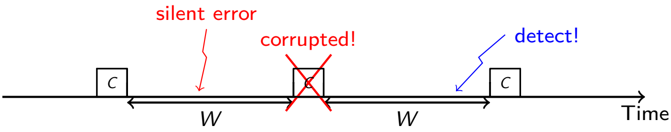


Coping with Silent Errors

Silent errors (or silent data corruptions): e.g., soft faults in L1 cache, ALU, double bit flip, due to cosmic radiation, packaging pollution, etc.

- Arbitrary detection latency

Same approach?

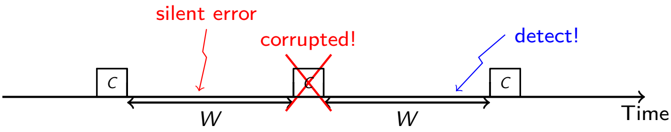


Coping with Silent Errors

Silent errors (or silent data corruptions): e.g., soft faults in L1 cache, ALU, double bit flip, due to cosmic radiation, packaging pollution, etc.

- Arbitrary detection latency

Same approach?



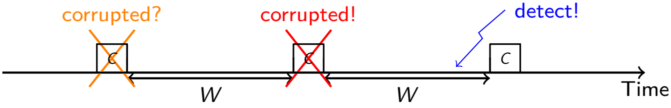
Keep multiple checkpoints?

Coping with Silent Errors

Silent errors (or silent data corruptions): e.g., soft faults in L1 cache, ALU, double bit flip, due to cosmic radiation, packaging pollution, etc.

- Arbitrary detection latency

Same approach?



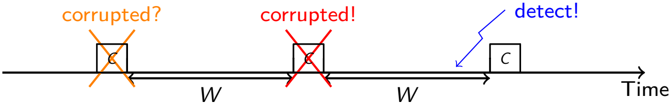
Keep multiple checkpoints?

Coping with Silent Errors

Silent errors (or silent data corruptions): e.g., soft faults in L1 cache, ALU, double bit flip, due to cosmic radiation, packaging pollution, etc.

- Arbitrary detection latency

Same approach?



Keep multiple checkpoints?

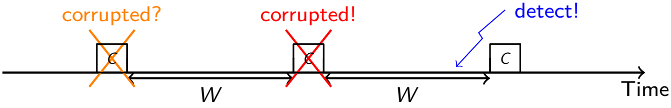
Which checkpoint to recover from?

Coping with Silent Errors

Silent errors (or silent data corruptions): e.g., soft faults in L1 cache, ALU, double bit flip, due to cosmic radiation, packaging pollution, etc.

- Arbitrary detection latency

Same approach?



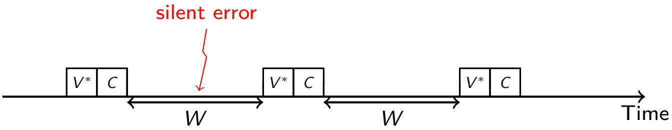
Keep multiple checkpoints?

Which checkpoint to recover from?

Need an active method to detect silent errors!

Coping with Silent Errors

Promising approach: coupling checkpointing with verification



- Before each checkpoint, run some verification mechanism or error detection test
- Silent error, if any, is detected by verification
- Need to maintain only one checkpoint, which is always valid ☺

Methods for Detecting Silent Errors

General-purpose approaches

- Replication [*Fiala et al. 2012*] or triple modular redundancy and voting [*Lyons and Vanderkulk 1962*]

Application-specific approaches

- Algorithm-based fault tolerance (ABFT): checksums in dense matrices Limited to one error detection and/or correction in practice [*Huang and Abraham 1984*]
- Partial differential equations (PDE): use lower-order scheme as verification mechanism [*Benson, Schmit and Schreiber 2014*]
- Generalized minimal residual method (GMRES): inner-outer iterations [*Hoemmen and Heroux 2011*]
- Preconditioned conjugate gradients (PCG): orthogonalization check every k iterations, re-orthogonalization if problem detected [*Sao and Vuduc 2013, Chen 2013*]

Data-analytics approaches

- Dynamic monitoring of HPC datasets based on physical laws (e.g., temperature limit, speed limit) and space or temporal proximity [*Bautista-Gomez and Cappello 2014*]
- Time-series prediction, spatial multivariate interpolation [*Di et al. 2014*]

Methods for Detecting Silent Errors

General-purpose approaches

- R… …ting […

Applica…

Our focus is *not* about the design of silent error detectors

- A… …ces Limited to one error detection and/or correction in practice [*Huang and A*…

Instead, it is about how to make the best use of detectors (verifications) to design efficient resilience algorithms

- P… v…
- G… …ns [*…*
- Preconditioned conjugate gradients (PCG): orthogonalization check every k iterations, re-orthogonalization if problem detected [*Sao and Vuduc 2013, Chen 2013*]

Data-analytics approaches

- Dynamic monitoring of HPC datasets based on physical laws (e.g., temperature limit, speed limit) and space or temporal proximity [*Bautista-Gomez and Cappello 2014*]
- Time-series prediction, spatial multivariate interpolation [*Di et al. 2014*]

Methods for Detecting Silent Errors

General-purpose approaches

- R… …ting [… …

Applica…

- A… …es Limited to one error detection and/or correction in practice [*Huang and A*…
- P… v…
- G… …ns [… …
- Preconditioned conjugate gradients (PCG): orthogonalization check every *k* … …*uc* 2…

Data-a…

- D… te… [… …
- T… …*014*]

Our focus is ***not*** about the design of silent error detectors

Instead, it is about how to make the best use of detectors (verifications) to design efficient resilience algorithms

What is optimal checkpointing interval?
Does intermediate verification help?
What is the optimal verification position?
How to cope with inaccurate detectors?

Outline

1. Coping with Silent Errors
   - Models
   - Analysis of several patterns
2. Coping with Fail-stop and Silent Errors
3. Conclusion and Future Work

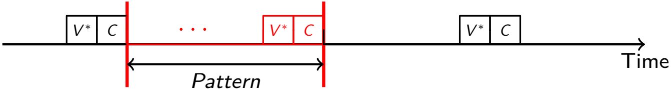
Outline

Models

Failure arrivals follow exponential law $Exp(\lambda)$, where $\lambda = 1/\mu$.

- $P(\lambda, w) = 1 - e^{\lambda w}$ (memoryless)

Design a periodic computing pattern that minimizes the expected execution time (or makespan) of the application.



A pattern has the following characteristics:

- End with a verified checkpoint (avoid saving corrupted checkpoints)
- May contain intermediate verifications (for better performance)

Models

Execution overhead

Suppose an application is divided into periodic patterns of work W. If the expected execution time of a pattern is $\mathbb{E}(W)$, then

$$\begin{aligned} \mathit{Makespan} &\approx \frac{\mathit{Total\_work}}{W} \cdot \mathbb{E}(W) \\ &= (1 + \mathbb{H}) \cdot \mathit{Total\_work} \end{aligned}$$

where

$$\mathbb{H} = \frac{\mathbb{E}(W)}{W} - 1$$

denote the execution overhead of the pattern.

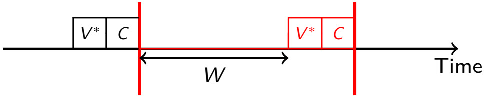
E.x. if $W = 100$, $\mathbb{E}(W) = 125$, then $\mathbb{H} = 25\%$.

Proposition

For large applications, minimizing expected makespan is equivalent to minimizing the execution overhead of a pattern.

Outline

Base Pattern P_c (Revisiting Young/Daly)



Proposition

The optimal checkpointing interval W^* and optimal execution overhead $\mathbb{H}^*$ of the base pattern P_c are

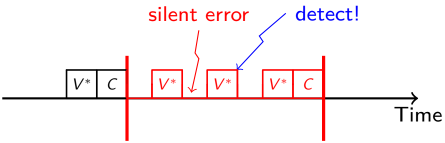
$$W^* = \sqrt{\frac{V^* + C}{\lambda}}$$

$$\mathbb{H}^* = 2\sqrt{\lambda(V^* + C)} + O(\lambda)$$

	Fail-stop errors	Silent errors
Pattern	$W + C$	$W + V^* + C$
Optimal W^*	$\sqrt{\frac{2C}{\lambda}}$	$\sqrt{\frac{V^*+C}{\lambda}}$
Optimal $\mathbb{H}^*$	$\sqrt{2\lambda C}$	$2\sqrt{\lambda(V^* + C)}$

Pattern P_{V^*C} with Intermediate Verifications

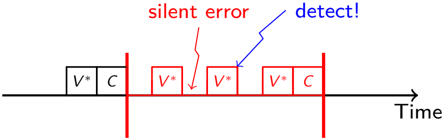
Can we do better by adding intermediate verifications in a pattern?



- Silent errors detected earlier in the pattern ☺
- Additional overhead in fault-free execution ☹

Pattern P_{V^*C} with Intermediate Verifications

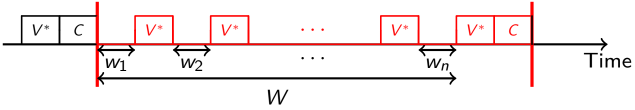
Can we do better by adding intermediate verifications in a pattern?



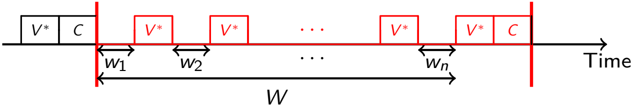
- Silent errors detected earlier in the pattern ☺
- Additional overhead in fault-free execution ☹

When is it better to use intermediate verifications?
What is the optimal checkpointing period?
How many verifications to use?
What are their positions?

Pattern P_{V^*C} with Intermediate Verifications



Pattern P_{V^*C} with Intermediate Verifications



Proposition

The optimal P_{V^*C} pattern has checkpointing interval W^* and contains n^* equi-spaced verifications:

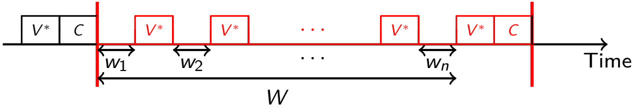
$$n^* = \sqrt{\frac{C}{V^*}} \quad \Leftarrow \text{necessary condition: } C > V^*$$

$$W^* = \sqrt{\frac{n^* V^* + C}{\frac{1}{2}\left(1 + \frac{1}{n^*}\right)\lambda}} = \sqrt{\frac{2C}{\lambda}} > \sqrt{\frac{V^* + C}{\lambda}} \quad \Leftarrow \text{base pattern}$$

$$\mathbb{H}^* = \sqrt{2\lambda V^*} + \sqrt{2\lambda C} + O(\lambda)$$

$$< 2\sqrt{\lambda(V^* + C)} + O(\lambda) \quad \Leftarrow \text{base pattern}$$

Pattern P_{V^*C} with Intermediate Verifications



Proposition

The optimal P_{V^*C} pattern has checkpointing interval W^* and contains n^* equi-spaced verifications:

$$n^* = \sqrt{\frac{C}{V^*}} \quad \Leftarrow \text{necessary condition: } C > V^*$$

$$W^* = \sqrt{\frac{n^* V^* + C}{\frac{1}{2}\left(1 + \frac{1}{n^*}\right)\lambda}} = \sqrt{\frac{2C}{\lambda}} > \sqrt{\frac{V^* + C}{\lambda}} \quad \Leftarrow \text{base pattern}$$

$$\mathbb{H}^* = \sqrt{2\lambda V^*} + \sqrt{2\lambda C} + O(\lambda)$$

$$< 2\sqrt{\lambda(V^* + C)} + O(\lambda) \quad \Leftarrow \text{base pattern}$$

Practical no. of verifications must be an integer: $\max(1, \lfloor n^* \rfloor)$ or $\lceil n^* \rceil$

Observations

Observation 1

The expected time to execute a pattern of length W is

$$\mathbb{E}(W) = \underbrace{W + o_{\text{ff}}}_{\text{error-free time}} + \underbrace{\lambda W}_{\text{expected \#errors}} \cdot \underbrace{\left(f_{\text{re}} \cdot W + O(V^*) + R\right)}_{\text{expected re-execution time}} + O(\lambda)$$

- o_{ff}: overhead in a fault-free execution, i.e., $\sum$resilience ops.
- f_{re}: fraction of re-executed work in case of faults.

Observations

Observation 1

The expected time to execute a pattern of length W is

$$\mathbb{E}(W) = \underbrace{W + o_{\text{ff}}}_{\text{error-free time}} + \underbrace{\lambda W}_{\text{expected \#errors}} \cdot \underbrace{\left(f_{\text{re}} \cdot W + O(V^*) + R\right)}_{\text{expected re-execution time}} + O(\lambda)$$

- o_{ff}: overhead in a fault-free execution, i.e., $\sum$resilience ops.
- f_{re}: fraction of re-executed work in case of faults.

Observation 2

The optimal pattern satisfies

$$W^* = \sqrt{\frac{o_{\text{ff}}}{\lambda f_{\text{re}}}}$$

$$\mathbb{H}^* = 2\sqrt{\lambda \cdot f_{\text{re}} o_{\text{ff}}} + O(\lambda)$$

Observations

Observation 1

The expected time to execute a pattern of length W is

$$\mathbb{E}(W) = \underbrace{W + o_{ff}}_{\text{error-free time}} + \underbrace{\lambda W}_{\text{expected \#errors}} \cdot \underbrace{\left(f_{re} \cdot W + O(V^*) + R\right)}_{\text{expected re-execution time}} + O(\lambda)$$

- o_{ff}: overhead in a fault-free execution, i.e., $\sum$resilience ops.
- f_{re}: fraction of re-executed work in case of faults.

Observation 2

The optimal pattern satisfies

$$W^* = \sqrt{\frac{o_{ff}}{\lambda f_{re}}}$$
$$\mathbb{H}^* = 2\sqrt{\lambda \cdot f_{re} o_{ff}} + O(\lambda)$$

Asymptotically, minimizing $\mathbb{H}$ is equivalent to minimizing $f_{re} o_{ff}$

Some Observations

Example 1: Base pattern P_c

$$\mathbb{E}(W) = W + \underbrace{V^* + C}_{o_{\mathrm{ff}}} + \lambda W(\underbrace{1}_{f_{\mathrm{re}}} \cdot W + V^* + R) + O(\lambda)$$

$$W^* = \sqrt{\frac{V^* + C}{\lambda}} \text{ and } \mathbb{H}^* \approx 2\sqrt{\lambda(V^* + C)}$$

Example 2: Pattern P_{v^*c}

$$\mathbb{E}(W) = W + \underbrace{nV^* + C}_{o_{\mathrm{ff}}} + \lambda W\left(\underbrace{\frac{1}{2}(1 + \frac{1}{n})}_{f_{\mathrm{re}}} \cdot W + \frac{n+1}{2}V^* + R\right) + O(\lambda)$$

$$W^* = \sqrt{\frac{nV^* + C}{\frac{1}{2}\left(1 + \frac{1}{n}\right)\lambda}} \text{ and } \mathbb{H}^* \approx 2\sqrt{\lambda\frac{1}{2}(nV^* + C)\left(1 + \frac{1}{n}\right)}$$

Pattern P_{vc} with Partial Verifications

Guaranteed/perfect verifications can be very expensive

For HPC applications, many silent error detectors are partial

- Lower cost ☺
- Lower accuracy ☹

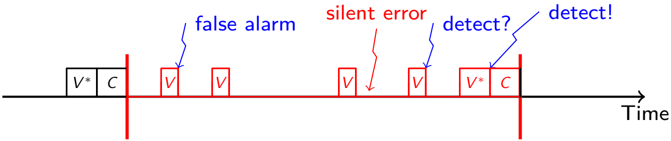
$$\text{cost } V \quad \ll \quad \text{cost } V^* \text{ of guaranteed verification}$$

$$\text{recall } r \quad = \quad \frac{\#\text{detected\_errors}}{\#\text{total\_errors}} < 1 \text{ (false negative)}$$

$$\text{precision } p \quad = \quad \frac{\#\text{true\_errors}}{\#\text{detected\_errors}} < 1 \text{ (false positive)}$$

Pattern P_{vc} with Partial Verifications

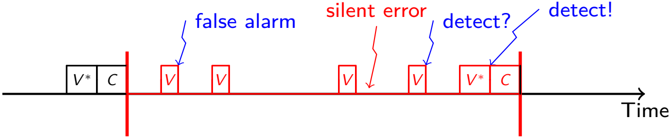
Can we do better by using partial verifications in a pattern?



- A partial verification may raise false alarms (with prob. $1 - p$)
- A partial verification may miss errors (with prob. $1 - r$)
- Last verification guaranteed to avoid saving invalid checkpoints

Pattern P_{VC} with Partial Verifications

Can we do better by using partial verifications in a pattern?



- A partial verification may raise false alarms (with prob. $1 - p$)
- A partial verification may miss errors (with prob. $1 - r$)
- Last verification guaranteed to avoid saving invalid checkpoints

When is it better to use partial verifications?
What is the optimal checkpointing period?
How many partial verifications to use?
What are their positions?

Pattern P_{vc} with Partial Verifications

Proposition

The optimal pattern P_{vc} does not use any partial verification with constant precision $p < 1$

In particular, the result holds if the precision satisfies $p = 1 - \Omega(\lambda^{1/2})$

- Intuitively, an imprecise verification becomes another error source with error probability $1 - p$
- With first-order approximation, probability of a silent error in the pattern is $1 - e^{\lambda W} \approx \lambda W = \Theta(\lambda^{1/2})$

Pattern P_{vc} with Partial Verifications

Proposition

The optimal pattern P_{vc} does not use any partial verification with constant precision $p < 1$

In particular, the result holds if the precision satisfies $p = 1 - \Omega(\lambda^{1/2})$

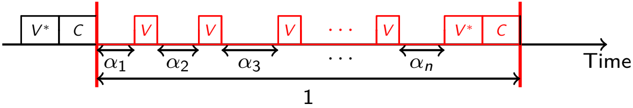
- Intuitively, an imprecise verification becomes another error source with error probability $1 - p$
- With first-order approximation, probability of a silent error in the pattern is $1 - e^{\lambda W} \approx \lambda W = \Theta(\lambda^{1/2})$

Having a recall $r < 1$ is fine, because errors are rare and will eventually be detected by the final guaranteed verification

Tradeoff between recall and precision $\Rightarrow$ maximize precision
(e.g. $p > 0.999$ for $\lambda = 10^{-6}$)

We will assume $p = 1$ for subsequent analysis

Pattern P_{vc} with Partial Verifications



(1) Apply the $f_{re}o_{ff}$ analysis

Proposition

Suppose a pattern P_{vc} has n segments ($n-1$ partial verifications and one guaranteed verification), and the i-th segment has α_i fraction of work. Then the pattern is characterized by

$$\begin{aligned} o_{ff} &= (n-1)V + V^* + C \\ f_{re} &= \boldsymbol{\alpha}^T A \boldsymbol{\alpha} \end{aligned}$$

where $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_n]^T$ and A is a symmetric positive definite matrix defined by $A_{i,j} = \frac{1}{2}\left(1 + (1-r)^{|i-j|}\right)$ for $1 \le i, j \le n$

Pattern P_{vc} with Partial Verifications

(2) Determine α to minimize $f_{\mathbf{re}}$ (involved analysis)

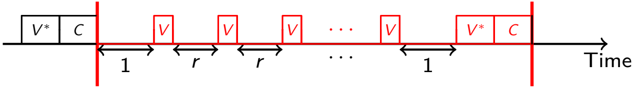
Proposition

The re-execution fraction f_{re} of a pattern P_{vc} with n segments is minimized when $\boldsymbol{\alpha} = \boldsymbol{\alpha}^*$, where

$$\alpha_i^* = \begin{cases} \frac{1}{(n-2)r+2} & \text{for } i = 1, n \\ \frac{r}{(n-2)r+2} & \text{for } i = 2, 3, \ldots, n-1 \end{cases}$$

and the optimal value of f_{re} is

$$f_{re}^* = \frac{1}{2}\left(1 + \frac{2-r}{(n-2)r+2}\right)$$



If all verifications are perfect ($r = 1$), we retrieve equal-length segments, i.e., $\alpha_i^* = \frac{1}{n}$ for all $1 \le i \le n$ and $f_{\mathrm{re}}^* = \frac{1}{2}\left(1 + \frac{1}{n}\right)$

Pattern P_{vc} with Partial Verifications

(3) Minimize $f_{\text{re}}o_{\text{ff}} = \frac{1}{2}\left(1 + \frac{2-r}{(n-2)r+2}\right)\left((n-1)V + V^* + C\right)$

- accuracy $a = \frac{r}{2-r}$ and relative cost $b = \frac{V}{V^*+C}$
- accuracy-to-cost ratio $\phi = \frac{a}{b}$

Proposition

The optimal P_{vc} *pattern satisfies*

$$n^* = 1 - \frac{1}{a} + \sqrt{\frac{1}{a}\left(\frac{1}{b} - \frac{1}{a}\right)} \quad \Leftarrow \text{ necessary condition: } \phi > 2$$

$$W^* = \sqrt{\frac{2(V^*+C)}{\lambda}\left(1 - \frac{1}{\phi}\right)} > \sqrt{\frac{2C}{\lambda}} \quad \Leftarrow \text{ Pattern } \mathrm{P}_{v^*c}$$

$$\mathbb{H}^* = \sqrt{2\lambda(V^*+C)}\left(\sqrt{1-\frac{1}{\phi}} + \sqrt{\frac{1}{\phi}}\right) + O(\lambda)$$

$$< \sqrt{2\lambda V^*} + \sqrt{2\lambda C} + O(\lambda) \quad \Leftarrow \text{ Pattern } \mathrm{P}_{v^*c}$$

Pattern P_{vc} with Partial Verifications

Assessing the benefit of partial verifications on realistic platform

- 10^5 computing nodes with individual MTBF of 100 years
  $\Rightarrow$ platform MTBF $\mu = 31536s \approx 8.7$ hours
- Checkpoint size of 300GB with throughput of 0.5GB/s
  $\Rightarrow C = 600s = 10$ mins, and V^* in same order
- Partial verifications using lightweight detectors
  $\Rightarrow V$ typically tens of seconds, and $r \in [0.5, 0.95]$

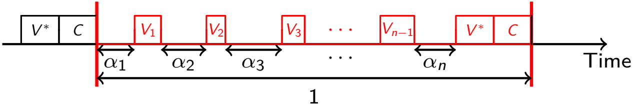
e.g., $C = 600$, $V^* = 300$, $V = 30$ and $p = 1$, $r = 0.8$

	Pattern P_{vc}	Pattern P_{v^*c}	Pattern P_{c}
W^*	$7335s \approx 2.04$ hours	$7103s \approx 1.97$ hours	$5328s \approx 1.48$ hours
n^*	6	2	1
$\boldsymbol{\alpha}^*$	$\alpha_i = \begin{cases} 0.20, i = 1, 6 \\ 0.15, i = 2..5 \end{cases}$	$[0.5, 0.5]$	$[1]$
$\mathbb{H}^*$	28.6%	33.3%	33.8%

Pattern with Multiple Partial Detectors

Can we do better by using multiple types of partial verifications?

$D^{(1)} = \left(V^{(1)}, r^{(1)}\right)$, $D^{(2)} = \left(V^{(2)}, r^{(2)}\right)$, $\ldots$, $D^{(k)} = \left(V^{(k)}, r^{(k)}\right)$



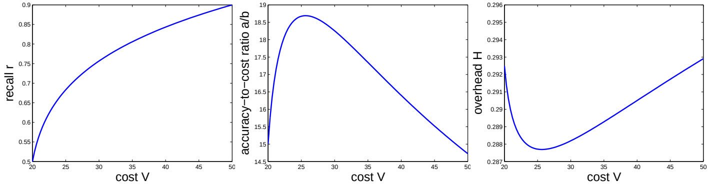
The i-th partial verification has type j, i.e., $V_i = V^{(j)}$ for some $1 \leq j \leq k$

Which verification is the optimal one to use?
What is the optimal combination of partial verifications?

Pattern with Multiple Partial Detectors

Proposition

The optimal pattern P_{vc} uses the partial verification with the highest accuracy-to-cost ratio

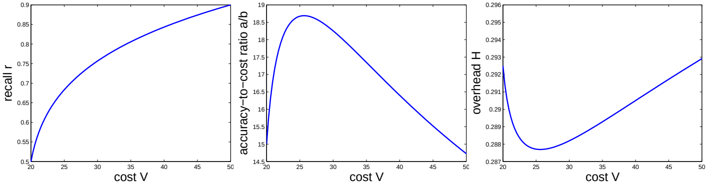


- Result is based on optimal rational solution (n^*)
- Overhead of rounded integer solution may no longer be optimal

Pattern with Multiple Partial Detectors

Proposition

The optimal pattern P_{vc} uses the partial verification with the highest accuracy-to-cost ratio



- Result is based on optimal rational solution (n^*)
- Overhead of rounded integer solution may no longer be optimal

What is the optimal integer solution?

Pattern with Multiple Partial Detectors

Proposition

Finding the optimal P_{vc} pattern with k verification types is NP-complete, even when all verification types share the same accuracy-to-cost ratio, i.e., $\frac{a^{(j)}}{b^{(j)}} = \phi$ for all $1 \leq j \leq k$

Pattern with Multiple Partial Detectors

Proposition

Finding the optimal P_{vc} pattern with k verification types is NP-complete, even when all verification types share the same accuracy-to-cost ratio, i.e., $\frac{a^{(j)}}{b^{(j)}} = \phi$ for all $1 \leq j \leq k$

Approximation algorithms:

- FPTAS (Fully Polynomial-Time Approximation Scheme)
  - Overhead within $1 + \epsilon$ times the optimal with running time polynomial in the input size and $1/\epsilon$ for any $\epsilon > 0$.
  - The solution is independent of the ordering of the verifications
- Greedy algorithm
  - Compute the optimal solution using the one detector with the highest accuracy-to-cost ratio, and then round up the solution
  - This algorithm has approximation ratio $\sqrt{3/2} < 1.23$

Pattern with Multiple Partial Detectors

Performance evaluation on realistic platform

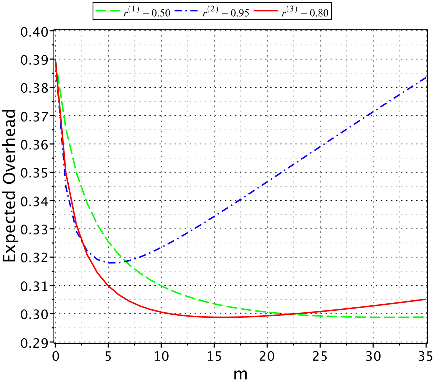
- 10^5 computing nodes with individual MTBF of 100 years
  $\Rightarrow$ platform MTBF $\mu \approx 8.7$ hours
- Checkpoints size of 300GB with throughput of 0.5GB/s
  $\Rightarrow$ $C = 600s = 10$ mins, and V^* in same order
- Several realistic partial detectors based on data-analytics approach

	cost	recall	ACR
Time series prediction	$V^{(1)} = 3s$	$r^{(1)} = [0.5, 0.9]$	$\phi^{(1)} = [133, 327]$
Spatial interpolation	$V^{(2)} = 30s$	$r^{(2)} = [0.75, 0.95]$	$\phi^{(2)} = [24, 36]$
Combination of two	$V^{(3)} = 6s$	$r^{(3)} = [0.8, 0.99]$	$\phi^{(3)} = [133, 196]$
Perfect verification	$V^* = 600s$	$r^* = 1$	$\phi^* = 2$

A detector's recall may vary depending on the application or dataset

Pattern with Multiple Partial Detectors

Using one type of verification ($r^{(1)} = 0.5$, $r^{(2)} = 0.95$, $r^{(3)} = 0.8$)



Best partial detectors offer ~9% improvement in overhead
Saving ~55 minutes for every 10 hours of computation!

Pattern with Multiple Partial Detectors

Using multiple types of verifications

	m	overhead H	diff. from opt.
Scenario 1: $r^{(1)} = 0.51$, $r^{(3)} = 0.82$, $\phi^{(1)} \approx 137$, $\phi^{(3)} \approx 139$			
Optimal solution	(1, 15)	29.828%	0%
Greedy with $V^{(3)}$	(0, 16)	29.829%	0.001%
Scenario 2: $r^{(1)} = 0.58$, $r^{(3)} = 0.9$, $\phi^{(1)} \approx 163$, $\phi^{(3)} \approx 164$			
Optimal solution	(1, 14)	29.659%	0%
Greedy with $V^{(3)}$	(0, 15)	29.661%	0.002%
Scenario 3: $r^{(1)} = 0.64$, $r^{(3)} = 0.97$, $\phi^{(1)} \approx 188$, $\phi^{(3)} \approx 188$			
Optimal solution	(1, 13)	29.523%	0%
Greedy with $V^{(1)}$	(27, 0)	29.524%	0.001%
Greedy with $V^{(3)}$	(0, 14)	29.525%	0.002%

The Greedy algorithm works very well in this practical setting!

Outline

1. Coping with Silent Errors
   - Models
   - Analysis of several patterns
2. Coping with Fail-stop and Silent Errors
3. Conclusion and Future Work

Coping with Fail-stop and Silent Errors

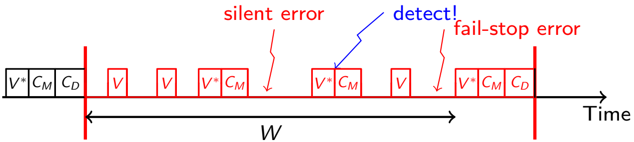
Fail-stop errors and silent errors coexist in large-scale platforms
A resilience pattern needs to cope with both error sources simultaneously

Coping with Fail-stop and Silent Errors

Fail-stop errors and silent errors coexist in large-scale platforms
A resilience pattern needs to cope with both error sources simultaneously

Two-level checkpointing and verification framework

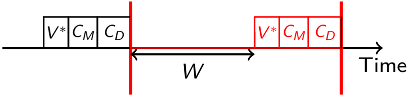
- Fail-stop errors (λ_f) are handled by disk checkpoints (C_D)
- Silent errors (λ_s) are handled by in-memory checkpoints (C_M) and verifications (guaranteed V^* or partial V)



Framework enforcing two properties:

- *A guaranteed verification before each memory checkpoint*
  ⇒ *Checkpoints always valid*
- *A memory checkpoint before each disk checkpoint*
  ⇒ *Always recover from latest checkpoints*

Two-level Base Pattern P_D (Revisiting Young/Daly Again)



Proposition

The optimal checkpointing interval W^* and optimal execution overhead H^* of the two-level base pattern P_D are

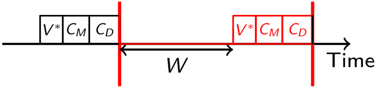
$$W^* = \sqrt{\frac{V^* + C_M + C_D}{\lambda_s + \frac{\lambda_f}{2}}}$$

$$\mathbb{H}^* = 2\sqrt{\left(\lambda_s + \frac{\lambda_f}{2}\right)(V^* + C_M + C_D)} + O(\lambda)$$

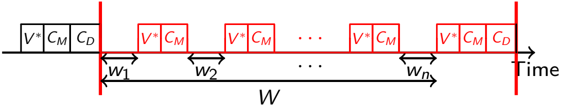
	Fail-stop errors	Silent errors	Both errors
Pattern	$W + C_D$	$W + V^* + C_M$	$W + V^* + C_M + C_D$
Optimal W^*	$\sqrt{\frac{2C_D}{\lambda_f}}$	$\sqrt{\frac{V^*+C_M}{\lambda_s}}$	$\sqrt{\frac{V^*+C_M+C_D}{\lambda_s+\frac{\lambda_f}{2}}}$
Optimal $\mathbb{H}^*$	$\sqrt{2\lambda_f C_D}$	$2\sqrt{\lambda_s(V^* + C_M)}$	$2\sqrt{\left(\lambda_s + \frac{\lambda_f}{2}\right)(V^* + C_M + C_D)}$

Various Two-level Patterns

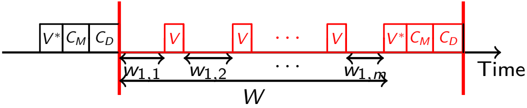
- Pattern P_D



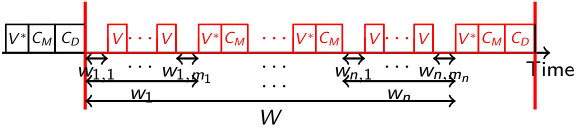
- Pattern P_{DM}



- Pattern P_{DV^*} or P_{DV}



- Pattern P_{DMV^*} or P_{DMV}



Summary of Results

Parameters of various optimal patterns

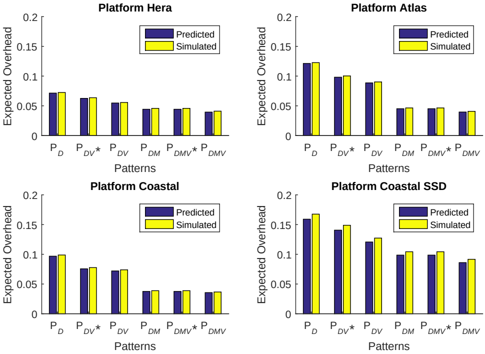
- W^*: optimal pattern length
- n^*: optimal #memory checkpoints between two disk checkpoints
- m^*: optimal #verifications between two memory checkpoints

Pattern	W^*	n^*	m^*	$\mathbb{H}^*$
P_D	$\sqrt{\frac{V^*+C_M+C_D}{\lambda_s+\frac{\lambda_f}{2}}}$	–	–	$2\sqrt{\left(\lambda_s+\frac{\lambda_f}{2}\right)(V^*+C_M+C_D)}$
P_{DV^*}	$\sqrt{\frac{m^*V^*+C_M+C_D}{\frac{1}{2}\left(1+\frac{1}{m^*}\right)\lambda_s+\frac{\lambda_f}{2}}}$	–	$\sqrt{\frac{\lambda_s}{\lambda_s+\lambda_f}\cdot\frac{C_M+C_D}{V^*}}$	$\sqrt{2(\lambda_s+\lambda_f)C_M+C_D}+\sqrt{2\lambda_s V^*}$
P_{DV}	$\sqrt{\frac{(m^*-1)V+V^*+C_M+C_D}{\frac{1}{2}\left(1+\frac{2-r}{(m^*-2)r+2}\right)\lambda_s+\frac{\lambda_f}{2}}}$	–	$2-\frac{2}{r}+\sqrt{\frac{\lambda_s}{\lambda_s+\lambda_f}}$ $\times\sqrt{\frac{2-r}{r}\left(\frac{V^*+C_M+C_D}{V}-\frac{2-r}{r}\right)}$	$\sqrt{2(\lambda_s+\lambda_f)\left(V^*-\frac{2-r}{r}V+C_M+C_D\right)}$ $+\sqrt{2\lambda_s\frac{2-r}{r}V}$
P_{DM}	$\sqrt{\frac{n^*(V^*+C_M)+C_D}{\frac{\lambda_s}{n^*}+\frac{\lambda_f}{2}}}$	$\sqrt{\frac{2\lambda_s}{\lambda_f}\cdot\frac{C_D}{V^*+C_M}}$	–	$2\sqrt{\lambda_s(V^*+C_M)}+\sqrt{2\lambda_f C_D}$
P_{DMV^*}	$\sqrt{\frac{n^*m^*V^*+n^*C_M+C_D}{\frac{1}{2}\left(1+\frac{1}{m^*}\right)\frac{\lambda_s}{n^*}+\frac{\lambda_f}{2}}}$	$\sqrt{\frac{\lambda_s}{\lambda_f}\cdot\frac{C_D}{C_M}}$	$\sqrt{\frac{C_M}{V^*}}$	$\sqrt{2\lambda_f C_D}+\sqrt{2\lambda_s C_M}+\sqrt{2\lambda_s V^*}$
P_{DMV}	$\sqrt{\frac{n^*(m^*-1)V+n^*(V^*+C_M)+C_D}{\frac{1}{2}\left(1+\frac{2-r}{(m^*-2)r+2}\right)\frac{\lambda_s}{n^*}+\frac{\lambda_f}{2}}}$	$\sqrt{\frac{\lambda_s}{\lambda_f}\cdot\frac{C_D}{V^*-\frac{2-r}{r}V+C_M}}$	$2-\frac{2}{r}$ $+\sqrt{\frac{2-r}{r}\left(\frac{V^*+C_M}{V}-\frac{2-r}{r}\right)}$	$\sqrt{2\lambda_f C_D}+\sqrt{2\lambda_s\left(V^*-\frac{2-r}{r}V+C_M\right)}$ $+\sqrt{2\lambda_s\frac{2-r}{r}V}$

Performance Evaluation

- Parameters of four real platforms [*Moody et al. 2010*]
- $V^* = C_M$, $V = C_M/100$ and $r = 0.8$

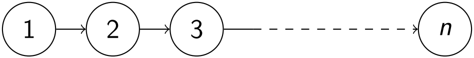
platform	#nodes	λ_f	λ_s	C_D	C_M
Hera	256	9.46e-7	3.38e-6	300s	15.4s
Atlas	512	5.19e-7	7.78e-6	439s	9.1s
Coastal	1024	4.02e-7	2.01e-6	1051s	4.5s
Coastal SSD	1024	4.02e-7	2.01e-6	2500s	180.0s



Dealing with Linear Task Graph

A linear chain of n task, and each task T_i is characterized by a work w_i

Resilience operations (e.g., checkpoint, verification) possible only at the end of a task



Which tasks to checkpoint (memory or disk) and which tasks to verify (guaranteed or partial) to minimize the expected makespan?

Optimal algorithm based on dynamic programming:

- Complexity $O(n^4)$ using only guaranteed verification
- Complexity $O(n^6)$ using also partial verification

Outline

Conclusion

Summary

- First comprehensive analysis of computing patterns to cope with silent errors
- Two-level checkpointing and verification framework to cope with fail-stop and silent errors
- Optimal dynamic programming algorithms for linear task graph
- Performance evaluation based on parameters from real platforms

Future Work

- Analysis of multi-level/hierarchical checkpointing patterns
- Coping with failures in computational workflows modeled as directed acyclic graphs (DAGs)

Thanks for Your Attention!

References

[1] Assessing general-purpose algorithms to cope with fail-stop and silent errors. *In Proceedings of PMBS'14, extended version to appear in ACM TOPC, available as INRIA research report RR-8599*.

[2] Assessing the impact of partial verifications against silent data corruptions. *In Proceedings of ICPP'15, available as INRIA research report RR-8711*.

[3] Which verification for soft error detection? *In Proceedings of HiPC'15, available as INRIA research report RR-8741*.

[4] Coping with recall and precision of soft error detectors. *INRIA research report RR-8832*.

[5] Optimal resilience patterns to cope with fail-stop and silent errors. *To appear in IPDPS'16, available as INRIA research report RR-8786*.

[6] Two-level checkpointing and verifications for linear task graphs. *To appear in PDSEC'16, available as INRIA research report RR-8794*.