Assessing the Impact of Partial Verifications Against Silent Data Corruptions

Aurélien Cavelan[1], Saurabh K. Raina[2], Yves Robert[1,3] and Hongyang Sun[1]

1. Ecole Normale Superieure de Lyon & INRIA, France
2. Jaypee Institute of Information Technology, India
3. University of Tennessee Knoxville, USA

hongyang.sun@ens-lyon.fr

ICPP – September 3, 2015

HPC at Scale

Scale is a major opportunity:

- Exascale platform: 10^5 or 10^6 nodes, each with 10^2 or 10^3 cores.

Scale is also a major threat:

- 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	120 years
MTBF (platform of 10^6 nodes)	30 sec	5 mn	1 h

HPC at Scale

Scale is a major opportunity:

- Exascale platform: 10^5 or 10^6 nodes, each with 10^2 or 10^3 cores.

Scale is also a major threat:

- Shorter Mean Time Between Failures (MTBF) μ.

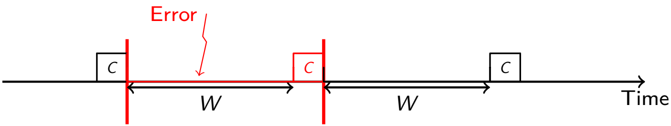
Theorem: $\mu_p = \frac{\mu_{\text{ind}}}{p}$ for arbitrary distributions

MTBF (individual node)	1 year	10 years	120 years
MTBF (platform of 10^6 nodes)	30 sec	5 mn	1 h

Need more reliable components!!
Need more resilient techniques!!!

General-purpose approach

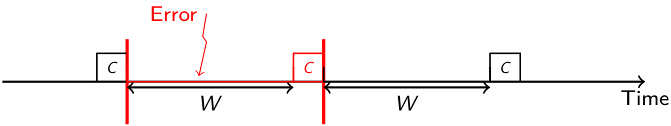
Periodic checkpoint, rollback and recovery:



- Fail-stop errors: e.g., hardware crash, node failure
  - Instantaneous error detection.

General-purpose approach

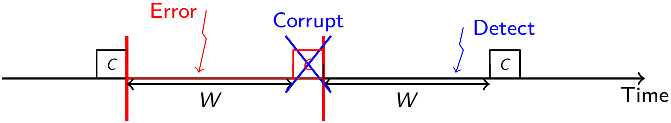
Periodic checkpoint, rollback and recovery:



- Fail-stop errors: e.g., hardware crash, node failure
  - Instantaneous error detection.
- Silent errors (aka silent data corruptions): e.g., soft faults in L1 cache, ALU, multiple bit flip due to cosmic radiation.
  - Detected only when corrupted data leads to unexpected results, which could happen long after its occurrence.
  - Become a serious concern in Exascale systems.

General-purpose approach

Periodic checkpoint, rollback and recovery:

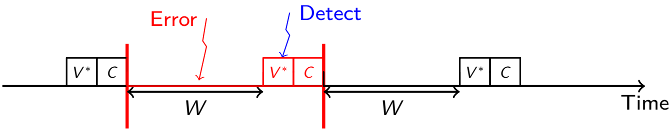


- Fail-stop errors: e.g., hardware crash, node failure
  - Instantaneous error detection.
- Silent errors (aka silent data corruptions): e.g., soft faults in L1 cache, ALU, multiple bit flip due to cosmic radiation.
  - Detected only when corrupted data leads to unexpected results, which could happen long after its occurrence.
  - Become a serious concern in Exascale systems.

Detection latency ⇒ risk of saving corrupted checkpoint!

Coping with silent errors

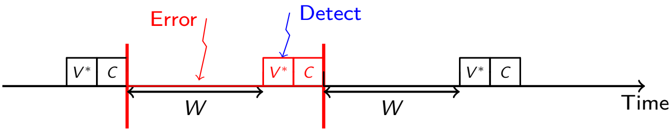
Couple checkpointing with verification:



- Before each checkpoint, run some verification mechanism (checksum, ECC, coherence tests, TMR, etc).
- Silent error is detected by verification ⇒ checkpoint always valid ☺

Coping with silent errors

Couple checkpointing with verification:



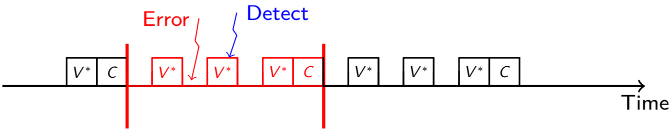
- Before each checkpoint, run some verification mechanism (checksum, ECC, coherence tests, TMR, etc).
- Silent error is detected by verification ⇒ checkpoint always valid ☺

What is the optimal checkpointing period (Young/Daly)?

	Fail-stop (classical)	Silent errors
Pattern	$T = W + C$	$T = W + V^* + C$
Optimal	$W^* = \sqrt{2C\mu}$	$W^* = \sqrt{(C + V^*)\mu}$

One step further: intermediate verifications

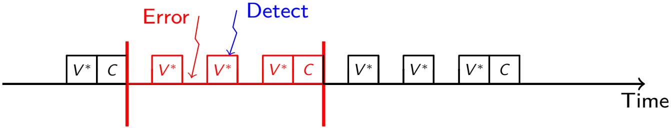
Perform several intermediate verifications before each checkpoint:



- Pro: silent error is detected earlier in the execution ☺
- Con: additional overhead in error-free executions ☹

One step further: intermediate verifications

Perform several intermediate verifications before each checkpoint:



- Pro: silent error is detected earlier in the execution ☺
- Con: additional overhead in error-free executions ☹

What is the optimal tradeoff?

One more step further: partial verification

Guaranteed/perfect verifications (V^*) can be very expensive!

Partial verifications (V) are available for many HPC applications!

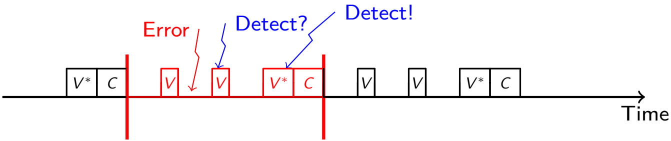
- Lower accuracy: recall (r) $= \frac{\#\text{detected errors}}{\#\text{total errors}} < 1$ ☹
- Much lower cost, i.e., $V \ll V^*$ ☺

One more step further: partial verification

Guaranteed/perfect verifications (V^*) can be very expensive!

Partial verifications (V) are available for many HPC applications!

- Lower accuracy: recall $(r) = \frac{\#\text{detected errors}}{\#\text{total errors}} < 1$ ☹
- Much lower cost, i.e., $V \ll V^*$ ☺

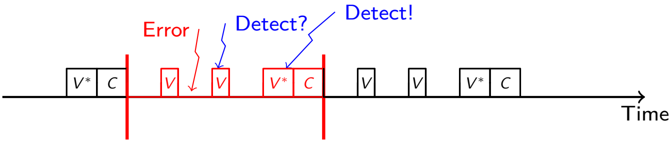


One more step further: partial verification

Guaranteed/perfect verifications (V^*) can be very expensive!

Partial verifications (V) are available for many HPC applications!

- Lower accuracy: recall $(r) = \frac{\#\text{detected errors}}{\#\text{total errors}} < 1$ ☹
- Much lower cost, i.e., $V \ll V^*$ ☺



What is the optimal checkpointing period?
How many partial verifications to use and their positions?

Outline

Model and Objective

Failure Model

- Silent errors strike randomly and are uniformly distributed with arrival rate $\lambda = 1/\mu$, where μ is platform MTBF.
  - Expect λT errors in computation of time T.
- Failures only affect computations; checkpointing, recovery, and verifications are protected.

Resilience parameters

- Cost of checkpointing C, cost of recovery R.
- Partial verification: cost V and recall $r < 1$.
- Guaranteed verification: cost V^* and recall $r^* = 1$.

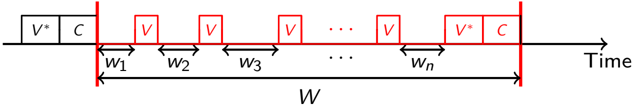
Objective

- Design an optimal periodic computing pattern that minimizes execution time (or makespan) of the application.

Pattern

Formally, a periodic computing pattern is defined by

- W: work length of the pattern (or period);
- n: number of segments in the pattern (or $m = n - 1$: number of partial verifications);
- $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_n]$: work fraction of each segment (or relative positions of partial verifications)
  - $\alpha_i = \frac{w_i}{W}$ and $\sum_{i=1}^{n} \alpha_i = 1$.



Last verification is perfect to avoid saving corrupted checkpoints.

Outline

Expected execution time of a pattern

Proposition

The expected time to execute a pattern with fixed W, n, α is

$$\mathbb{E}(W) = W + \underbrace{(n-1)V + V^* + C}_{o_{ff}} + \underbrace{\lambda W}_{\#errors} \left(\underbrace{\alpha^T A \alpha}_{f_{re}} \cdot W \right) + o(\lambda)$$

where A is a symmetric matrix defined by $A_{i,j} = \frac{1}{2}\left(1 + (1-r)^{|i-j|}\right)$.

Remarks:

- Two key parameters
  - o_{ff}: overhead in a fault-free execution.
  - f_{re}: fraction of re-executed work in case of fault.
- Same result if assuming exponential error distribution with first-order approximation (as in Young/Daly's classic formula).

Minimizing makespan

- Matrix A is essential to analysis. For instance, when $n = 4$ we have:

$$A = \frac{1}{2}\begin{bmatrix} 2 & 1+(1-r) & 1+(1-r)^2 & 1+(1-r)^3 \\ 1+(1-r) & 2 & 1+(1-r) & 1+(1-r)^2 \\ 1+(1-r)^2 & 1+(1-r) & 2 & 1+(1-r) \\ 1+(1-r)^3 & 1+(1-r)^2 & 1+(1-r) & 2 \end{bmatrix}$$

Minimizing makespan

- Matrix A is essential to analysis. For instance, when $n = 4$ we have:

$$A = \frac{1}{2}\begin{bmatrix} 2 & 1+(1-r) & 1+(1-r)^2 & 1+(1-r)^3 \\ 1+(1-r) & 2 & 1+(1-r) & 1+(1-r)^2 \\ 1+(1-r)^2 & 1+(1-r) & 2 & 1+(1-r) \\ 1+(1-r)^3 & 1+(1-r)^2 & 1+(1-r) & 2 \end{bmatrix}$$

- For an application with total work T_{base}, the makespan T_{final} is

$$T_{\text{final}} \approx \frac{\mathbb{E}(W)}{W} \cdot T_{\text{base}} = (1 + H(W)) \cdot T_{\text{base}}$$

where $H(W)$ is the total execution overhead given by

$$H(W) = \frac{\mathbb{E}(W)}{W} - 1 = \frac{o_{\text{ff}}}{W} + \lambda W f_{\text{re}} + o\left(\sqrt{\lambda}\right)$$

e.g., if $T_{\text{base}} = 100$ and $T_{\text{final}} = 120$, we have $H(W) = 20\%$.

Minimizing makespan

- Matrix A is essential to analysis. For instance, when $n = 4$ we have:

$$A = \frac{1}{2}\begin{bmatrix} 2 & 1+(1-r) & 1+(1-r)^2 & 1+(1-r)^3 \\ 1+(1-r) & 2 & 1+(1-r) & 1+(1-r)^2 \\ 1+(1-r)^2 & 1+(1-r) & 2 & 1+(1-r) \\ 1+(1-r)^3 & 1+(1-r)^2 & 1+(1-r) & 2 \end{bmatrix}$$

- For an application with total work T_{base}, the makespan T_{final} is

$$T_{\text{final}} \approx \frac{\mathbb{E}(W)}{W} \cdot T_{\text{base}} = (1 + H(W)) \cdot T_{\text{base}}$$

where $H(W)$ is the total execution overhead given by

$$H(W) = \frac{\mathbb{E}(W)}{W} - 1 = \frac{o_{\text{ff}}}{W} + \lambda W f_{\text{re}} + o\left(\sqrt{\lambda}\right)$$

e.g., if $T_{\text{base}} = 100$ and $T_{\text{final}} = 120$, we have $H(W) = 20\%$.

Minimizing makespan is equivalent to minimizing overhead!

Optimal work length

Theorem

The execution overhead of a pattern is minimized when its length is

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

The optimal overhead is

$$H(W^*) = 2\sqrt{\lambda o_{ff} f_{re}} + o(\sqrt{\lambda}).$$

- When the platform MTBF $\mu = 1/\lambda$ is large, $o(\sqrt{\lambda})$ is negligible.
- Minimizing overhead is equivalent to minimizing product $o_{\text{ff}} f_{\text{re}}$.
  - Tradeoff between fault-free overhead and fault-induced re-execution.

Optimal segment lengths

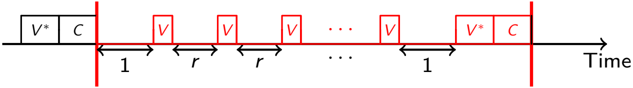
Theorem

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

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

and the optimal value of f_{re} is

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



Optimal segment lengths

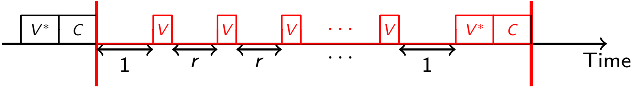
Theorem

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

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

and the optimal value of f_{re} is

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



Special case: if all verifications are perfect, we get equal-length segments, i.e., $\alpha_k^* = \frac{1}{n}, \forall 1 \leq k \leq n$ and $f_{\text{re}}^* = \frac{1}{2}\left(1 + \frac{1}{n}\right)$.

Optimal number of segments

Theorem

The execution overhead of a pattern is minimized when the number of segments is

$$n^* = \begin{cases} 1 - \frac{1}{a} + \sqrt{\frac{1}{a}\left(\frac{1}{b} - \frac{1}{a}\right)} & \text{if } \frac{a}{b} > 2 \\ 1 & \text{if } \frac{a}{b} \leq 2 \end{cases}$$

and the optimal overhead is

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

where $a = \frac{r}{2-r}$ represents accuracy and $b = \frac{V}{C+V^*}$ denotes relative cost of the partial verification.

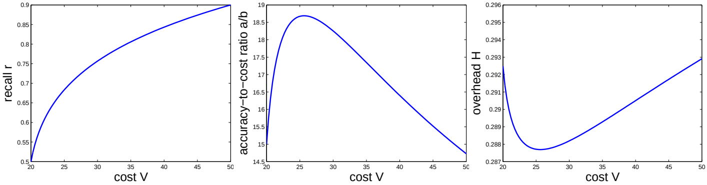
- In practice, the number of segments can only be an integer. Thus, the optimal number is either $\lceil n^* \rceil$ or $\lfloor n^* \rfloor$.

Optimal accuracy-cost tradeoff

Suppose a tradeoff exists between the cost V and recall r of a partial verification. What is the optimal tradeoff?

Theorem

The execution overhead is minimized when the (V, r) pair maximizes the accuracy-to-cost ratio $\frac{a}{b} = \frac{\frac{r}{2-r}}{\frac{V}{V^*+C}}$



Remark:

- The result is based on the optimal fractional solution (n^*). Thus, the overhead in the optimal integer solution contains rounding error, which, however, is small for practical parameter settings.

Outline

Evaluation setup

Parameters in Exascale Platform:

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

Evaluation setup

Parameters in Exascale Platform:

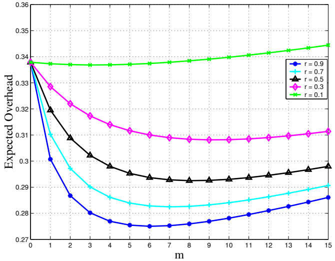
- 10^5 computing nodes with individual MTBF of 100 years
  $\Rightarrow$ platform MTBF $\mu \approx 8.7$ hours.
- Checkpoint size of 300GB with throughput of 0.5GB/s
  $\Rightarrow C = 600s = 10$ mins, and V^* in same order.
- Partial verifications (from Argonne National Laboratory, USA)
  $\Rightarrow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

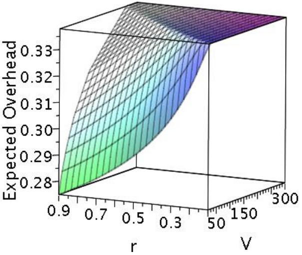
e.g., $C = 600$, $V^* = 300$, $V = 30$ and $r = 0.8$.

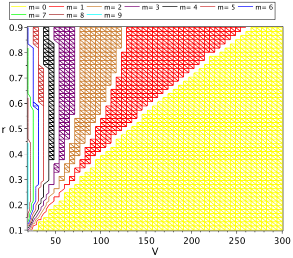
	using partial verifications	using perfect verifications
W	$7335s \approx 2$ hours	$5328s \approx 1.5$ hours
n	6	2
α	(0.19, 0.15, 0.15, 0.15, 0.15, 0.19)	(0.5, 0.5)
H	28.6%	33.8%

Using partial verifications gains 5% improvement in overhead.
$\Rightarrow$ Saving 1 hour for every 20 hours of computation!

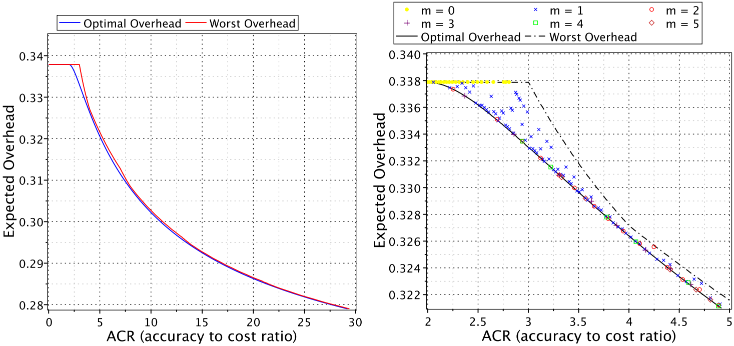
Impacts of m, V and r







Impact of ACR and rounding error



- Overhead decreases for increased accuracy-to-cost ratio (ACR).
- Different (V, r) pair could share same ACR with different m^*, H^*.
- Rounding error to theoretical optimal overhead H^* is insignificant.

Outline

Conclusion

Summary

- A first analysis of computing patterns to include partial verifications for silent error detection.
- Theoretically: derive the optimal pattern parameters, i.e., period, number of partial verifications and their positions.
- Practically: assess and compare the performance of the optimal pattern with realistic parameters.

Conclusion

Summary

- A first analysis of computing patterns to include partial verifications for silent error detection.
- Theoretically: derive the optimal pattern parameters, i.e., period, number of partial verifications and their positions.
- Practically: assess and compare the performance of the optimal pattern with realistic parameters.

Future work

- Partial verifications with false positives/alarms

$$\textit{precision}(p) = \frac{\#\text{true errors}}{\#\text{detected errors}} < 1.$$

- Coexistence of fail-stop and silent errors.