Resilient algorithms for coping with silent errors

Anne Benoit

ENS Lyon

Anne.Benoit@ens-lyon.fr

http://graal.ens-lyon.fr/~abenoit/

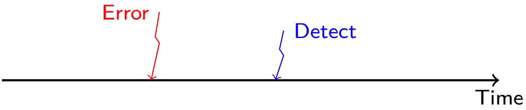
CR02 - 2016/2017

Outline

What is silent error?

- Fail-stop error: e.g., hardware crash, node failure
  - Instantaneous error detection.
- Silent error (a.k.a. silent data corruption, or SDC): e.g., soft faults in L1 cache, ALU, multiple bit flip due to cosmic radiation.
  - Cannot always be detected by ECC memory.

Silent error detected only when corrupted data is activated, which could happen long after the occurrence.



Quotes

- Soft Error: An unintended change in the state of an electronic device that alters the information that it stores without destroying its functionality, e.g. a bit flip caused by a cosmic-ray-induced neutron. (*Hengartner et al., 2008*)
- SDC occurs when incorrect data is delivered by a computing system to the user without any error being logged (*Cristian Constantinescu, AMD*)
- Silent errors are the black swan of errors (*Marc Snir*)

Fear of the Unknown

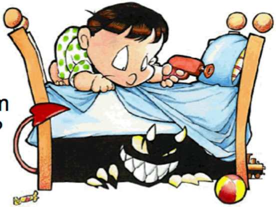
Hard errors – permanent component failure either HW or SW (hung or crash)

Transient errors –a blip or short term failure of either HW or SW

Silent errors – undetected errors either hard or soft, due to lack of detectors for a component or inability to detect (transient effect too short). Real danger is that answer may be incorrect but the user wouldn't know.

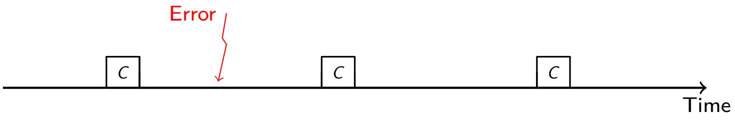
Statistically, silent error rates are increasing.
Are they really? Its fear of the unknown

Are silent errors really a problem
or just monsters under our bed?



General-purpose approach

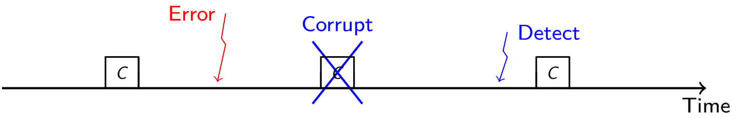
Periodic checkpointing, rollback and recovery:



- Works fine for fail-stop errors.

General-purpose approach

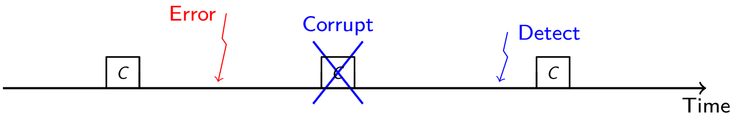
Periodic checkpointing, rollback and recovery:



- Works fine for fail-stop errors.
- Detection latency in silent errors $\Rightarrow$ risk of saving corrupted checkpoint(s).

General-purpose approach

Periodic checkpointing, rollback and recovery:



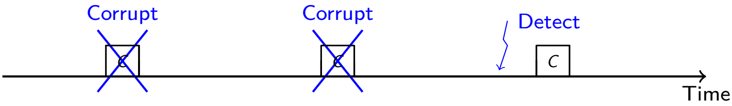
- Works fine for fail-stop errors.
- Detection latency in silent errors ⇒ risk of saving corrupted checkpoint(s).

Maintaining multiple checkpoints (*Lu, Zheng and Chien, 2013*)

- Requires more stable storage.
- Which checkpoint to roll back to?
- Critical failure when all live checkpoints are invalid.

General-purpose approach

Periodic checkpointing, rollback and recovery:



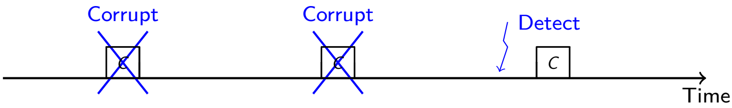
- Works fine for fail-stop errors.
- Detection latency in silent errors ⇒ risk of saving corrupted checkpoint(s).

Maintaining multiple checkpoints (*Lu, Zheng and Chien, 2013*)

- Requires more stable storage.
- Which checkpoint to roll back to?
- Critical failure when all live checkpoints are invalid.

General-purpose approach

Periodic checkpointing, rollback and recovery:



- Works fine for fail-stop errors.
- Detection latency in silent errors ⇒ risk of saving corrupted checkpoint(s).

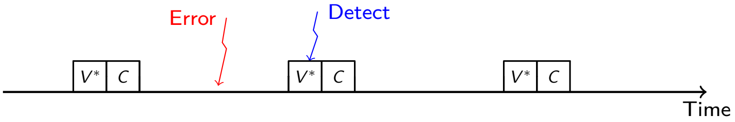
Maintaining multiple checkpoints (*Lu, Zheng and Chien, 2013*)

- Requires more stable storage.
- Which checkpoint to roll back to?
- Critical failure when all live checkpoints are invalid.

Need to know when silent error occurred.

Coping with silent errors

Couple checkpointing with verification:



- Before each checkpoint, run some verification mechanism or error detection test (some examples in next slide).
- Silent error, if any, is detected by verification $\Rightarrow$ need to maintain only one checkpoint, which is always valid ☺

Methods for detecting silent errors

General-purpose methods

- Checksum, error correcting code, coherence tests.
- Triple modular redundancy and voting.

Application-specific methods

- 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*).

Methods for detecting silent errors

On-line ABFT scheme for PCG (*Chen, 2013*)

Compute $r^{(0)} = b - Ax^{(0)}, z^{(0)} = M^{-1}r^{(0)}, p^{(0)} = z^{(0)}$,
  and $\rho_0 = {r^{(0)}}^T z^{(0)}$ for some initial guess $x^{(0)}$
checkpoint: A, M, and b
for $i = 0, 1, \ldots$
  if ((i>0) and ($i\%d = 0$))
    if ($\frac{{p^{(i+1)}}^T q^{(i)}}{||p^{(i+1)}||.||q^{(i)}||} > 10^{-10}$
      or $\frac{||r^{(i+1)} + Ax^{(i+1)} - b||}{||b||.||A||} > 10^{-10}$)
      recover: A, M, b, i, ρ_i,
        $p^{(i)}$, $x^{(i)}$, and $r^{(i)}$.
    else if ($i\%(cd) = 0$)
      checkpoint: i, ρ_i, $p^{(i)}$, and $x^{(i)}$
    endif
  endif
  $q^{(i)} = Ap^{(i)}$
  $\alpha_i = \rho_i / {p^{(i)}}^T q^{(i)}$
  $x^{(i+1)} = x^{(i)} + \alpha_i p^{(i)}$
  $r^{(i+1)} = r^{(i)} - \alpha_i q^{(i)}$
  solve $Mz^{(i+1)} = r^{(i+1)}$, where $M = M^T$
  $\rho_{i+1} = {r^{(i+1)}}^T z^{(i+1)}$
  $\beta_i = \rho_{i+1}/\rho_i$
  $p^{(i+1)} = z^{(i+1)} + \beta_i p^{(i)}$
  check convergence; continue if necessary
end

- Iterate PCG
  Cost: SpMV, preconditioner solve, 5 linear kernels
- Detect soft errors by checking orthogonality and residual
- Verification every *d* iterations
  Cost: scalar product+SpMV
- Checkpoint every *c* iterations
  Cost: three vectors, or two vectors + SpMV at recovery
- Experimental method to choose *c* and *d*

Methods for detecting silent errors

Data analytics methods

- 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

Data analytics methods

- 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*).

Some verifications are guaranteed to detect all the errors.
Some are not always accurate ⇒ partial verifications.

- ☹ Lower accuracy
- ☺ Much lower cost

Methods for detecting silent errors

Data analytics methods

- 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*).

Some verifications are guaranteed to detect all the errors.
Some are not always accurate ⇒ partial verifications.

- ☹ Lower accuracy
- ☺ Much lower cost

Approach is agnostic of the nature of verification mechanism.

Outline

Models and Objective

Failure model

- Silent errors arrive following exponential law $Exp(\lambda)$
  $\Rightarrow$ memoryless.
- Error rate $\lambda = \frac{1}{\mu}$ with Mean Time Between Failure (MTBF) μ.
- Probability of having an error in a computation of length w

$$\begin{aligned} \mathbb{P}(X \leq w) &= 1 - e^{-\lambda w} && \text{(by definition)} \\ &\approx \lambda w && \text{(Taylor expansion } e^x = \textstyle\sum_{n=0}^{\infty} \frac{x^n}{n!}\text{)} \end{aligned}$$

  $\Rightarrow$ same as uniform distribution in first-order approximation.
- Errors strike computation only, not checkpointing, recovery, and verification.
  $\Rightarrow$ much simplified analysis, but same asymptotic results in first-order approximation.

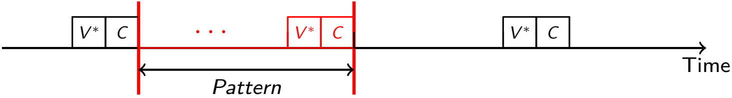
Models and Objective

Resilience parameters

- C: Cost of checkpointing;
- R: Cost of recovery;
- V^*: Cost of perfect/guaranteed verification;
- V: Cost of partial verification.

Objective

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



Last verification of a pattern is always perfect to avoid saving corrupted checkpoints.

Models and Objective

Overhead and Waste

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

$$\begin{aligned} W_{\text{final}} &\approx \frac{W_{\text{base}}}{W} \cdot \mathbb{E}(W) \\ &= (1 + \textsc{Overhead}) \cdot W_{\text{base}} \\ &= \frac{1}{1 - \textsc{Waste}} \cdot W_{\text{base}} \end{aligned}$$

where

$$\begin{aligned} \textsc{Overhead} &= \frac{\mathbb{E}(W)}{W} - 1 \\ \textsc{Waste} &= 1 - \frac{W}{\mathbb{E}(W)} \end{aligned}$$

denote the execution overhead and execution waste of the pattern, respectively.

Models and Objective

Proposition

For large applications, minimizing total execution time is equivalent to minimizing overhead or waste of a computing pattern.

E.x. $W = 100, \mathbb{E}(W) = 125 \Rightarrow \textsc{Overhead} = 25\%, \textsc{Waste} = 20\%.$

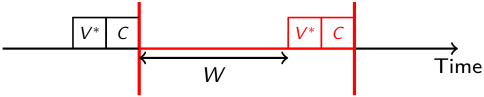
In fact, when platform MTBF μ is large, both overhead and waste are in the same order $O(\sqrt{\lambda})$.

Outline

1. Introduction
2. Problem statement
3. Computing optimal patterns
    - Revisiting Young/Daly (base pattern)
    - Pattern with guaranteed verifications
    - Interleaving checkpoints and verifications
    - Pattern with partial verifications
    - Using multiple types of partial verifications
4. Coping with both fail-stop and silent errors
5. Algorithms for a linear chain of tasks
6. Conclusion

Outline

Revisiting Young/Daly (Base Pattern P_c)



Proposition

The expected time to execute a base pattern P_c of work length W is

$$\mathbb{E}(W) = W + V^* + C + \lambda W(W + V^* + R) + O(\lambda^2 W^3)$$

Proof. First, express the expected execution time recursively:

$$\mathbb{E}(W) = W + V^* + (1 - e^{-\lambda W}) \cdot (R + \mathbb{E}(W)) + e^{-\lambda W} \cdot C$$

Then, solve the recursion and take first-order approximation.

Approximation is accurate if platform MTBF is large in front of the resilience parameters.

Revisiting Young/Daly (Base Pattern P_c)

Proposition

The optimal work length W^* of the base pattern P_c is

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

and the optimal expected overhead is

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

Proof. Derive the overhead from the expected execution time:

$$\begin{aligned} \textsc{Overhead} &= \frac{\mathbb{E}(W)}{W} - 1 \\ &= \frac{V^* + C}{W} + \lambda W + \lambda(V^* + R) + O(\lambda^2 W^2) \end{aligned}$$

Balance W to minimize $\textsc{Overhead}$.

Revisiting Young/Daly (Base Pattern P_c)

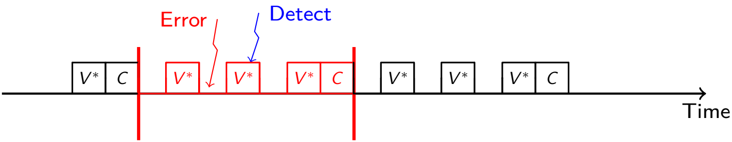
Recall from the waste analysis:

	Fail-stop errors	Silent errors
Pattern	$T = W + C$	$S = W + V^* + C$
WASTE_{ff}	$\frac{C}{T}$	$\frac{V^*+C}{S}$
$\text{WASTE}_{\text{fail}}$	$\lambda(D + R + \frac{W}{2})$	$\lambda(R + W + V^*)$
Optimal	$T_{\text{opt}} = \sqrt{\frac{2C}{\lambda}}$	$S_{\text{opt}} = \sqrt{\frac{V^*+C}{\lambda}}$
$\text{WASTE}_{\text{opt}}$	$\sqrt{2\lambda C}$	$2\sqrt{\lambda(V^* + C)}$

Outline

Pattern with Guaranteed Verifications (P_{V^*C})

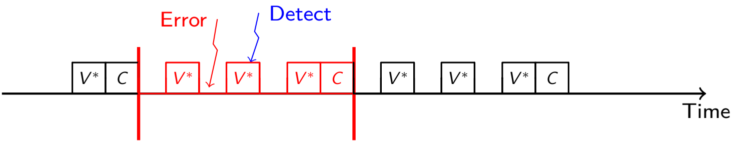
Perform several verifications before each checkpoint:



- ☺ silent error is detected earlier in the pattern.
- ☹ additional overhead in fault-free executions.

Pattern with Guaranteed Verifications (P_{V^*C})

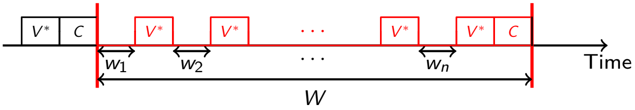
Perform several verifications before each checkpoint:



- ☺ silent error is detected earlier in the pattern.
- ☹ additional overhead in fault-free executions.

What is the optimal checkpointing period?
How many verifications to use?
Where are their positions?

Pattern with Guaranteed Verifications (P_{V^*C})



Proposition

Suppose a pattern P_{V^*C} *has length* W *and* n *segments. The* i*-th segment has work* $w_i = \alpha_i W$*. The expected time to execute the pattern is*

$$\mathbb{E}(W) = W + nV^* + C + \lambda W \left(f \cdot W + g \cdot V^* + R \right) + O(\lambda^2 W^3)$$

where

$$\begin{aligned} f &= \sum_{i=1}^{n} \alpha_i \left(\sum_{j=1}^{i} \alpha_j \right) \\ g &= \sum_{i=1}^{n} i \cdot \alpha_i \end{aligned}$$

Pattern with Guaranteed Verifications (P_{V^*C})

Proof. Recursive expression for expected execution time:

$$\mathbb{E}(W) = \sum_{i=1}^{n} \left(e^{-\lambda \sum_{j=1}^{i-1} w_j} \cdot (1 - e^{-\lambda w_i}) \cdot \left(\sum_{j=1}^{i} w_j + i \cdot V^* + R + \mathbb{E}(W) \right) \right)$$
$$+ e^{-\lambda W} \left(W + nV^* + C \right)$$

For instance, when $n = 3$, i.e., $W = w_1 + w_2 + w_3$

$$\begin{aligned}
\mathbb{E}(W) = \; & (1 - e^{-\lambda w_1}) \big(w_1 + V^* + R + \mathbb{E}(W) \big) \\
& + e^{-\lambda w_1} \left(1 - e^{-\lambda w_2}\right) \left(w_1 + w_2 + 2V^* + R + \mathbb{E}(W) \right) \\
& + e^{-\lambda (w_1 + w_2)} \left(1 - e^{-\lambda w_3}\right) \left(w_1 + w_2 + w_3 + 3V^* + R + \mathbb{E}(W) \right) \\
& + e^{-\lambda W} \left(W + 3V^* + C \right)
\end{aligned}$$

Approximate after solving the recursion.

Pattern with Guaranteed Verifications (P_{V^*C})

Proposition

The optimal work length W^*, the optimal number n^* of segments, and the optimal positions of the verifications in pattern P_{V^*C} satisfy

$$
\begin{aligned}
n^* &= \sqrt{\frac{C}{V^*}} \\
W^* &= \sqrt{\frac{n^* V^* + C}{\frac{1}{2}\left(1 + \frac{1}{n^*}\right)\lambda}} \\
\alpha_i^* &= \frac{1}{n^*} \text{ for all } i = 1, 2, \ldots, n^*
\end{aligned}
$$

and the optimal expected overhead is

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

Practically, the number of segments must be a positive integer, i.e., $\max(1, \lfloor n^* \rfloor)$ or $\lceil n^* \rceil$.

Pattern with Guaranteed Verifications (P_{V^*C})

Proof. Derive the overhead from the expected execution time:

$$\textsc{Overhead} = \frac{nV^* + C}{W} + \lambda f \cdot W + \lambda\left(g \cdot V^* + R\right) + O(\lambda^2 W^2)$$

① Optimize W

$$W^* = \sqrt{\frac{nV^* + C}{\lambda f}} \Rightarrow \textsc{Overhead} \approx 2\sqrt{\lambda f\left(nV^* + C\right)}$$

② Convex function $f = \sum_{i=1}^{n} \alpha_i \left(\sum_{j=1}^{i} \alpha_j\right)$ minimized when $\alpha_i = \frac{1}{n}$

$$f^* = \frac{1}{2}\left(1 + \frac{1}{n}\right) \Rightarrow \textsc{Overhead} \approx \sqrt{2\lambda\left(nV^* + V^* + C + \frac{C}{n}\right)}$$

③ Optimize n

$$n^* = \sqrt{\frac{C}{V^*}} \Rightarrow \textsc{Overhead} \approx \sqrt{2\lambda\left(\sqrt{V^*} + \sqrt{C}\right)^2}$$

Some Observations

Observation 1

The expected time to execute a pattern of length W is

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

with two important parameters

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

Some Observations

Derive the overhead from the expected execution time:

$$\begin{aligned}\textsc{Overhead} &= \frac{\mathbb{E}(W)}{W} - 1 \\ &= \frac{o_{\text{ff}}}{W} + \lambda f_{\text{re}} W + O(\lambda)\end{aligned}$$

Observation 2

The optimal work length and the optimal overhead of a pattern are

$$\begin{aligned}W^* &= \sqrt{\frac{o_{\text{ff}}}{\lambda f_{\text{re}}}} \\ \textsc{Overhead}^* &= 2\sqrt{\lambda \cdot f_{\text{re}} o_{\text{ff}}} + O(\lambda)\end{aligned}$$

Asymptotically, minimizing overhead is equivalent to minimizing the product $f_{\text{re}} o_{\text{ff}}$!

Some Observations

Base pattern P_c

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

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

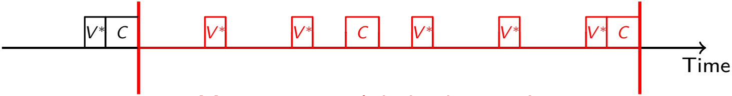
Pattern P_{v^*c}

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

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

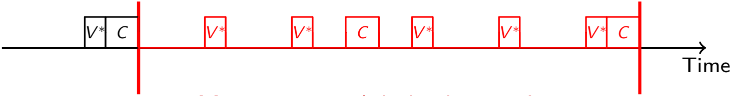
Outline

Interleaving Checkpoints and Verifications



May store invalid checkpoints!

Interleaving Checkpoints and Verifications



May store invalid checkpoints!

BALANCEDALGORITHM (*Benoit, Raina and Robert, 2014*)

① Equipartition p checkpoints and q guaranteed verifications.

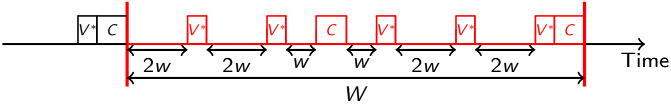
- $p \leq q \Rightarrow$ need only two checkpoints in memory.
- $gcd(p, q) = 1 \Rightarrow$ no verified checkpoint in the pattern.

② After each successful verification, mark preceding checkpoint valid.

③ After detecting an error, roll back to the last checkpoint.

- If marked valid, recover from this checkpoint.
- Otherwise, verify this checkpoint
  - If valid, recover from this checkpoint and mark it valid.
  - If invalid, recover from the preceding checkpoint (valid).

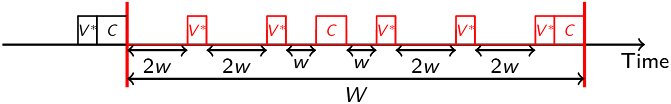
Interleaving Checkpoints and Verifications



E.x. $p = 2$, $q = 5 \Rightarrow W = 10w$, six chunks of size w or $2w$

In this pattern, $o_{\text{ff}} = 2C + 5V^*$ and $f_{\text{re}} = \frac{7}{20}$

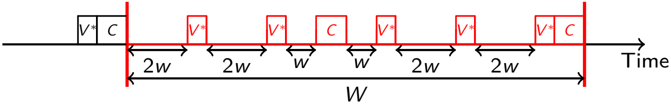
Interleaving Checkpoints and Verifications



E.x. $p = 2$, $q = 5$ $\Rightarrow$ $W = 10w$, six chunks of size w or $2w$
In this pattern, $o_{\mathrm{ff}} = 2C + 5V^*$ and $f_{\mathrm{re}} = \frac{7}{20}$

- (Prob. $\frac{2w}{W} = \frac{1}{5}$) $T_{\mathrm{re}} = R + \frac{1}{5}W + V^*$
- (Prob. $\frac{2w}{W} = \frac{1}{5}$) $T_{\mathrm{re}} = R + \frac{2}{5}W + 2V^*$
- (Prob. $\frac{w}{W} = \frac{1}{10}$) $T_{\mathrm{re}} = 2R + \frac{3}{5}W + C + 4V^*$
- (Prob. $\frac{w}{W} = \frac{1}{10}$) $T_{\mathrm{re}} = R + \frac{1}{10}W + 2V^*$
- (Prob. $\frac{2w}{W} = \frac{1}{5}$) $T_{\mathrm{re}} = R + \frac{3}{10}W + 2V^*$
- (Prob. $\frac{2w}{W} = \frac{1}{5}$) $T_{\mathrm{re}} = R + \frac{1}{2}W + 3V^*$

Interleaving Checkpoints and Verifications



E.x. $p = 2$, $q = 5$ $\Rightarrow$ $W = 10w$, six chunks of size w or $2w$
In this pattern, $o_{\mathrm{ff}} = 2C + 5V^*$ and $f_{\mathrm{re}} = \frac{7}{20}$

- (Prob. $\frac{2w}{W} = \frac{1}{5}$) $T_{\mathrm{re}} = R + \frac{1}{5}W + V^*$
- (Prob. $\frac{2w}{W} = \frac{1}{5}$) $T_{\mathrm{re}} = R + \frac{2}{5}W + 2V^*$
- (Prob. $\frac{w}{W} = \frac{1}{10}$) $T_{\mathrm{re}} = 2R + \frac{3}{5}W + C + 4V^*$
- (Prob. $\frac{w}{W} = \frac{1}{10}$) $T_{\mathrm{re}} = R + \frac{1}{10}W + 2V^*$
- (Prob. $\frac{2w}{W} = \frac{1}{5}$) $T_{\mathrm{re}} = R + \frac{3}{10}W + 2V^*$
- (Prob. $\frac{2w}{W} = \frac{1}{5}$) $T_{\mathrm{re}} = R + \frac{1}{2}W + 3V^*$

$$\boxed{\mathbb{E}(T_{\mathrm{re}}) = \frac{7}{20}W + O(R, V^*, C)}$$

$$W = \sqrt{\frac{20(2C + 5V^*)}{7\lambda}} \text{ and } \textsc{Overhead} \approx 2\sqrt{\lambda\frac{7(2C + 5V^*)}{20}}$$

Interleaving Checkpoints and Verifications

Theorem ($p = 1$)

The minimal value of $f_{re}(1, q)$ is obtained when all verifications are equi-spaced. In this case, we have $f^*_{re}(1, q) = \frac{1}{2}(1 + 1/q)$.

Interleaving Checkpoints and Verifications

Theorem ($p = 1$)

The minimal value of $f_{re}(1, q)$ is obtained when all verifications are equi-spaced. In this case, we have $f^*_{re}(1, q) = \frac{1}{2}(1 + 1/q)$.

Theorem ($p > 1$)

$f_{re}(p, q) \geq \frac{1}{2}(1/p + 1/q)$, *bound is matched by* BalancedAlgorithm.

Proof. Assess gain due to the $p - 1$ intermediate checkpoints.

$$\delta = f_{\mathrm{re}}(1, q) - f_{\mathrm{re}}(p, q) = \sum_{i=1}^{p} \left(\alpha_i \sum_{j=1}^{i-1} \alpha_j \right)$$

where α_i is the fraction of the i-th checkpointing segment.

Interleaving Checkpoints and Verifications

Theorem ($p = 1$)

The minimal value of $f_{re}(1, q)$ is obtained when all verifications are equi-spaced. In this case, we have $f^*_{re}(1, q) = \frac{1}{2}(1 + 1/q)$.

Theorem ($p > 1$)

$f_{re}(p, q) \geq \frac{1}{2}(1/p + 1/q)$, *bound is matched by* BALANCEDALGORITHM.

Proof. Assess gain due to the $p - 1$ intermediate checkpoints.

$$\delta = f_{\mathrm{re}}(1, q) - f_{\mathrm{re}}(p, q) = \sum_{i=1}^{p} \left(\alpha_i \sum_{j=1}^{i-1} \alpha_j \right)$$

where α_i is the fraction of the i-th checkpointing segment.

- δ maximized when $\alpha_i = 1/p$ for all $i \Rightarrow$ equi-spaced checkpoints.
- Hence, we have $\delta \leq \frac{1}{2}(1 - 1/p)$.
- $f_{\mathrm{re}}(p, q) = f_{\mathrm{re}}(1, q) - \delta \geq \frac{1}{2}(1/p + 1/q)$.

Interleaving Checkpoints and Verifications

Proposition

The optimal work length W^* and the optimal numbers p^* and q^* of the interleaving pattern satisfy

$$W^* = \sqrt{\frac{p^* C + q^* V^*}{\frac{1}{2}\left(\frac{1}{p^*} + \frac{1}{q^*}\right)\lambda}} \text{ and } \frac{q^*}{p^*} = \sqrt{\frac{C}{V^*}}$$

and the optimal expected overhead is

$$\textsc{Overhead}^* \approx \sqrt{2\lambda C} + \sqrt{2\lambda V^*}$$

Proof. We have $o_{\text{ff}} = pC + qV^*$ and $f_{\text{re}} = \frac{1}{2}\left(\frac{1}{p} + \frac{1}{q}\right)$.
Minimize $o_{\text{ff}} f_{\text{re}} = \frac{1}{2}\left(C + C/\gamma + \gamma V^* + V^*\right)$, where $\gamma = q/p \geq 1$.
Optimal $\gamma^* = \sqrt{C/V^*}$.

Interleaving Checkpoints and Verifications

Proposition

The optimal work length W^* and the optimal numbers p^* and q^* of the interleaving pattern satisfy

$$W^* = \sqrt{\frac{p^*C + q^*V^*}{\frac{1}{2}\left(\frac{1}{p^*} + \frac{1}{q^*}\right)\lambda}} \text{ and } \frac{q^*}{p^*} = \sqrt{\frac{C}{V^*}}$$

and the optimal expected overhead is

$$\textsc{Overhead}^* \approx \sqrt{2\lambda C} + \sqrt{2\lambda V^*}$$

Proof. We have $o_{\mathrm{ff}} = pC + qV^*$ and $f_{\mathrm{re}} = \frac{1}{2}\left(\frac{1}{p} + \frac{1}{q}\right)$.
Minimize $o_{\mathrm{ff}} f_{\mathrm{re}} = \frac{1}{2}\left(C + C/\gamma + \gamma V^* + V^*\right)$, where $\gamma = q/p \geq 1$.
Optimal $\gamma^* = \sqrt{C/V^*}$.

- When $p = 1$, same results as the pattern P_{V^*C}.
- E.x. $C = 9$ and $V^* = 4 \Rightarrow q^* = 3$ and $p^* = 2$ (avoid rounding).

Outline

Pattern with Partial Verifications (P_{VC})

Guaranteed/perfect verifications can be very expensive! Partial verifications are available for many HPC applications!

- ☺ Much lower cost, i.e., $V \ll V^*$
- ☹ Lower accuracy

$$\begin{aligned} \text{recall } (r) &= \frac{\#\text{detected errors}}{\#\text{total errors}} < 1 \text{ (false negative)} \\ \text{precision } (p) &= \frac{\#\text{true errors}}{\#\text{detected errors}} < 1 \text{ (false positive)} \end{aligned}$$

Pattern with Partial Verifications (P_{vc})

Guaranteed/perfect verifications can be very expensive! Partial verifications are available for many HPC applications!

- ☺ Much lower cost, i.e., $V \ll V^*$
- ☹ Lower accuracy

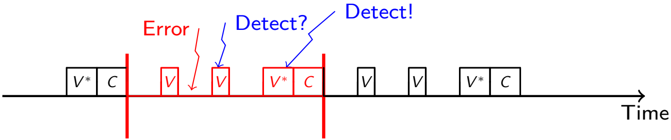
$$\text{recall } (r) = \frac{\#\text{detected errors}}{\#\text{total errors}} < 1 \text{ (false negative)}$$

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

In the following, assume $p = 1$.

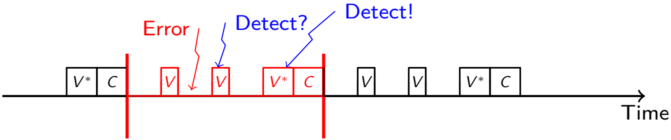
- Matched by many fault filters.
- $p < 1$ seems to render verification useless; real impact not well understood.

Pattern with Partial Verifications (P_{VC})



- A partial verification may miss an error (with probability $g = 1 - r$).
- Last verification is perfect to avoid saving invalid checkpoints.

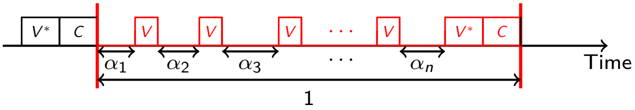
Pattern with Partial Verifications (P_{vc})



- A partial verification may miss an error (with probability $g = 1 - r$).
- Last verification is perfect to avoid saving invalid checkpoints.

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

Pattern with Partial Verifications (P_{vc})



(1) Apply the $f_{\mathrm{re}} o_{\mathrm{ff}}$ 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 matrix defined by $A_{i,j} = \frac{1}{2}\left(1 + g^{|i-j|}\right)$.

Pattern with Partial Verifications (P_{vc})

Proof. Derive the expected re-execution fraction.

$$f_{\text{re}} = \sum_{i=1}^{n} \alpha_i \left(\sum_{j=1}^{i} \alpha_j + \sum_{j=i+1}^{n} g^{j-i} \alpha_j \right)$$

Pattern with Partial Verifications (P_{vc})

Proof. Derive the expected re-execution fraction.

$$f_{\mathrm{re}} = \sum_{i=1}^{n} \alpha_i \left(\sum_{j=1}^{i} \alpha_j + \sum_{j=i+1}^{n} g^{j-i} \alpha_j \right)$$

E.x., when $n = 3$, i.e., $\alpha_1 + \alpha_2 + \alpha_3 = 1$.

$$f_{\mathrm{re}} = \begin{array}{l} \alpha_1 \left(\alpha_1 + g\alpha_2 + g^2\alpha_3\right) \\ +\alpha_2 \left(\alpha_1 + \alpha_2 + g\alpha_3\right) \\ +\alpha_3(\alpha_1 + \alpha_2 + \alpha_3) \end{array} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}^T \begin{bmatrix} 1 & g & g^2 \\ 1 & 1 & g \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \boldsymbol{\alpha}^T M \boldsymbol{\alpha}$$

Pattern with Partial Verifications (P_{vc})

Proof. Derive the expected re-execution fraction.

$$f_{\text{re}} = \sum_{i=1}^{n} \alpha_i \left(\sum_{j=1}^{i} \alpha_j + \sum_{j=i+1}^{n} g^{j-i} \alpha_j \right)$$

E.x., when $n = 3$, i.e., $\alpha_1 + \alpha_2 + \alpha_3 = 1$.

$$f_{\text{re}} = \begin{array}{l} \quad\alpha_1 \left(\alpha_1 + g\alpha_2 + g^2\alpha_3\right) \\ +\alpha_2 \left(\alpha_1 + \alpha_2 + g\alpha_3\right) \\ +\alpha_3(\alpha_1 + \alpha_2 + \alpha_3) \end{array} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}^T \begin{bmatrix} 1 & g & g^2 \\ 1 & 1 & g \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \boldsymbol{\alpha}^T M \boldsymbol{\alpha}$$

But M is not symmetric. Replace it by

$$A = \frac{M + M^T}{2} = \frac{1}{2} \begin{bmatrix} 2 & 1+g & 1+g^2 \\ 1+g & 2 & 1+g \\ 1+g^2 & 1+g & 2 \end{bmatrix}$$

(2) Minimize $f_{\mathbf{re}}$.

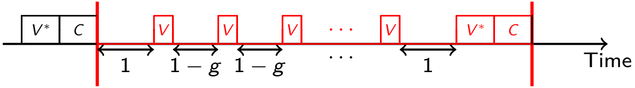
Proposition

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

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

and the optimal value of f_{re} is

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



(2) Minimize $f_{\mathbf{re}}$.

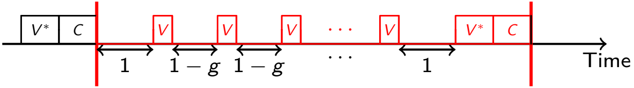
Proposition

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

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

and the optimal value of f_{re} is

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



If all verifications are perfect ($g = 0$), we retrieve equal-length segments, i.e., $\alpha_i^* = \frac{1}{n}$ for all $1 \leq i \leq n$ and $f_{\mathrm{re}}^* = \frac{1}{2}\left(1 + \frac{1}{n}\right)$.

Proof. Quadratic optimization (define $c = [1, 1, \ldots, 1]^T$):

$$\begin{array}{ll} \text{minimize} & f_{\text{re}} = \boldsymbol{\alpha}^T A \boldsymbol{\alpha} \\ \text{subject to} & \mathbf{c}^T \boldsymbol{\alpha} = 1 \end{array}$$

If matrix A is *symmetric positive definite (SPD)*, unique global minimum

$$\begin{aligned} f_{\text{re}}^{\text{opt}} &= \frac{1}{\mathbf{c}^T A^{-1} \mathbf{c}} \\ \boldsymbol{\alpha}^{\text{opt}} &= \frac{A^{-1}\mathbf{c}}{\mathbf{c}^T A^{-1} \mathbf{c}} \end{aligned}$$

Pattern with Partial Verifications (P_{vc})

Proof. Quadratic optimization (define $c = [1, 1, \ldots, 1]^T$):

$$\begin{array}{ll} \text{minimize} & f_{\text{re}} = \boldsymbol{\alpha}^T A \boldsymbol{\alpha} \\ \text{subject to} & \mathbf{c}^T \boldsymbol{\alpha} = 1 \end{array}$$

If matrix A is *symmetric positive definite (SPD)*, unique global minimum

$$\begin{aligned} f_{\text{re}}^{\text{opt}} &= \frac{1}{\mathbf{c}^T A^{-1} \mathbf{c}} \\ \boldsymbol{\alpha}^{\text{opt}} &= \frac{A^{-1}\mathbf{c}}{\mathbf{c}^T A^{-1} \mathbf{c}} \end{aligned}$$

We will prove:

- A is SPD.
- $A\boldsymbol{\alpha}^* = f_{\text{re}}^* \mathbf{c}$.

$$\begin{aligned} &\Rightarrow \quad \boldsymbol{\alpha}^* = f_{\text{re}}^* A^{-1} \mathbf{c} \\ &\Rightarrow \quad 1 = \mathbf{c}^T \boldsymbol{\alpha}^* = f_{\text{re}}^* (\mathbf{c}^T A^{-1} \mathbf{c}) \\ &\Rightarrow \quad f_{\text{re}}^* = \frac{1}{\mathbf{c}^T A^{-1} \mathbf{c}} = f_{\text{re}}^{\text{opt}} \\ &\Rightarrow \quad \boldsymbol{\alpha}^* = \frac{A^{-1}\mathbf{c}}{\mathbf{c}^T A^{-1} \mathbf{c}} = \boldsymbol{\alpha}^{\text{opt}} \end{aligned}$$

Pattern with Partial Verifications (P_{vc})

Proposition

$A^{(n)}$ is symmetric positive definite (SPD), for all $n \geq 1$.

Proof. Show all leading principle minors are positive, by induction.

Pattern with Partial Verifications (P_{vc})

Proposition

$A^{(n)}$ is symmetric positive definite (SPD), for all $n \geq 1$.

Proof. Show all leading principle minors are positive, by induction.

Base case: $A^{(1)} = [1]$ and $\det\left(A^{(1)}\right) = 1$.

Inductive step: Suppose $\det\left(A^{(k)}\right) > 0$ for all $k = 1, 2, \cdots, n-1$.
Using co-factor method,

$$\left(A^{(n)}\right)^{-1}_{1,1} = \frac{\det\left(A^{(n-1)}\right)}{\det\left(A^{(n)}\right)}$$

In fact, the inverse of $A^{(n)}$ is known! (*Dow, 2003*)

$$\left(A^{(n)}\right)^{-1}_{1,1} = \frac{2(n(1-g)+4g)}{(1-g^2)(n(1-g)+1+3g)} > 0$$

Proposition

$A^{(n)}$ is symmetric positive definite (SPD), for all $n \geq 1$.

Proof. Show all leading principle minors are positive, by induction.

Base case: $A^{(1)} = [1]$ and $\det\left(A^{(1)}\right) = 1$.

Inductive step: Suppose $\det\left(A^{(k)}\right) > 0$ for all $k = 1, 2, \cdots, n-1$.
Using co-factor method,

$$\left(A^{(n)}\right)^{-1}_{1,1} = \frac{\det\left(A^{(n-1)}\right)}{\det\left(A^{(n)}\right)}$$

In fact, the inverse of $A^{(n)}$ is known! (*Dow, 2003*)

$$\left(A^{(n)}\right)^{-1}_{1,1} = \frac{2(n(1-g)+4g)}{(1-g^2)(n(1-g)+1+3g)} > 0$$

We can even compute the determinant of $A^{(n)}$:

$$\det\left(A^{(n)}\right) = \frac{(1-g)^{n-1}(1+g)^{n-2}((n-3)(1-g)+4)}{2^n}$$

Proposition

$A\boldsymbol{\alpha}^* = f_{re}^* \mathbf{c}$

Proof. Write $A = \frac{1}{2}(J + B)$, where J is all-one matrix and $B_{i,j} = g^{|i-j|}$.

Write $\boldsymbol{\alpha}^* = \frac{\boldsymbol{\beta}^*}{(n-2)(1-g)+2}$, where $\beta_i^* = \begin{cases} 1 & \text{for } i = 1, n \\ 1-g & \text{for } 1 < i < n \end{cases}$

$$\begin{aligned} &\Leftarrow \quad \frac{1}{2}(J+B)\boldsymbol{\alpha}^* = \frac{1}{2}\left(1 + \frac{1+g}{(n-2)(1-g)+2}\right)\mathbf{c} \\ &\Leftarrow \quad B\boldsymbol{\alpha}^* = \frac{1+g}{(n-2)(1-g)+2}\mathbf{c}, \text{ since } J\boldsymbol{\alpha}^* = \mathbf{c} \\ &\Leftarrow \quad B\boldsymbol{\beta}^* = (1+g)\mathbf{c} \end{aligned}$$

We can show $(B\boldsymbol{\beta}^*)_i = 1 + g$ for all $1 \le i \le n$.

Pattern with Partial Verifications (P_{vc})

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

Proposition

The optimal number of segments in the pattern P_{vc} *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 expected overhead is

$$\textsc{Overhead}^* \approx \sqrt{2\lambda(V^* + C)}\left(\sqrt{1 - \frac{1}{\phi}} + \sqrt{\frac{1}{\phi}}\right)$$

where $a = \frac{1-g}{1+g}$ *represents accuracy,* $b = \frac{V}{V^*+C}$ *denotes relative cost, and* $\phi = \frac{a}{b}$ *is the accuracy-to-cost ratio of the partial verification.*

Use partial verification only when its accuracy-to-cost ratio $\phi > 2$.

Pattern with Partial Verifications (P_{vc})

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 (from Argonne National Laboratory, USA)
  $\Rightarrow V$ typically tens of seconds, and $r \in [0.5, 0.95]$.

Pattern with Partial Verifications (P_{vc})

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 (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 $r = 0.8$.

	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
$\boldsymbol{\alpha}^*$	$\alpha_i = \begin{cases} 0.20, i = 1, 6 \\ 0.15, i = 2..5 \end{cases}$	$[0.5, 0.5]$	$[1]$
O.H.	28.6%	33.3%	33.8%

Outline

Using Multiple Types of Partial Verifications

Suppose there are k types of partial verifications available:
$\left(V^{(1)}, r^{(1)}\right), \left(V^{(2)}, r^{(2)}\right), \ldots, \left(V^{(k)}, r^{(k)}\right)$

Which verification is the optimal one to use?

Using Multiple Types of Partial Verifications

Suppose there are k types of partial verifications available:
$\left(V^{(1)}, r^{(1)}\right)$, $\left(V^{(2)}, r^{(2)}\right)$, $\ldots$, $\left(V^{(k)}, r^{(k)}\right)$

Which verification is the optimal one to use?

Proposition

The execution overhead is minimized when using the partial verification with the maximum accuracy-to-cost ratio, i.e.,

$$\phi_{\max} = \max_i \phi^{(i)} = \max_i \left(\frac{1-g^{(i)}}{1+g^{(i)}} \Big/ \frac{V^{(i)}}{V^* + C} \right)$$

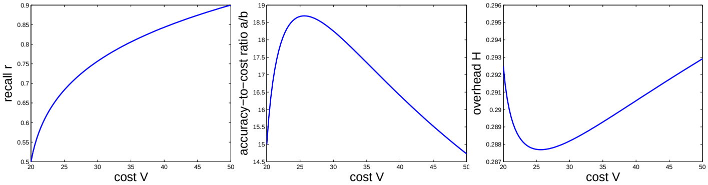
.

Proof. For a given partial verification type, say type i with $\phi^{(i)} > 2$.

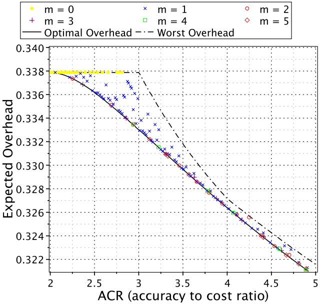
$$\textsc{Overhead}^* \approx \sqrt{2\lambda(V^* + C)} \left(\sqrt{1 - \frac{1}{\phi^{(i)}}} + \sqrt{\frac{1}{\phi^{(i)}}} \right)$$

The function $f = \sqrt{1-x} + \sqrt{x}$ is increasing in $[0, 1/2]$.

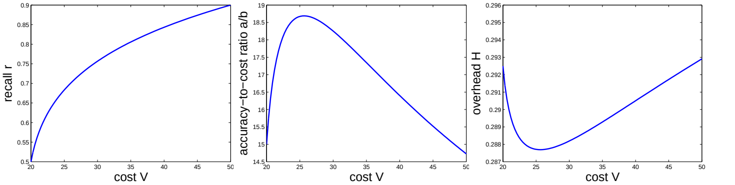
Using Multiple Types of Partial Verifications



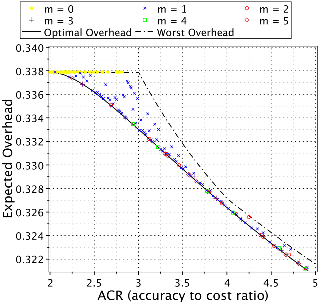
- Result is based on optimal rational solution (n^*).
- Overhead of integer solution may contain rounding error.
- Different partial verifications could share same ϕ, but lead to different n^* and $\textsc{Overhead}^*$.



Using Multiple Types of Partial Verifications

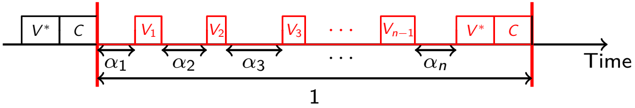


- Result is based on optimal rational solution (n^*).
- Overhead of integer solution may contain rounding error.
- Different partial verifications could share same ϕ, but lead to different n^* and $\textsc{Overhead}^*$.



What is the optimal integer solution?
Using multiple types simultaneously may help!

Using Multiple Types of Partial Verifications



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

(1) Go back to the $f_{\mathrm{re}} o_{\mathrm{ff}}$ analysis.

Proposition

Suppose a pattern P_{vc} that uses multiple types of partial verifications has n segments. Then the pattern is characterized by

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

where A is a symmetric matrix defined by $A_{ij} = \frac{1}{2}\left(1 + \prod_{k=i}^{j-1} g_k\right)$ for $i \leq j$.

Proof. Derive the expected re-execution fraction.

$$f_{\text{re}} = \sum_{i=1}^{n} \alpha_i \left(\sum_{j=1}^{i} \alpha_j + \sum_{j=i+1}^{n} \left(\prod_{k=i}^{j-1} g_k \right) \alpha_j \right)$$

The rest goes the same as before.

E.x., when $n = 4$,

$$A = \frac{1}{2} \begin{bmatrix} 2 & 1+g_1 & 1+g_1g_2 & 1+g_1g_2g_3 \\ 1+g_1 & 2 & 1+g_2 & 1+g_2g_3 \\ 1+g_1g_2 & 1+g_2 & 2 & 1+g_3 \\ 1+g_1g_2g_3 & 1+g_2g_3 & 1+g_3 & 2 \end{bmatrix}$$

Using Multiple Types of Partial Verifications

(2) Minimize $f_{\mathbf{re}}$.

Theorem

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

$$\alpha_i^* = \frac{1}{U_n} \times \frac{1 - g_{i-1} g_i}{(1 + g_{i-1})(1 + g_i)} \text{ for all } i = 1, \dots, n$$

where $g_0 = g_n = 0$ and

$$U_n = 1 + \sum_{i=1}^{n-1} \frac{1 - g_i}{1 + g_i}$$

In this case, the optimal value of f_{re} is

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

If all partial verifications are same ($g_i = g$), we retrieve previous results.

Using Multiple Types of Partial Verifications

The proof is similar as before, but the analysis is more involved.

- A is SPD.
- $A\boldsymbol{\alpha}^* = f^*_{\text{re}}\mathbf{c}$.

$$\boxed{\det\left(A^{(n)}\right) = \frac{U_n + 1}{2}\prod_{k=1}^{n-1}(1 - g_k^2)}$$

Using Multiple Types of Partial Verifications

The proof is similar as before, but the analysis is more involved.

- A is SPD.
- $A\boldsymbol{\alpha}^* = f_{\text{re}}^* \mathbf{c}$.

$$\boxed{\det\left(A^{(n)}\right) = \frac{U_n + 1}{2} \prod_{k=1}^{n-1}(1 - g_k^2)}$$

Corollary

For a given set of partial verifications in pattern P_{vc}, the minimum re-execution fraction f_{re}^* is independent of their ordering.

$$f_{\text{re}}^* = \frac{1}{2}\left(1 + \frac{1}{1 + \sum_{i=1}^{n-1} \frac{1-g_i}{1+g_i}}\right) \qquad o_{\text{ff}} = \sum_{i=1}^{n-1} V_i + V^* + C$$

Using Multiple Types of Partial Verifications

The proof is similar as before, but the analysis is more involved.

- A is SPD.
- $A\boldsymbol{\alpha}^* = f^*_{\text{re}}\mathbf{c}$.

$$\boxed{\det\left(A^{(n)}\right) = \frac{U_n + 1}{2}\prod_{k=1}^{n-1}(1 - g_k^2)}$$

Corollary

For a given set of partial verifications in pattern P_{vc}, the minimum re-execution fraction f^*_{re} is independent of their ordering.

$$f^*_{\text{re}} = \frac{1}{2}\left(1 + \frac{1}{1 + \sum_{i=1}^{n-1}\frac{1-g_i}{1+g_i}}\right) \qquad o_{\text{ff}} = \sum_{i=1}^{n-1} V_i + V^* + C$$

$$= \frac{1}{2}\left(1 + \frac{1}{1 + \sum_{j=1}^{k} m_j a^{(j)}}\right) \qquad = (V^* + C)\left(1 + \sum_{j=1}^{k} m_j b^{(j)}\right)$$

where $a^{(j)} = \frac{1-g^{(j)}}{1+g^{(j)}}$ and $b^{(j)} = \frac{V^{(j)}}{V^*+C}$ are the accuracy and relative cost of verification type j, and $\sum_{j=1}^{k} m_j = n - 1$.

Using Multiple Types of Partial Verifications

(3) Minimize $f_{\text{re}} o_{\text{ff}} = \frac{V^*+C}{2}\left(1+\frac{1}{1+\sum_{j=1}^{k} m_j a^{(j)}}\right)\left(1+\sum_{j=1}^{k} m_j b^{(j)}\right)$

Multi-type Partial Verification (MPV) Problem

Given k types of partial verifications and a bound K, is there a solution $\mathbf{m} = [m_1, m_2, \cdots, m_k]$ that satisfies

$$\left(1+\frac{1}{1+\sum_{j=1}^{k} m_j a^{(j)}}\right)\left(1+\sum_{j=1}^{k} m_j b^{(j)}\right) \leq K?$$

Proposition

The MPV problem is NP-complete, even when all the verification types share the same accuracy-to-cost ratio, i.e., $\frac{a^{(j)}}{b^{(j)}} = \phi$ *for all* $1 \leq j \leq k$.

Using Multiple Types of Partial Verifications

Proof. Reduction from Unbounded Subset Sum (USS) problem.

Unbounded Subset Sum (USS) Problem

Given a set $S = \{s_1, s_2, \ldots, s_k\}$ of k positive integers and a positive integer I, is there an integer solution $\mathbf{m} = [m_1, m_2, \ldots, m_j] \in \mathbb{N}_0^k$ such that $\sum_{j=1}^{k} m_j s_j = I$?

Using Multiple Types of Partial Verifications

Proof. Reduction from Unbounded Subset Sum (USS) problem.

Unbounded Subset Sum (USS) Problem

Given a set $S = \{s_1, s_2, \ldots, s_k\}$ of k positive integers and a positive integer I, is there an integer solution $\mathbf{m} = [m_1, m_2, \ldots, m_j] \in \mathbb{N}_0^k$ such that $\sum_{j=1}^{k} m_j s_j = I$?

Let a virtual verification $V^{(0)} = (a, b)$ with accuracy-to-cost ratio $\frac{a}{b} = \phi$ have integer solution $I = -\frac{1}{a} + \sqrt{\frac{1}{a}\left(\frac{1}{b} - \frac{1}{a}\right)}$ and bound $\left(\sqrt{\frac{1}{\phi}} + \sqrt{1 - \frac{1}{\phi}}\right)^2 = K$.
Construct k partial verifications from $V^{(0)}$ by setting $a^{(j)} = s_j a$ and $b^{(j)} = s_j b$.
Using any partial verification alone has no integer solution.

Using Multiple Types of Partial Verifications

Proof. Reduction from Unbounded Subset Sum (USS) problem.

Unbounded Subset Sum (USS) Problem

Given a set $S = \{s_1, s_2, \ldots, s_k\}$ of k positive integers and a positive integer I, is there an integer solution $\mathbf{m} = [m_1, m_2, \ldots, m_j] \in \mathbb{N}_0^k$ such that $\sum_{j=1}^{k} m_j s_j = I$?

Let a virtual verification $V^{(0)} = (a, b)$ with accuracy-to-cost ratio $\frac{a}{b} = \phi$ have integer solution $I = -\frac{1}{a} + \sqrt{\frac{1}{a}\left(\frac{1}{b} - \frac{1}{a}\right)}$ and bound $\left(\sqrt{\frac{1}{\phi}} + \sqrt{1 - \frac{1}{\phi}}\right)^2 = K$.
Construct k partial verifications from $V^{(0)}$ by setting $a^{(j)} = s_j a$ and $b^{(j)} = s_j b$.
Using any partial verification alone has no integer solution.
$(\Rightarrow)$ Suppose an integer solution exists for the USS problem:

$$
\begin{aligned}
& \left(1 + \frac{1}{1 + \sum_{j=1}^{k} m_j a^{(j)}}\right)\left(1 + \sum_{j=1}^{k} m_j b^{(j)}\right) \\
= \quad & \left(1 + \frac{1}{1 + a\sum_{j=1}^{k} m_j s_j}\right)\left(1 + b\sum_{j=1}^{k} m_j s_j\right) = \left(1 + \frac{1}{1 + aI}\right)(1 + bI) = K
\end{aligned}
$$

Using Multiple Types of Partial Verifications

Proof. Reduction from Unbounded Subset Sum (USS) problem.

Unbounded Subset Sum (USS) Problem

Given a set $S = \{s_1, s_2, \ldots, s_k\}$ of k positive integers and a positive integer I, is there an integer solution $\mathbf{m} = [m_1, m_2, \ldots, m_j] \in \mathbb{N}_0^k$ such that $\sum_{j=1}^{k} m_j s_j = I$?

Let a virtual verification $V^{(0)} = (a, b)$ with accuracy-to-cost ratio $\frac{a}{b} = \phi$ have integer solution $I = -\frac{1}{a} + \sqrt{\frac{1}{a}\left(\frac{1}{b} - \frac{1}{a}\right)}$ and bound $\left(\sqrt{\frac{1}{\phi}} + \sqrt{1 - \frac{1}{\phi}}\right)^2 = K$.
Construct k partial verifications from $V^{(0)}$ by setting $a^{(j)} = s_j a$ and $b^{(j)} = s_j b$.
Using any partial verification alone has no integer solution.
($\Rightarrow$) Suppose an integer solution exists for the USS problem:

$$\begin{aligned}
&\left(1 + \frac{1}{1 + \sum_{j=1}^{k} m_j a^{(j)}}\right)\left(1 + \sum_{j=1}^{k} m_j b^{(j)}\right) \\
= \quad &\left(1 + \frac{1}{1 + a\sum_{j=1}^{k} m_j s_j}\right)\left(1 + b\sum_{j=1}^{k} m_j s_j\right) = \left(1 + \frac{1}{1 + aI}\right)(1 + bI) = K
\end{aligned}$$

Need to prove ($\Leftarrow$) and need to choose ϕ small enough s.t. every $a^{(j)} < 1$.

Using Multiple Types of Partial Verifications

(3) Designing approximation algorithms.

- FPTAS (Fully Polynomial-Time Approximation Scheme): overhead within $(1+\epsilon)$ times the optimal with running time polynomial in the input size and $1/\epsilon$.

Using Multiple Types of Partial Verifications

(3) Designing approximation algorithms.

- FPTAS (Fully Polynomial-Time Approximation Scheme): overhead within $(1+\epsilon)$ times the optimal with running time polynomial in the input size and $1/\epsilon$.
- Greedy algorithm:
  - Employ the type of partial verification with the highest accuracy-to-cost ratio.
  - Compute the optimal solution using this type of verification only

$$\text{Optimal number: } m^* = -\frac{1}{a} + \sqrt{\frac{1}{a}\left(\frac{1}{b} - \frac{1}{a}\right)}$$

  - Round up the optimal rational solution $\lceil m^* \rceil$.

The Greedy algorithm has an approximation ratio $\sqrt{3/2} < 1.23$.

Using Multiple Types of Partial Verifications

Performance evaluation on realistic platform

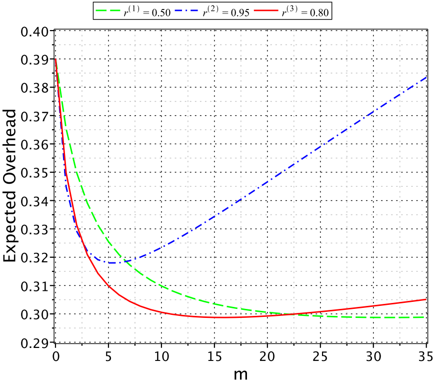
- 10^5 computing nodes with individual MTBF of 100 years
  $\Rightarrow$ platform MTBF $\mu \approx 8.7$ hours.
- Checkpoints size of 300GB with throughput of 0.5GB/s
  $\Rightarrow C = 600s$.
- Partial verifications (from Argonne National Laboratory, USA)

	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 the two	$V^{(3)} = 6s$	$r^{(3)} = [0.8, 0.99]$	$\phi^{(3)} = [133, 196]$
Perfect verification	$V^* = 600s$	$r^* = 1$	$\phi^* = 2$

Depending on the application or dataset, a verification's recall may vary, but its cost stays the same.

Using Multiple Types of Partial Verifications

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!

Using Multiple Types of Partial Verifications

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 scenario!

Outline

Coping with Both Fail-stop and Silent Errors

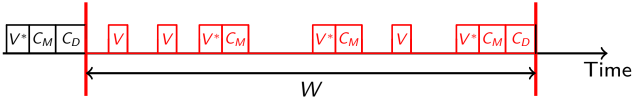
Fail-stop errors and silent errors coexist in large-scale platforms.
A resilience pattern needs to cope with both error sources simultaneously.

Coping with Both 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 with verifications

- 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).

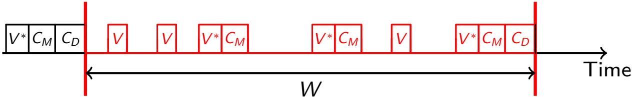


Coping with Both Fail-stop and Silent Errors

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 with verifications

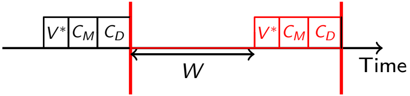
- Fail-stop errors (λ_f) are handled by disk checkpoints (C_D).
- Silent errors (λ_s) are handled by in-memory checkpoints (C_M) and verifications (guaranteed V^* or partial V).



Framework enforces following properties:

- *A guaranteed verification before each memory checkpoint.*
  $\Rightarrow$ *Checkpoints are always valid.*
- *A memory checkpoint before each disk checkpoint.*
  $\Rightarrow$ *Always recover from latest checkpoints.*

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



Proposition

The expected time to execute a base pattern P_D of work length W is

$$\begin{aligned}\mathbb{E}(W) &= W + V^* + C_M + C_D + \lambda_s W(W + V^* + R_M) \\ & \lambda_f W\left(\frac{W}{2} + R_M + R_D\right) + O(\lambda^2 W^3)\end{aligned}$$

Proof. Two error sources are independent.

$$\begin{aligned}\mathbb{E}(W) = p^f &\left(\frac{W}{2} + R_D + R_M + \mathbb{E}(W)\right) \\ &+ (1 - p^f)\big(W + V^* + p^s(R_M + \mathbb{E}(W)) \\ &\qquad + (1 - p^s)(C_M + C_D)\big) ,\end{aligned}$$

where $p^f = 1 - e^{\lambda_f W}$ and $p^s = 1 - e^{\lambda_s W}$.

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

Proposition

The optimal work length W^* of the base pattern P_D is

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

and the optimal expected overhead is

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

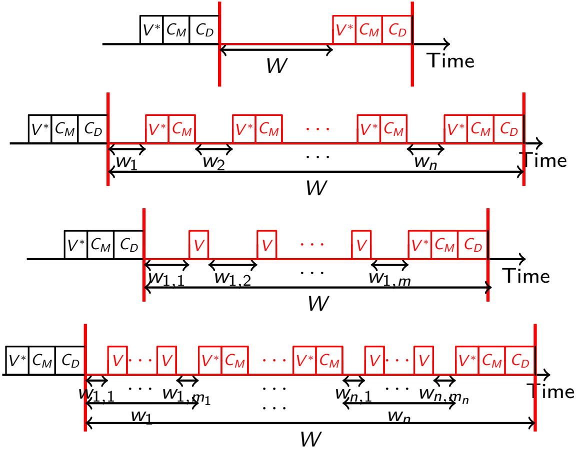
Proof. Derive the overhead from the expected execution time:

$$\textsc{Overhead} = \frac{\mathbb{E}(W)}{W} - 1 = \frac{V^* + C_M + C_D}{W} + \left(\lambda_s + \frac{\lambda_f}{2}\right) W + O(\lambda)$$

Similar analysis can be applied to more complex patterns.

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 an optimal pattern

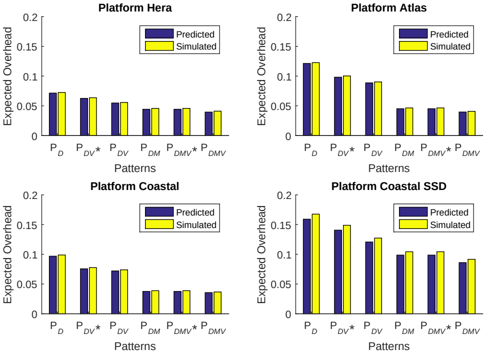
- W^*: optimal pattern period.
- n^*: optimal number of memory checkpoints in a pattern.
- m^*: optimal number of verifications between two memory checkpoints.

Pattern	W^*	n^*	m^*	OVERHEAD*
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}	$\sqrt{\frac{n^*(m^*-1)V+n^*(V^*+C_M)+C_D}{\frac{1}{2}\left(1+\frac{2-r}{(m^*-2)r+2}\right)\frac{\lambda_s}{n^*}+\frac{\lambda_f}{2}}}$	$\sqrt{\frac{\lambda_s}{\lambda_f}\cdot\frac{C_D}{V^*-\frac{2-r}{r}V+C_M}}$	$2-\frac{2}{r}$ $+\sqrt{\frac{2-r}{r}\left(\frac{V^*+C_M}{V}-\frac{2-r}{r}\right)}$	$\sqrt{2\lambda_f C_D}+\sqrt{2\lambda_s\left(V^*-\frac{2-r}{r}V+C_M\right)}$ $+\sqrt{2\lambda_s\frac{2-r}{r}V}$

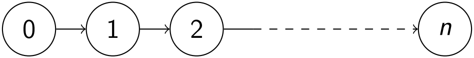
Performance Evaluation

- 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



Outline



Model

- A linear chain of n tasks $\{T_1, T_2, \ldots, T_n\}$, and each task T_i is characterized by a work w_i
- Two sources of errors
  - Fail-stop errors (λ_f)
  - Silent errors (λ_s)
- Resilience operations (only at the end of a task)
  - Disk checkpointing (C_D)
  - In-memory checkpointing (C_M)
  - Verification (V^* or V)

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

Dynamic Programming

Using only guaranteed verifications

- **Placing disk checkpoints**

$$E_{disk}(d_2) = \min_{0 \le d_1 < d_2} \{E_{disk}(d_1) + E_{mem}(d_1, d_2) + C_D\}$$

- **Placing memory checkpoints**

$$E_{mem}(d_1, m_2) = \min_{d_1 \le m_1 < m_2} \{E_{mem}(d_1, m_1) + E_{verif}(d_1, m_1, m_2) + C_M\}$$

- **Placing guaranteed verifications**

$$E_{verif}(d_1, m_1, v_2) = \min_{m_1 \le v_1 < v_2} \{E_{verif}(d_1, m_1, v_1) + E(d_1, m_1, v_1, v_2)\}$$

- **Computing expected execution time between two verifications**

$$\begin{aligned} E(d_1, m_1, v_1, v_2) = \\ & p^f \left(T^{\text{lost}} + R_D + E_{mem}(d_1, m_1) + E_{verif}(d_1, m_1, v_1) + E(d_1, m_1, v_1, v_2)\right) \\ & + (1 - p^f)\left(W_{v_1, v_2} + V^* + p^s\left(R_M + E_{verif}(d_1, m_1, v_1) + E(d_1, m_1, v_1, v_2)\right)\right) \end{aligned}$$

Using only guaranteed verifications

- Expected time lost due to a fail-stop error when executing W_{v_1,v_2}

$$\begin{aligned} T^{\text{lost}} &= \int_0^{\infty} x\mathbb{P}(X = x | X < W_{v_1,v_2}) dx \\ &= \frac{1}{\mathbb{P}(X < W_{v_1,v_2})} \int_0^{W_{v_1,v_2}} x\mathbb{P}(X = x) dx \\ &= \frac{1}{\lambda_f} - \frac{W_{v_1,v_2}}{e^{\lambda_f W_{v_1,v_2}} - 1} \qquad \text{(Integration by parts)} \end{aligned}$$

- Optimal expected makespan is given by $E_{disk}(n)$.
- Complexity is $O(n^4)$, dominated by table for $E_{verif}(d_1, m_1, v_2)$.

Dynamic Programming

Using only guaranteed verifications

- Expected time lost due to a fail-stop error when executing W_{v_1,v_2}

$$
\begin{aligned}
T^{\text{lost}} &= \int_0^\infty x\mathbb{P}(X = x | X < W_{v_1,v_2})dx \\
&= \frac{1}{\mathbb{P}(X < W_{v_1,v_2})} \int_0^{W_{v_1,v_2}} x\mathbb{P}(X = x)dx \\
&= \frac{1}{\lambda_f} - \frac{W_{v_1,v_2}}{e^{\lambda_f W_{v_1,v_2}} - 1} \qquad \text{(Integration by parts)}
\end{aligned}
$$

- Optimal expected makespan is given by $E_{disk}(n)$.
- Complexity is $O(n^4)$, dominated by table for $E_{verif}(d_1, m_1, v_2)$.

Using partial verifications

- Additional level for placing partial verifications.
- Due to imperfect recall, analysis is more involved.
- Complexity is $O(n^6)$.

Outline

Conclusion

Summary

- Comprehensive analysis of computing patterns to cope with silent errors.
- Two-level checkpointing scheme to deal with co-existence of fail-stop and silent errors.
- Resilient algorithms for linear chain of tasks.
- Performance evaluation based on parameters from real platforms.

Conclusion

Summary

- Comprehensive analysis of computing patterns to cope with silent errors.
- Two-level checkpointing scheme to deal with co-existence of fail-stop and silent errors.
- Resilient algorithms for linear chain of tasks.
- Performance evaluation based on parameters from real platforms.

Future directions

- What is the impact of partial verifications with imperfect precision (false positive)?

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

- How to cope with silent errors in computational workflows modeled as directed acyclic graphs (DAGs)?

Acknowledgement

Thanks to Hongyang Sun for the slides!

Joint work with

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

Presented materials are based on

- Efficient checkpoint/verification patterns for silent error detection. *ICL Research report RR-1403*, 2014
- Assessing general-purpose algorithms to cope with fail-stop and silent errors. *INRIA report RR-8599*, 2014.
- Assessing the impact of partial verifications against silent data corruptions. *INRIA report RR-8711*, 2015
- Which verification for soft error detection? *INRIA report RR-8741*, 2015
- Optimal resilience patterns to cope with fail-stop and silent errors. *INRIA report RR-8786*, 2015
- Two-level checkpointing and partial verifications for linear task graphs. *INRIA report RR-8794*, 2015