A nice little scheduling problem

Yves Robert

ENS Lyon & Institut Universitaire de France
University of Tennessee Knoxville

yves.robert@ens-lyon.fr

CCDSC 2014 – Dareizé, September 3, 2014

A nice little scheduling problem

Lame motivation

A nice little scheduling problem

Theorem 1
Theorem 2
Theorem 3
Theorem 4
Theorem 5
Theorem 6

Theorem 8
Theorem 9

A nice little scheduling problem

Conclusion: proving Theorem 7 would be nice

Algorithms for coping with silent errors

Yves Robert

ENS Lyon & Institut Universitaire de France
University of Tennessee Knoxville

yves.robert@ens-lyon.fr

CCDSC 2014 – Dareizé, September 3, 2014

Exascale platforms

- Hierarchical
  - 10^5 or 10^6 nodes
  - Each node equipped with 10^4 or 10^3 cores
- Failure-prone

MTBF – one node	10 years	120 years
MTBF – platform of 10^6 nodes	5mn	1h

More nodes $\Rightarrow$ Shorter MTBF (Mean Time Between Failures)

Definitions

- Instantaneous error detection $\Rightarrow$ fail-stop failures, e.g. resource crash
- Silent errors (data corruption) $\Rightarrow$ detection latency

Silent error detected only when the corrupt data is activated

- Includes some software faults, some hardware errors (soft errors in L1 cache), double bit flip
- Cannot always be corrected by ECC memory

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)

Should we be afraid? (courtesy Al Geist)

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?

Probability distributions for silent errors

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

Probability distributions for silent errors

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

Lesson learnt for fail-stop failures

(Not so) Secret data

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

$$T_{\text{opt}} = \sqrt{2\mu C} \quad \Rightarrow \quad \text{WASTE}_{\text{opt}} \approx \sqrt{\frac{2C}{\mu}}$$

Petascale:	$C = 20$ min	$\mu = 24$ hrs	$\Rightarrow \text{WASTE}_{\text{opt}} = 17\%$
Scale by 10:	$C = 20$ min	$\mu = 2.4$ hrs	$\Rightarrow \text{WASTE}_{\text{opt}} = 53\%$
Scale by 100:	$C = 20$ min	$\mu = 0.24$ hrs	$\Rightarrow \text{WASTE}_{\text{opt}} = 100\%$

Lesson learnt for fail-stop failures

(Also) Secret data

- Tsubame: 962 failures during last 18 months so MTBF = 13 hrs
- Blue Waters: 2-3 node failures per day
- Titan: a few failures per day
- Tianhe

Exascale ≠ Petascale ×1000
Need more reliable components
Need to checkpoint faster

Petascale:	$C = 20$ min	$\mu = 24$ hrs	$\Rightarrow$ WASTE$_{\text{opt}} = 17\%$
Scale by 10:	$C = 20$ min	$\mu = 2.4$ hrs	$\Rightarrow$ WASTE$_{\text{opt}} = 53\%$
Scale by 100:	$C = 20$ min	$\mu = 0.24$ hrs	$\Rightarrow$ WASTE$_{\text{opt}} = 100\%$

Lesson learnt for fail-stop failures

(Not so) Secret data

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

Silent errors:

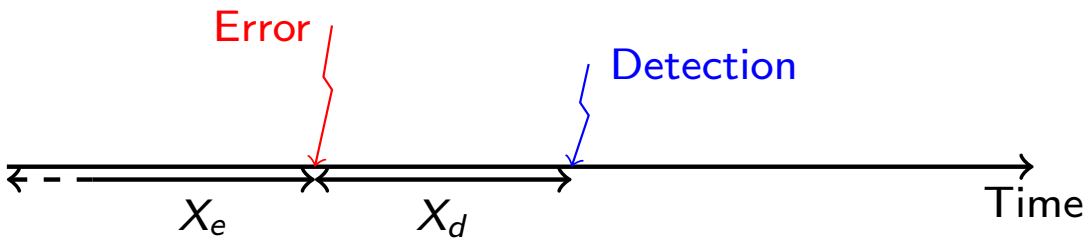
detection latency $\Rightarrow$ additional problems

Petascale:	$C = 20$ min	$\mu = 24$ hrs	$\Rightarrow \text{WASTE}_{\text{opt}} = 17\%$
Scale by 10:	$C = 20$ min	$\mu = 2.4$ hrs	$\Rightarrow \text{WASTE}_{\text{opt}} = 53\%$
Scale by 100:	$C = 20$ min	$\mu = 0.24$ hrs	$\Rightarrow \text{WASTE}_{\text{opt}} = 100\%$

Outline

Outline

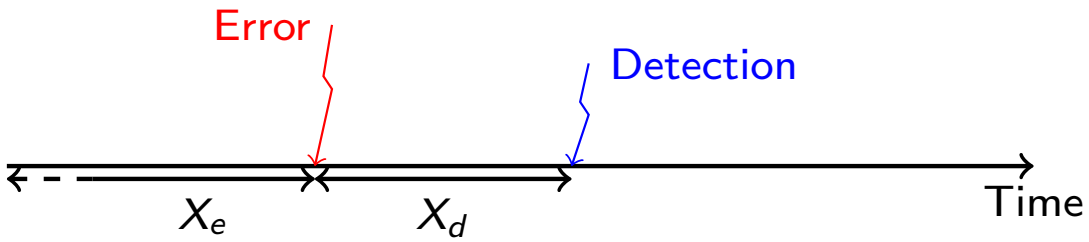
General-purpose approach



Error and detection latency

- Last checkpoint may have saved an already corrupted state
- Saving k checkpoints (Lu, Zheng and Chien):
  1. Critical failure when all live checkpoints are invalid
  2. Which checkpoint to roll back to?

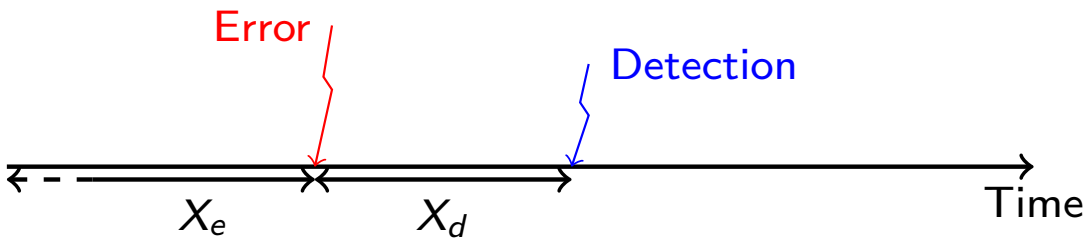
General-purpose approach



Error and detection latency

- Last checkpoint may have saved an already corrupted state
- Saving k checkpoints (Lu, Zheng and Chien):
  1. Critical failure when all live checkpoints are invalid
     Assume unlimited storage resources
  2. Which checkpoint to roll back to?
     Assume verification mechanism

Optimal period?



Error and detection latency

- X_e inter arrival time between errors; mean time μ_e
- X_d error detection time; mean time μ_d
- Assume X_d and X_e independent

Arbitrary distribution

$$\text{WASTE}_{\text{FF}} = \frac{C}{T}$$

$$\text{WASTE}_{\text{Fail}} = \frac{\frac{T}{2} + R + \mu_d}{\mu_e}$$

Only valid if $\frac{T}{2} + R + \mu_d \ll \mu_e$

Theorem

- *Best period is* $T_{opt} \approx \sqrt{2\mu_e C}$
- *Independent of* X_d

Limitation of the model

It is not clear how to detect when the error has occurred (hence to identify the last valid checkpoint) ☹ ☹ ☹

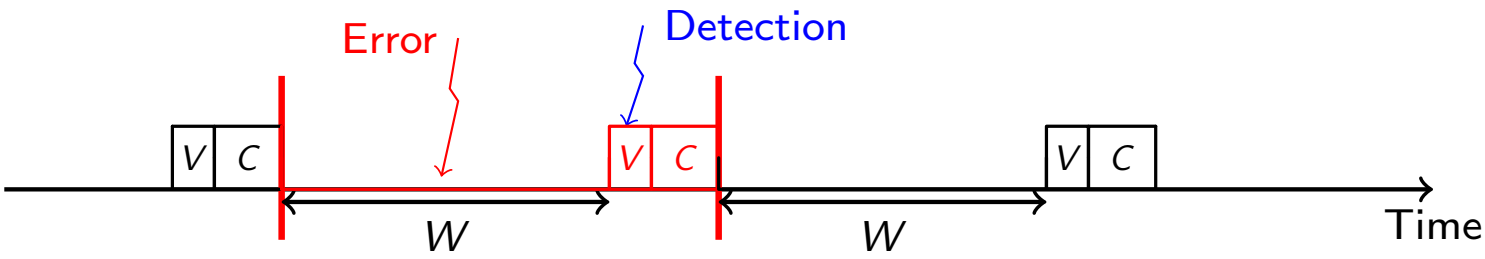
Need a verification mechanism to check the correctness of the checkpoints. This has an additional cost!

Outline

Coupling checkpointing and verification

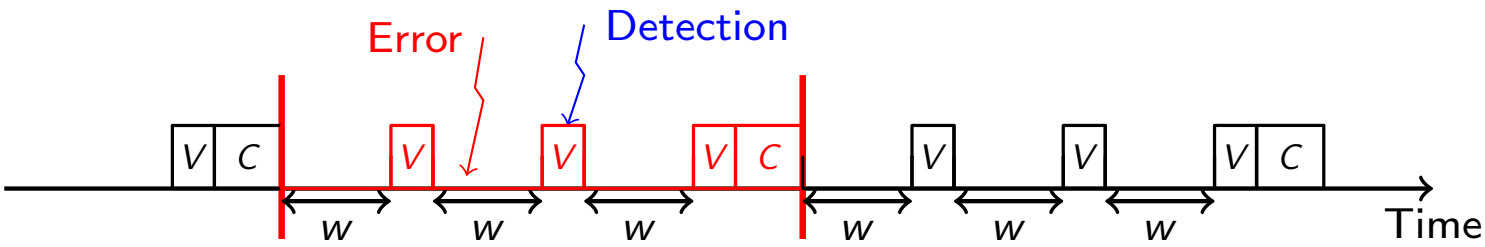
- Verification mechanism of cost V
- Silent errors detected only when verification is executed
- Approach agnostic of the nature of verification mechanism (checksum, error correcting code, coherence tests, etc)
- Fully general-purpose (application-specific information, if available, can always be used to decrease V)

Base pattern (and revisiting Young/Daly)



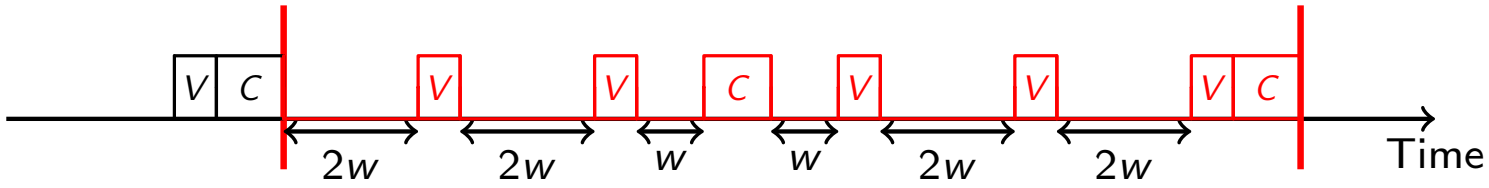
	Fail-stop (classical)	Silent errors
Pattern	$T = W + C$	$S = W + V + C$
WASTE_{FF}	$\frac{C}{T}$	$\frac{V+C}{S}$
$\text{WASTE}_{\text{fail}}$	$\frac{1}{\mu}(D + R + \frac{W}{2})$	$\frac{1}{\mu}(R + W + V)$
Optimal	$T_{\text{opt}} = \sqrt{2C\mu}$	$S_{\text{opt}} = \sqrt{(C+V)\mu}$
$\text{WASTE}_{\text{opt}}$	$\sqrt{\frac{2C}{\mu}}$	$2\sqrt{\frac{C+V}{\mu}}$

With $p = 1$ checkpoint and $q = 3$ verifications



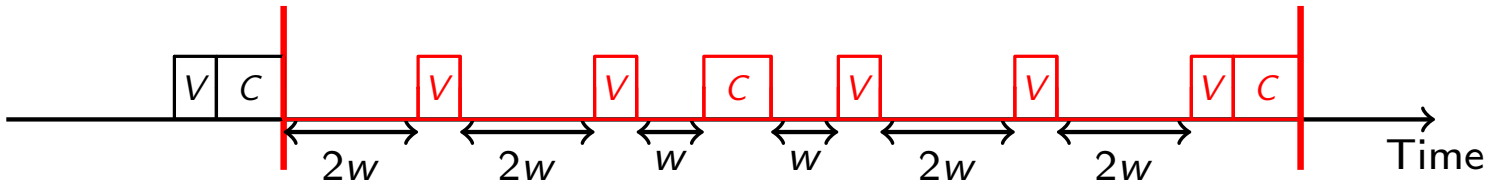
Base Pattern	$p = 1, q = 1$	$\text{WASTE}_{\text{opt}} = 2\sqrt{\frac{C+V}{\mu}}$
New Pattern	$p = 1, q = 3$	$\text{WASTE}_{\text{opt}} = 2\sqrt{\frac{4(C+3V)}{6\mu}}$

BalancedAlgorithm



- p checkpoints and q verifications, $p \leq q$
- $p = 2$, $q = 5$, $S = 2C + 5V + W$
- $W = 10w$, six chunks of size w or $2w$
- May store invalid checkpoint (error during third chunk)
- After successful verification in fourth chunk, preceding checkpoint is valid
- Keep only two checkpoints in memory and avoid any fatal failure

BalancedAlgorithm



① (proba $2w/W$) $T_{\text{lost}} = R + 2w + V$

② (proba $2w/W$) $T_{\text{lost}} = R + 4w + 2V$

③ (proba w/W) $T_{\text{lost}} = 2R + 6w + C + 4V$

④ (proba w/W) $T_{\text{lost}} = R + w + 2V$

⑤ (proba $2w/W$) $T_{\text{lost}} = R + 3w + 2V$

⑥ (proba $2w/W$) $T_{\text{lost}} = R + 5w + 3V$

$$\text{WASTE}_{\text{opt}} \approx 2\sqrt{\frac{7(2C+5V)}{20\mu}}$$

Outline

Literature

- ABFT: dense matrices / fail-stop, extended to sparse / silent. Limited to one error detection and/or correction in practice
- Asynchronous (chaotic) iterative methods (old work)
- Partial differential equations: use lower-order scheme as verification mechanism (detection only, Benson, Schmit and Schreiber)
- FT-GMRES: inner-outer iterations (Hoemmen and Heroux)
- PCG: orthogonalization check every k iterations, re-orthogonalization if problem detected (Sao and Vuduc)
- . . . Many others

On-line ABFT scheme for PCG

1 : 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 $x^{(0)}$
2 : checkpoint: A, M, and b
3 : for $i = 0, 1, \ldots$
4 : if ((i>0) and $(i\%d = 0)$)
5 : if ($\frac{p^{(i+1)^T} q^{(i)}}{||p^{(i+1)}||.||q^{(i)}||} > 10^{-10}$ or $\frac{||r^{(i+1)} + Ax^{(i+1)} - b||}{||b||.||A||} > 10^{-10}$)
6 : recover: A, M, b, i, ρ_i, $p^{(i)}$, $x^{(i)}$, and $r^{(i)}$.
7 : else if ($i\%(cd) = 0$)
8 : checkpoint: i, ρ_i, $p^{(i)}$, and $x^{(i)}$
9: endif
10: endif
11: $q^{(i)} = Ap^{(i)}$
12: $\alpha_i = \rho_i / p^{(i)^T} q^{(i)}$
13: $x^{(i+1)} = x^{(i)} + \alpha_i p^{(i)}$
14: $r^{(i+1)} = r^{(i)} - \alpha_i q^{(i)}$
15: solve $Mz^{(i+1)} = r^{(i+1)}$, where $M = M^T$
16: $\rho_{i+1} = r^{(i+1)^T} z^{(i+1)}$
17: $\beta_i = \rho_{i+1} / \rho_i$
10: $p^{(i+1)} = z^{(i+1)} + \beta_i p^{(i)}$
19: check convergence; continue if necessary
20: end

Zizhong Chen, PPoPP'13

- 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

Conclusion

- Soft errors difficult to cope with, even for divisible workloads
- Investigate graphs of computational tasks
- Combine checkpointing and application-specific techniques
- Multi-criteria optimization problem
  execution time/energy/reliability
  best resource usage (performance trade-offs)

Several challenging algorithmic/scheduling problems ☺

A little game?

Framework

- Compute something
- Energy cost $C_1 = 10$ and failure probability $f_1 = 0.2$
- Energy cost $C_2 = 8$ and failure probability $f_2 = 0.3$
- . . . (many other modes) . . .

Problem

- You win when you get twice the same result (no false positive)
- Find optimal strategy and compute expected cost

A mutlicore game?

Framework

- Now you have p cores for each trial
- Can freely run each core in a different mode (including idle)
- Each configuration has a cost C, and several probabilities:
  - p^{tom} = two or more successes (then you know you won)
  - p^{one} = exactly one success (but you don't know it)
  - of course $f = 1 - p^{tom} - p^{one}$

Problem

- You win when you get twice the same result (no false positive)
- Find optimal strategy and compute expected cost

Back to task graphs?

Framework

- You're given a (very big) task graph
- Each task produces files that you can save (checkpoint) or not
- Each task can choose from different execution speeds, with different error probabilites
- You can replicate some tasks, either for verification or for faster execution of successor tasks
- You may also be able to verify results by some application-specific mechanism

Problem

- Given energy budget or power cap, minimize execution time
- For each task, many things to be decided by schedule ☹

Back to task graphs?

Framework

- You're given
- Each task pro checkpoint) or not
- Each task car on speeds, with different erro
- You can repli fication or for faster executi
- You may also me application-sp

Problem

- Given energy execution time
- For each task schedule ☹

Back to task graphs?

Fr

ıot

Pr

The fox wants to save the polar bears . . .

Thanks

INRIA & ENS Lyon

- Anne Benoit
- Frédéric Vivien
- PhD students (Guillaume Aupy, Dounia Zaidouni)

Univ. Tennessee Knoxville

- George Bosilca
- Aurélien Bouteiller
- Jack Dongarra
- Thomas Hérault

Others

- Franck Cappello, Argonne National Lab.
- Henri Casanova, Univ. Hawai'i
- Saurabh K. Raina, Jaypee IIT, Noida, India
- Marc Snir, Argonne National Lab.