Algorithms for coping with silent errors

Yves Robert

ENS Lyon & Institut Universitaire de France
University of Tennessee Knoxville

yves.robert@ens-lyon.fr

http://graal.ens-lyon.fr/~yrobert/silent-errors.pdf

Scheduling for large-scale systems – July 4, 2014

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

About theory/practice and equations

- Will have some equations
- Will have a theorem or two
- Want to tell you a little story ☺

Outline

Outline

Exascale platforms

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

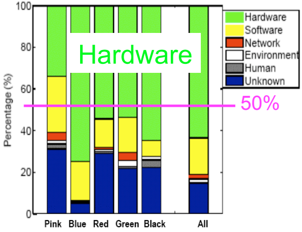
MTBF – one node	10 years	120 years
MTBF – platform of 10^6 nodes	5mn	1h

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

Error sources (courtesy Franck Cappello)

Sources of failures

- Analysis of error and failure logs
- In 2005 (Ph. D. of CHARNG-DA LU) : “Software halts account for the most number of outages (59-84 percent), and take the shortest time to repair (0.6-1.5 hours). Hardware problems, albeit rarer, need 6.3-100.7 hours to solve.”
- In 2007 (Garth Gibson, ICPP Keynote):



Relative frequency of root cause by system type.

- In 2008 (Oliner and J. Stearley, DSN Conf.):

Type	Raw Count	Raw %	Filtered Count	Filtered %
Hardware	174,586,516	98.04	1,999	18.78
Software	144,899	0.08	6,814	64.01
Indeterminate	3,350,044	1.88	1,832	17.21

Software errors: Applications, OS bug (kernel panic), communication libs, File system error and other.

Hardware errors, Disks, processors, memory, network

Conclusion: Both Hardware and Software failures have to be considered

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

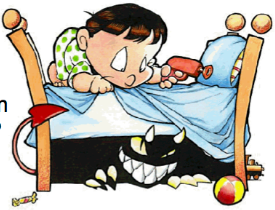
Hard errors – permanent component failure either HW or SW (hung or crash)

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

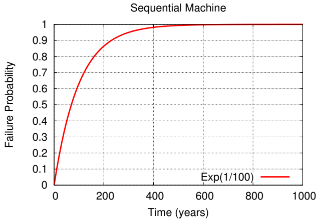
Silent errors – undetected errors either hard or soft, due to lack of detectors for a component or inability to detect (transient effect too short). Real danger is that answer may be incorrect but the user wouldn't know.

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?



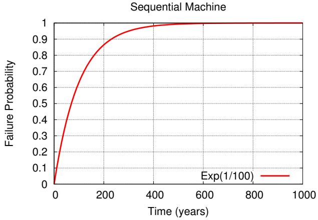
Failure distributions: (1) Exponential



Exp(λ): Exponential distribution law of parameter λ:

- Pdf: $f(t) = \lambda e^{-\lambda t} dt$ for $t \geq 0$
- Cdf: $F(t) = 1 - e^{-\lambda t}$
- Mean $= \frac{1}{\lambda}$

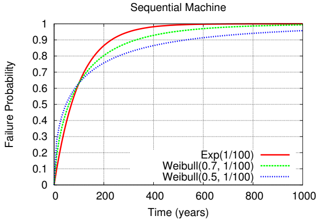
Failure distributions: (1) Exponential



X random variable for $Exp(\lambda)$ failure inter-arrival times:

- $\mathbb{P}(X \leq t) = 1 - e^{-\lambda t} dt$ (by definition)
- Memoryless property: $\mathbb{P}(X \geq t + s \,|\, X \geq s) = \mathbb{P}(X \geq t)$ at any instant, time to next failure does not depend upon time elapsed since last failure
- Mean Time Between Failures (MTBF) $\mu = \mathbb{E}(X) = \frac{1}{\lambda}$

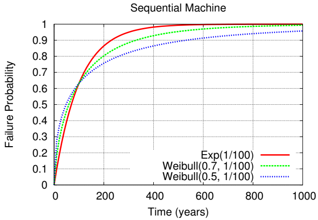
Failure distributions: (2) Weibull



Weibull(k,λ): Weibull distribution law of shape parameter k and scale parameter λ:

- Pdf: $f(t) = k\lambda(t\lambda)^{k-1}e^{-(\lambda t)^k}dt$ for $t \geq 0$
- Cdf: $F(t) = 1 - e^{-(\lambda t)^k}$
- Mean $= \frac{1}{\lambda}\Gamma(1+\frac{1}{k})$

Failure distributions: (2) Weibull



X random variable for $Weibull(k, \lambda)$ failure inter-arrival times:

- If $k < 1$: failure rate decreases with time
  "infant mortality": defective items fail early
- If $k = 1$: $Weibull(1, \lambda) = Exp(\lambda)$ constant failure time

Failure distributions: with several processors

- Processor (or node): any entity subject to failures
  $\Rightarrow$ approach agnostic to granularity

- If the MTBF is μ_{ind} with one processor,
  what is its value μ_p with p processors?

- Well, it depends ☹

Failure distributions: with several processors

- Processor (or node): any entity subject to failures
  $\Rightarrow$ approach agnostic to granularity

- If the MTBF is μ_{ind} with one processor,
  what is its value μ_p with p processors?

- Well, it depends ☹

With rejuvenation

- Rebooting all p processors after a failure
- Platform failure distribution
  $\Rightarrow$ minimum of p IID processor distributions
- With p distributions $Exp(\lambda)$:

$$\min_{1..p} \left(Exp(\lambda) \right) = Exp(p\lambda)$$

- With p distributions $Weibull(k, \lambda)$:

$$\min_{1..p} \left(Weibull(k, \lambda) \right) = Weibull(k, p^{1/k}\lambda)$$

Without rejuvenation (= real life)

- Rebooting only faulty processor
- Platform failure distribution
  $\Rightarrow$ superposition of p IID processor distributions

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

([illegible]so) Secret data

- Tsuba[illegible] 962 failures during last 18 months so[illegible] 13 hrs
- Blue Waters: 2-[illegible]de failures per day
- Titan: a few failures pe[illegible]
- Tianhe[illegible]

Exascale $\neq$ Petascale $\times 1000$
Need more reliable components
Need to checkpoint faster

Petascale[illegible]	$C = 20$ min	$\mu = 24$ hrs	$\Rightarrow$ W[illegible]TE$_{opt} = 17\%$
Scale [illegible] 10:	$C = 20$ min	$\mu = 2.4$ hrs	$\Rightarrow$ WAS[illegible]$_{opt} = 53\%$
Sca[illegible] by 100:	$C = 20$ min	$\mu = 0.24$ hrs	$\Rightarrow$ WASTE[illegible] $= 100\%$

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	$\Rightarrow \text{WASTE}_{\text{opt}} = 100\%$

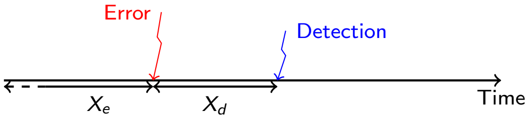
Application-specific methods

- 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

Outline

1. Introduction
2. Checkpointing for silent errors
3. Checkpointing and verification

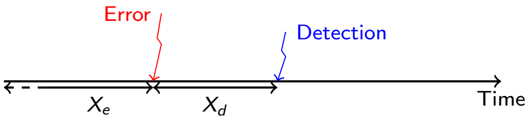
General-purpose approach



Error and detection latency

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

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

Exponential distribution

Theorem

- *At the end of the day,*

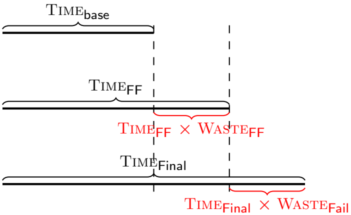
$$\mathbb{E}(T(w)) = e^{\lambda_e R} \left(\mu_e + \mu_d\right) \left(e^{\lambda_e (w+C)} - 1\right)$$

- *Optimal period independent of* μ_d
- *Good approximation is* $T = \sqrt{2\mu_e C}$ *(Young's formula)*

Arbitrary distribution X_e of mean μ_e

Waste: fraction of time not spent for useful computations

- $\textsc{Time}_{\text{base}}$: application base time
- $\textsc{Time}_{\text{FF}}$: with periodic checkpoints but failure-free
- $\textsc{Time}_{\text{Final}}$: expectation of time with failures



$$(1 - \textsc{Waste}_{\text{FF}})\textsc{Time}_{\text{FF}} = \textsc{Time}_{\text{base}}$$

$$(1 - \textsc{Waste}_{\text{Fail}})\textsc{Time}_{\text{Final}} = \textsc{Time}_{\text{FF}}$$

$$\textsc{Waste} = \frac{\textsc{Time}_{\text{Final}} - \textsc{Time}_{\text{base}}}{\textsc{Time}_{\text{Final}}}$$

$$\textsc{Waste} = 1 - (1 - \textsc{Waste}_{\text{FF}})(1 - \textsc{Waste}_{\text{Fail}})$$

Back to our model

We can show that

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

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

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 this model

Analytical optimal solutions, valid for arbitrary distributions, without any knowledge on X_d except its mean

However, if X_d can be arbitrary large:

- Do not know how far to roll back in time
- Need to store all checkpoints taken during execution

The case with limited resources

Assume that we can only save the last k checkpoints

Definition (Critical failure)

Error detected when all checkpoints contain corrupted data.
Happens with probability $\mathbb{P}_{\text{risk}}$ during whole execution.

$\mathbb{P}_{\text{risk}}$ decreases when T increases (when X_d is fixed).
Hence, $\mathbb{P}_{\text{risk}} \leq \varepsilon$ leads to a lower bound T_{min} on T

We have derived an analytical form for $\mathbb{P}_{\text{risk}}$ when X_d follows an Exponential law. We use it as a good(?) approximation for arbitrary laws

Limitation of the model

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

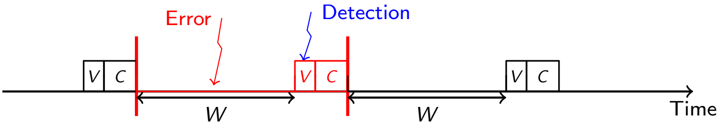
Need a verification mechanism to check the correctness of the checkpoints. This has an additional cost!

Outline

Coupling checkpointing and verification

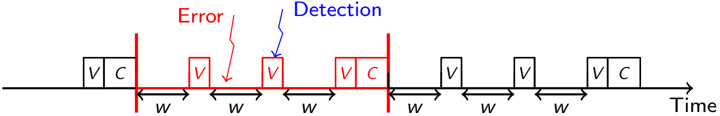
- Verification mechanism of cost V
- Silent errors detected only when verification is executed
- Approach agnostic of the nature of verification mechanism (checksum, error correcting code, coherence tests, etc)
- Fully general-purpose
  (application-specific information, if available, can always be used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

Base pattern (and revisiting Young/Daly)



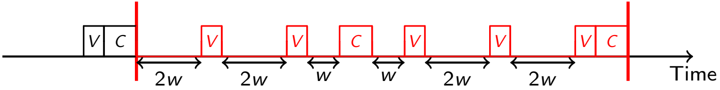
	Fail-stop (classical)	Silent errors
Pattern	$T = W + C$	$S = W + V + C$
WASTE_{FF}	$\frac{C}{T}$	$\frac{V+C}{S}$
$\text{WASTE}_{\text{fail}}$	$\frac{1}{\mu}(D + R + \frac{W}{2})$	$\frac{1}{\mu}(R + W + V)$
Optimal	$T_{\text{opt}} = \sqrt{2C\mu}$	$S_{\text{opt}} = \sqrt{(C+V)\mu}$
$\text{WASTE}_{\text{opt}}$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

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



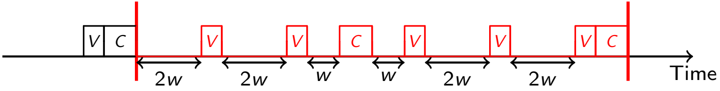
Base Pattern	$p = 1, q = 1$	$\text{WASTE}_{\text{opt}} = 2\sqrt{\frac{C+V}{\mu}}$
New Pattern	$p = 1, q = 3$	$\text{WASTE}_{\text{opt}} = 2\sqrt{\frac{4(C+3V)}{6\mu}}$

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

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

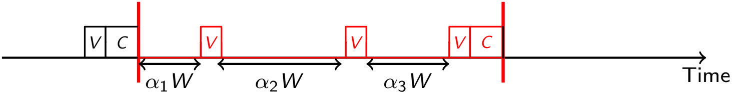
Analysis

- $S = pC + qV + pqw \ll \mu$
- $\textsc{Waste}_{\text{FF}} = \frac{o_{\text{ff}}}{S}$, where $o_{\text{ff}} = pC + qV$
- $\textsc{Waste}_{\text{Fail}} = \frac{T_{\text{lost}}}{\mu}$, where $T_{\text{lost}} = f_{\text{re}}S + \beta$
  - f_{re}: *fraction* of work that is *re-executed*
  - β: constant, linear combination of C, V and R
  - $f_{\text{re}} = \frac{7}{20}$ when $p = 2, q = 5$

$$S_{\text{opt}} = \sqrt{\frac{o_{\text{ff}}}{f_{\text{re}}}} \times \sqrt{\mu} + o(\sqrt{\mu})$$

$$\textsc{Waste}_{\text{opt}} = 2\sqrt{o_{\text{ff}} f_{\text{re}}}\sqrt{\frac{1}{\mu}} + o(\sqrt{\frac{1}{\mu}})$$

Computing f_{re} when $p = 1$

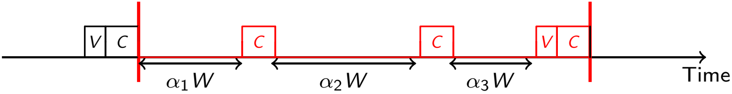


Theorem

The minimal value of $f_{re}(1, q)$ is obtained for same-size chunks

- $f_{re}(1, q) = \sum_{i=1}^{q} \left(\alpha_i \sum_{j=1}^{i} \alpha_j \right)$
- Minimal when $\alpha_i = 1/q$
- In that case, $f_{re}(1, q) = \frac{q+1}{2q}$

Computing f_{re} when $p \geq 1$



Theorem

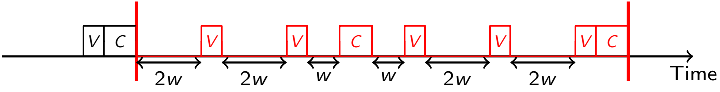
$f_{re}(p,q) \geq \frac{p+q}{2pq}$, *bound is matched by* BALANCEDALGORITHM.

- Assess gain due to the $p-1$ intermediate checkpoints
- $f_{\text{re}}^{(1)} - f_{\text{re}}^{(p)} = \sum_{i=1}^{p} \left(\alpha_i \sum_{j=1}^{i-1} \alpha_j \right)$
- Maximal when $\alpha_i = 1/p$ for all i
- In that case, $f_{\text{re}}^{(1)} - f_{\text{re}}^{(p)} = (p-1)/p^2$
- Now best with equipartition of verifications too
- In that case, $f_{\text{re}}^{(1)} = \frac{q+1}{2q}$ and $f_{\text{re}}^{(p)} = \frac{q+1}{2q} - \frac{p-1}{2p} = \frac{q+p}{2pq}$

Choosing optimal pattern

- Let $V = \gamma C$, where $0 < \gamma \leq 1$
- $o_{ff} f_{re} = \frac{p+q}{2pq}(pC + qV) = C \times \frac{p+q}{2}\left(\frac{1}{q} + \frac{\gamma}{p}\right)$
- Given γ, minimize $\frac{p+q}{2}\left(\frac{1}{q} + \frac{\gamma}{p}\right)$ with $1 \leq p \leq q$, and p, q taking integer values
- Let $p = \lambda \times q$. Then $\lambda_{opt} = \sqrt{\gamma} = \sqrt{\frac{V}{C}}$

Summary



- BALANCEDALGORITHM optimal when $C, R, V \ll \mu$
- Keep only 2 checkpoints in memory/storage
- Closed-form formula for $\text{WASTE}_{\text{opt}}$
- Given C and V, choose optimal pattern
- Gain of up to 20% over base pattern

Conclusion

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

Several challenging algorithmic/scheduling problems ☺