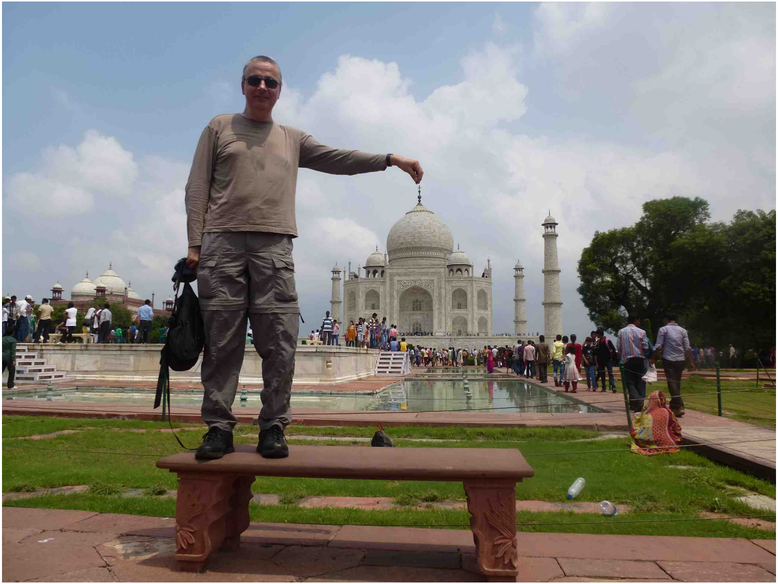
On the Combination of Silent Error Detection and Checkpointing

Guillaume Aupy, Anne Benoit, Thomas Hérault,
Yves Robert, Frédéric Vivien & Dounia Zaidouni

ICL Friday Lunch – September 6, 2013

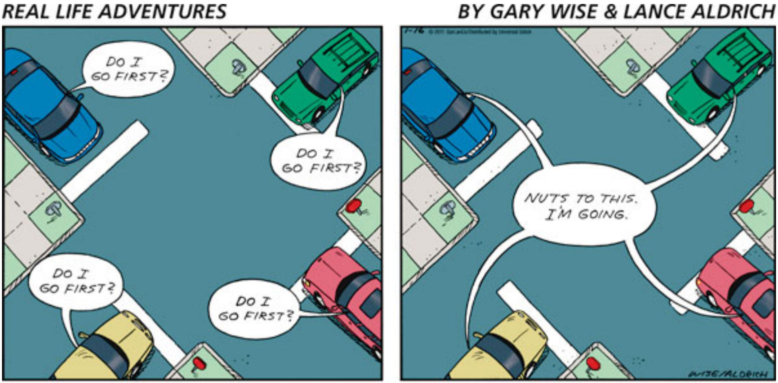
What about the retreat?

What about the retreat?



What about the retreat?

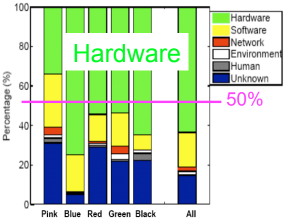
What about the retreat?



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

A few definitions

- Many types of faults: software error, hardware malfunction, memory corruption
- Many possible behaviors: silent, transient, unrecoverable
- Restrict to silent errors
- This includes some software faults, some hardware errors (soft errors in L1 cache), double bit flip
- Silent error detected when corrupt data is activated

A few definitions

- Many types of faults: software error, hardware malfunction, memory corruption
- Many possible behaviors: silent, transient, unrecoverable
- Restrict to silent errors
- This includes some software faults, some hardware errors (soft errors in L1 cache), double bit flip
- Silent error detected when corrupt data is activated
- *Silent errors are the black swans of errors* (Marc Snir)

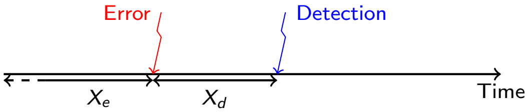


Figure : Error and detection latency.

- 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

Notations

- C checkpointing time
- R recovery time
- W total work
- w some piece of work

For one chunk

When X_e follows an Exponential law of parameter $\lambda_e = \frac{1}{\mu_e}$, in order to execute a total work of $w + C$, we need:

For one chunk

When X_e follows an Exponential law of parameter $\lambda_e = \frac{1}{\mu_e}$, in order to execute a total work of $w + C$, we need:

- Probability of execution without error

$$\mathbb{E}(T(w)) = e^{-\lambda_e(w+C)}(w + C)$$

$$+ \left(1 - e^{-\lambda_e(w+C)}\right)\left(\mathbb{E}(T_{lost}) + \mathbb{E}(X_d) + \mathbb{E}(T_{rec}) + \mathbb{E}(T(w))\right)$$

For one chunk

When X_e follows an Exponential law of parameter $\lambda_e = \frac{1}{\mu_e}$, in order to execute a total work of $w + C$, we need:

- Probability of execution without error

$$\mathbb{E}(T(w)) = e^{-\lambda_e(w+C)}(w + C)$$
$$+ \left(1 - e^{-\lambda_e(w+C)}\right)\left(\mathbb{E}(T_{lost}) + \mathbb{E}(X_d) + \mathbb{E}(T_{rec}) + \mathbb{E}(T(w))\right)$$

- Probability of error during $w + C$

For one chunk

When X_e follows an Exponential law of parameter $\lambda_e = \frac{1}{\mu_e}$, in order to execute a total work of $w + C$, we need:

- Probability of execution without error

$$\mathbb{E}(T(w)) = e^{-\lambda_e(w+C)}(w+C)$$
$$+ \left(1 - e^{-\lambda_e(w+C)}\right)\left(\mathbb{E}(T_{lost}) + \mathbb{E}(X_d) + \mathbb{E}(T_{rec}) + \mathbb{E}(T(w))\right)$$

- Probability of error during $w + C$
- Execution time with an error

Let us focus on the time lost due to an error:

$$\mathbb{E}(T_{lost}) + \mathbb{E}(X_d) + \mathbb{E}(T_{rec})$$

Let us focus on the time lost due to an error:

$\mathbb{E}(T_{lost}) + \mathbb{E}(X_d) + \mathbb{E}(T_{rec})$

This is the time elapsed between the completion of the last checkpoint and the error

$$\begin{aligned}\mathbb{E}(T_{lost}) &= \int_0^{\infty} x\mathbb{P}(X = x | X < w + C)dx \\ &= \frac{1}{\mathbb{P}(X < w + C)} \int_0^{w+C} x\lambda_e e^{-\lambda_e x} dx \\ &= \frac{1}{\lambda_e} - \frac{w + C}{e^{\lambda_e(w+C)} - 1}\end{aligned}$$

Let us focus on the time lost due to an error:

$$\mathbb{E}(T_{lost}) + \mathbb{E}(X_d) + \mathbb{E}(T_{rec})$$

This is the time needed for error detection, $\mathbb{E}(X_d) = \mu_d$

Let us focus on the time lost due to an error:

$$\mathbb{E}(T_{lost}) + \mathbb{E}(X_d) + \mathbb{E}(T_{rec})$$

This is the time to recover from the error (there can be a fault durnig recovery):

$$\mathbb{E}(T_{rec}) = e^{-\lambda_e R} R + (1 - e^{-\lambda_e R})(\mathbb{E}(R_{lost}) + \mathbb{E}(X_d) + \mathbb{E}(T_{rec}))$$

Let us focus on the time lost due to an error:

$\mathbb{E}(T_{lost}) + \mathbb{E}(X_d) + \mathbb{E}(T_{rec})$

This is the time to recover from the error (there can be a fault durnig recovery):

$$\mathbb{E}(T_{rec}) = e^{-\lambda_e R} R \\ + (1 - e^{-\lambda_e R})(\mathbb{E}(R_{lost}) + \mathbb{E}(X_d) + \mathbb{E}(T_{rec}))$$

Similarly to $\mathbb{E}(T_{lost})$, we have: $\mathbb{E}(R_{lost}) = \frac{1}{\lambda_e} - \frac{R}{e^{\lambda_e R} - 1}$.

Let us focus on the time lost due to an error:
$\mathbb{E}(T_{lost}) + \mathbb{E}(X_d) + \mathbb{E}(T_{rec})$

This is the time to recover from the error (there can be a fault durnig recovery):

$$\mathbb{E}(T_{rec}) = e^{-\lambda_e R} R + (1 - e^{-\lambda_e R})(\mathbb{E}(R_{lost}) + \mathbb{E}(X_d) + \mathbb{E}(T_{rec}))$$

Similarly to $\mathbb{E}(T_{lost})$, we have: $\mathbb{E}(R_{lost}) = \frac{1}{\lambda_e} - \frac{R}{e^{\lambda_e R} - 1}$.

So finally, $\mathbb{E}(T_{rec}) = (e^{\lambda_e R} - 1)(\mu_e + \mu_d)$

At the end of the day,

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

This is the exact solution!

For multiple chunks

Using n chunks of size w_i (with $\sum_{i=1}^{n} w_i = W$), we have:

$$\mathbb{E}(T(W)) = K \sum_{i=1}^{n} (e^{\lambda_e (w_i + C)} - 1)$$

with K constant.

Independent of μ_d!

Minimum when all the w_i's are equal to $w = W/n$.

For multiple chunks

Using n chunks of size w_i (with $\sum_{i=1}^{n} w_i = W$), we have:

$$\mathbb{E}(T(W)) = K \sum_{i=1}^{n} (e^{\lambda_e (w_i + C)} - 1)$$

with K constant.

Independent of μ_d!

Minimum when all the w_i's are equal to $w = W/n$.
Optimal n can be found by differentiation
A good approximation is $w = \sqrt{2\mu_e C}$ (Young's formula)

Arbitrary distributions

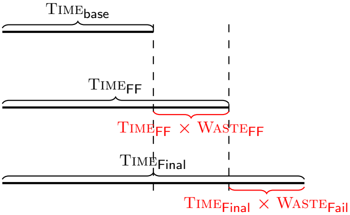
Extend results when X_e follows an arbitrary distribution of mean μ_e

Framework

Waste: fraction of time not spent for useful computations

Waste

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



$$(1 - \text{WASTE}_{\text{FF}})\text{TIME}_{\text{FF}} = \text{TIME}_{\text{base}}$$

$$(1 - \text{WASTE}_{\text{Fail}})\text{TIME}_{\text{Final}} = \text{TIME}_{\text{FF}}$$

$$\text{WASTE} = \frac{\text{TIME}_{\text{Final}} - \text{TIME}_{\text{base}}}{\text{TIME}_{\text{Final}}}$$

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

Back to our model

We can show that

$$\textsc{Waste}_{\mathsf{FF}} = \frac{C}{T}$$

$$\textsc{Waste}_{\mathsf{Fail}} = \frac{\frac{T}{2} + R + \mu_d}{\mu_e}$$

Back to our model

We can show that

$$\textsc{Waste}_{\mathsf{FF}} = \frac{C}{T}$$

$$\textsc{Waste}_{\mathsf{Fail}} = \frac{\frac{T}{2} + R + \mu_d}{\mu_e}$$

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

Back to our model

We can show that

$$\textsc{Waste}_{\textsf{FF}} = \frac{C}{T}$$

$$\textsc{Waste}_{\textsf{Fail}} = \frac{\frac{T}{2} + R + \mu_d}{\mu_e}$$

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

Then the waste is minimized for
$T_{\text{opt}} = \sqrt{2(\mu_e - (R + \mu_d))C} \approx \sqrt{2\mu_e C}$

Summary

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.

The case with limited resources

$\mathbb{P}_{\text{risk}}$ 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

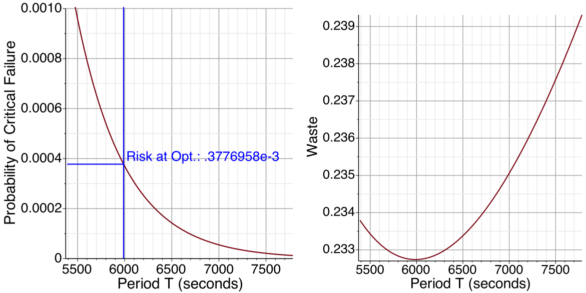


Figure : $k = 3$, $\lambda_e = \frac{10^5}{100y}$, $\lambda_d = 30\lambda_e$, $w = 10d$, $C = R = 600s$

$T_{\text{opt}} \approx 100min$, $\mathbb{P}_{\text{risk}}(T_{\text{opt}}) \approx 38 \cdot 10^{-5}$, for a waste of 23.45%

To reduce $\mathbb{P}_{\text{risk}}$ to 10^{-4}, a T_{min} of 8000 seconds is sufficient, increasing the waste by only 0.6%. In this case, the benefit of fixing the period to $\max(T_{\text{opt}}, T_{\text{min}})$ is obvious

More optimistic technologic scenario (smaller C and R):

T_{opt} is largely reduced (down to less than 35 minutes), but $\mathbb{P}_{\text{risk}}(T_{\text{opt}})$ climbs to 1/2, an unacceptable value.
To reduce $\mathbb{P}_{\text{risk}}$ to 10^{-4}, it becomes necessary to consider a T_{min} of 6650 seconds. The waste increases to 15%, significantly higher than the optimal one, which is below 10%

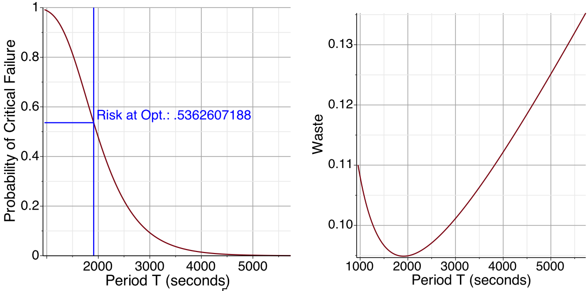


Figure : $k = 3$, $\lambda_e = \frac{10^5}{100y}$, $\lambda_d = 30\lambda_e$, $w = 10d$, $C = R = 60s$.

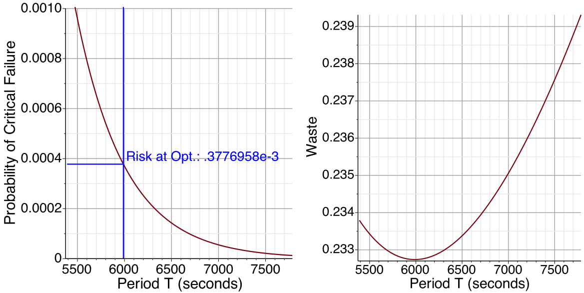


Figure : $k = 3$, $\lambda_e = \frac{10^5}{100y}$, $\lambda_d = 30\lambda_e$, $w = 10d$, $C = R = 600s$

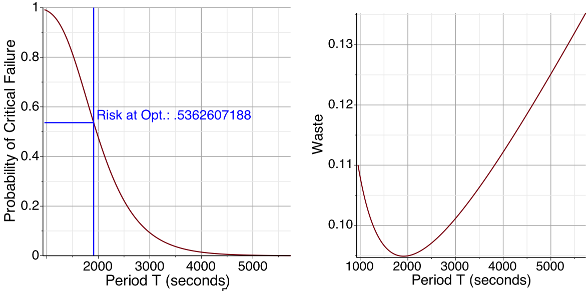


Figure : $k = 3$, $\lambda_e = \frac{10^5}{100y}$, $\lambda_d = 30\lambda_e$, $w = 10d$, $C = R = 60s$.

Limitation of the model

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

Need a verification mechanism to check the correctness of the checkpoints. This has a cost!
Possible solution: add verifications; use a periodic mechanism to verify that there were no silent errors in previous computations.

With detection

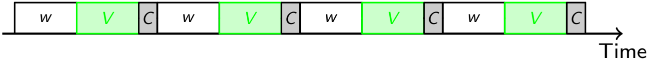
Assume there are no errors during checkpoints (less error sources when doing I/O)

Simple approach: Perform a verification before each checkpoint to eliminate risk of corrupted data.

Motivational Examples

$R = 0$:

$\text{WASTE}_{\text{FF}} = \frac{V+C}{w+V+C}$, $\text{WASTE}_{\text{Fail}} = \frac{w}{\mu_e}$

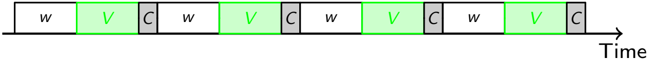


When V is large compared to w, WASTE_{FF} is large, can we improve that?

Motivational Examples

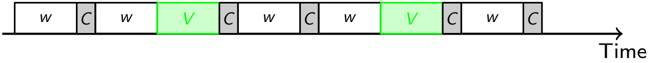
$R = 0$:

$\text{WASTE}_{\text{FF}} = \frac{V+C}{w+V+C}$, $\text{WASTE}_{\text{Fail}} = \frac{w}{\mu_e}$



When V is large compared to w, WASTE_{FF} is large, can we improve that?

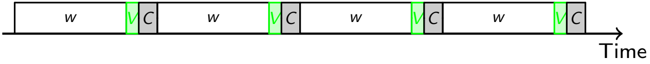
Is this better?



Motivational Examples

$R = 0$:

$\textsc{Waste}_{\mathsf{FF}} = \frac{V+C}{w+V+C}, \textsc{Waste}_{\mathsf{Fail}} = \frac{w}{\mu_e}$

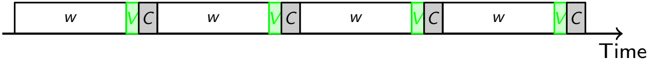


When V is small in front of w, $\textsc{Waste}_{\mathsf{Fail}}$ is large, can we improve that?

Motivational Examples

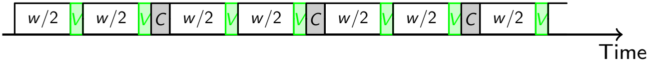
$R = 0$:

$\text{WASTE}_{\text{FF}} = \frac{V+C}{w+V+C}$, $\text{WASTE}_{\text{Fail}} = \frac{w}{\mu_e}$

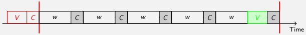


When V is small in front of w, $\text{WASTE}_{\text{Fail}}$ is large, can we improve that?

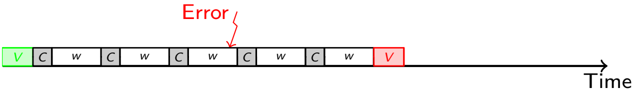
Is this better?



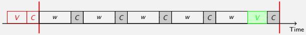
k checkpoints for 1 verification



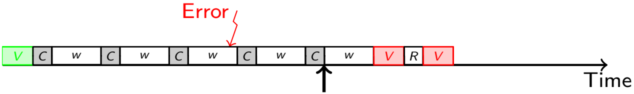
With multiple checkpoints, the problem is to find when the error occurred.



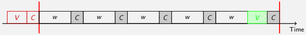
k checkpoints for 1 verification



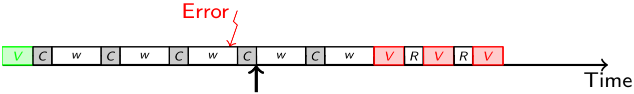
With multiple checkpoints, the problem is to find when the error occurred.



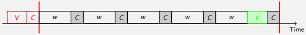
k checkpoints for 1 verification



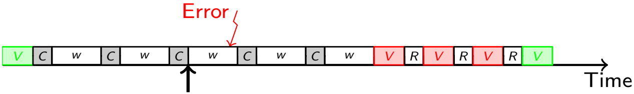
With multiple checkpoints, the problem is to find when the error occurred.



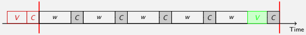
k checkpoints for 1 verification



With multiple checkpoints, the problem is to find when the error occurred.



k checkpoints for 1 verification

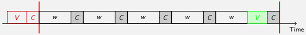


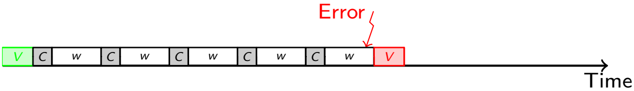
$$\text{WASTE}_{\mathsf{FF}} = \frac{kC + V}{k(w + C) + V}$$

$$\text{WASTE}_{\mathsf{Fail}} = \frac{\frac{1}{k}\sum_{i=1}^{k} T_{lost}(i)}{\mu_e}$$

where $T_{lost}(i)$ is the time lost if error occurred in i^{th} segment

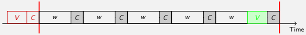
k checkpoints for 1 verification

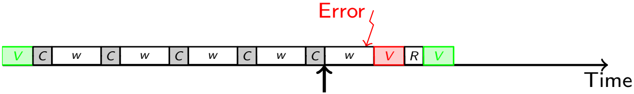




$$T_{lost}(k) = R + V + w + V$$

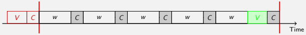
k checkpoints for 1 verification

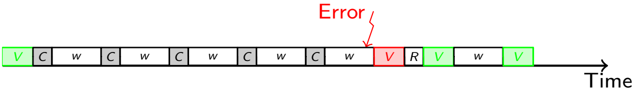




$$T_{lost}(k) = R + V + w + V$$

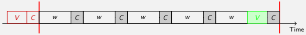
k checkpoints for 1 verification

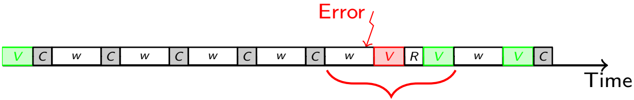




$$T_{lost}(k) = R + V + w + V$$

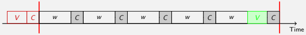
k checkpoints for 1 verification

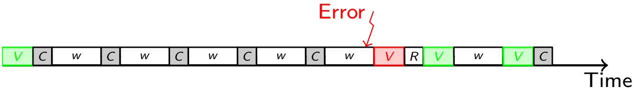




$$T_{lost}(k) = R + V + w + V$$

k checkpoints for 1 verification

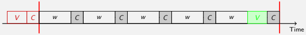


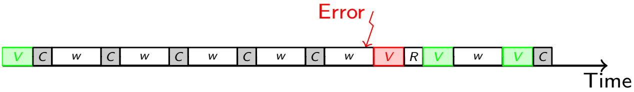


$$T_{lost}(k) = R + V + w + V$$

$$T_{lost}(i) = (k - i + 1)(R + V + w) + (k - i)C + V$$

k checkpoints for 1 verification



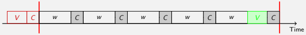


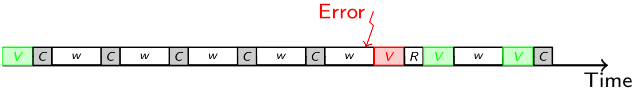
$$T_{lost}(k) = R + V + w + V$$

$$T_{lost}(i) = (k-i+1)(R+V+w) + (k-i)C + V$$

$$T_{lost}(1) = k(R+V+w) - V + (k-1)C + V$$

k checkpoints for 1 verification





$$T_{lost}(k) = R + V + w + V$$

$$T_{lost}(i) = (k - i + 1)(R + V + w) + (k - i)C + V$$

$$T_{lost}(1) = k(R + V + w) - V + (k - 1)C + V$$

And this leads us to optimal solution . . .

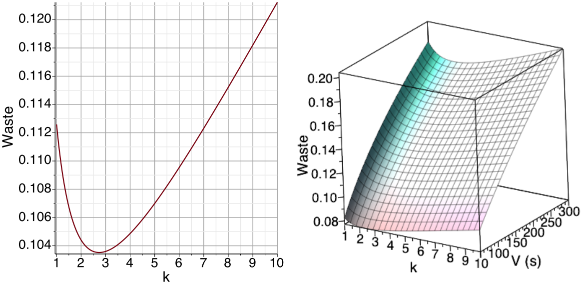


Figure : $V = 100s$, $C = R = 6s$, and $\mu = \frac{10y}{10^5}$.

$C = 6s \ll V$.

When $V = 100$ seconds, a verification is done only every $k = 3$ checkpoints optimally $\Rightarrow$ 10% improvement compared to $k = 1$.

$C = 60s$ is not negligible anymore in front of V ($V \approx 5C$). The waste is dominated by the cost of verification, and little improvement can be achieved by taking the optimal value for k.

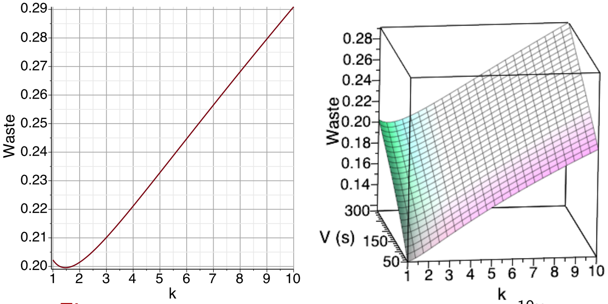


Figure : $V = 300s$, $C = R = 60s$, and $\mu = \frac{10y}{10^5}$

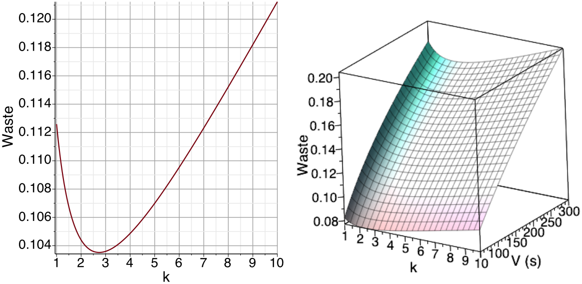


Figure : $V = 100s$, $C = R = 6s$, and $\mu = \frac{10y}{10^5}$.

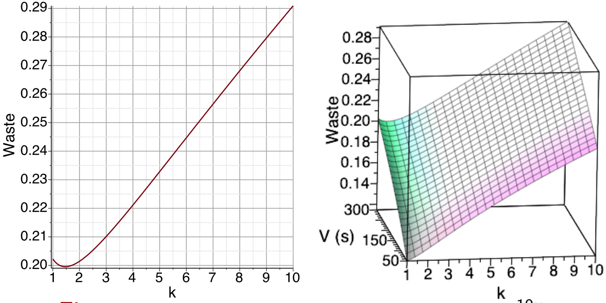
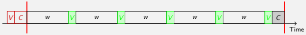


Figure : $V = 300s$, $C = R = 60s$, and $\mu = \frac{10y}{10^5}$.

k verifications for 1 checkpoint



Very similarly, we obtain:

$$\text{WASTE}_{\mathsf{FF}} = \frac{kV + C}{k(w+V)+C}$$

$$\text{WASTE}_{\mathsf{Fail}} = \frac{\frac{1}{k}\sum_{i=1}^{k} T_{lost}(i)}{\mu_e}$$

$$T_{lost}(i) = R + i(V+w)$$

where $T_{lost}(i)$ is the time lost if error occurred in i^{th} segment.

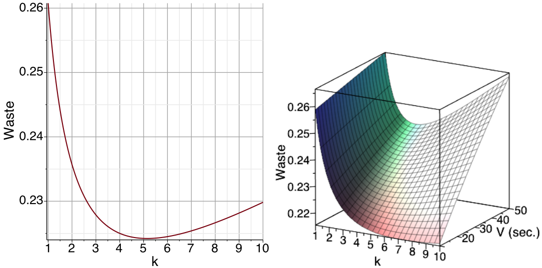


Figure : $V = 20s$, $C = R = 600s$, and $\mu = \frac{10y}{10^5}$.

$V = 20s \ll C$.

When $C = 600$ seconds, 5 verifications are done for every checkpoint optimally $\Rightarrow$ 14% improvement compared to $k = 1$.

$V = 2s \ll C$.

When $C = 60$ seconds, 5 verifications are done every checkpoint optimally $\Rightarrow$ 18% improvement compared to $k = 1$.

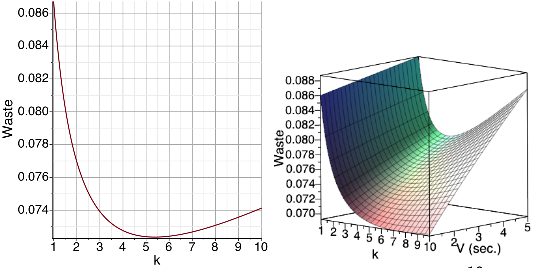


Figure : $V = 2s$, $C = R = 60s$, and $\mu = \frac{10y}{10^5}$.

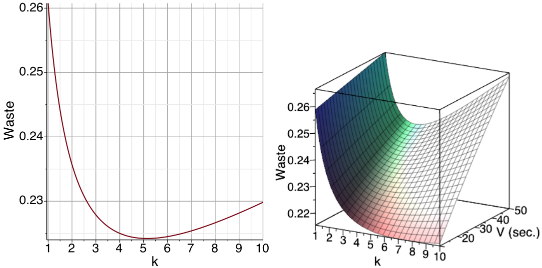


Figure : $V = 20s$, $C = R = 600s$, and $\mu = \frac{10y}{10^5}$.

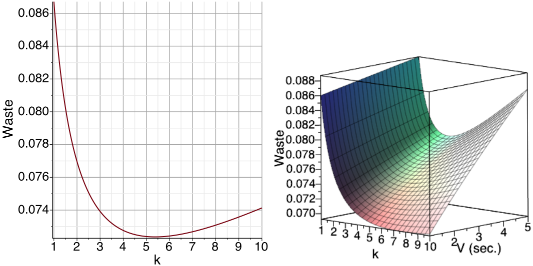


Figure : $V = 2s$, $C = R = 60s$, and $\mu = \frac{10y}{10^5}$.

Conclusion

- Study of optimal checkpointing strategy in presence of silent errors
- *Analytical* solution for the different probability distributions
- Study in presence of verification mechanisms

Future work

- *Without verification:* When we keep k checkpoints in memory, we do not have to keep the k last checkpoints: new strategies (Fibonacci, binary, ...)?
- *With verification:* We focused on an integer number of checkpoints per verification (or conversely): extensions?