Resilience

Guillaume Pallez
Inria

M2 CISD, Enseirb-Matmeca,
Automne 2019

Plan du cours d'aujourd'hui

Finishing your PhD on time

In a hypothetical future, in a galaxy far far away, you decided to do a PhD.

Finishing your PhD on time

In a hypothetical future, in a galaxy far far away, you decided to do a PhD. Even more hypothetical, you made it to the step where you are actually writing it!

Finishing your PhD on time

In a hypothetical future, in a galaxy far far away, you decided to do a PhD. Even more hypothetical, you made it to the step where you are actually writing it!

- Thanks to the lack of budget in research, your laptop is very old: it crashes often.
- You chose to write your thesis using a WYSIWYG software which takes approx 3 minutes to save (while freezing your laptop obviously).. Too many figures.

What do you do?

Finishing your PhD on time

In a hypothetical future, in a galaxy far far away, you decided to do a PhD. Even more hypothetical, you made it to the step where you are actually writing it!

- Thanks to the lack of budget in research, your laptop is very old: it crashes often.
- You chose to write your thesis using a WYSIWYG software which takes approx 3 minutes to save (while freezing your laptop obviously).. Too many figures.

What do you do?

Solution: Write your thesis in Latex, faster to save (but it's too late now)

In a hypothetical future, in a galaxy far far away, you decided to do a PhD. Even more hypothetical, you made it to the step where you are actually writing it!

- Thanks to the lack of budget in research, your laptop is very old: it crashes often.
- You chose to write your thesis using a WYSIWYG software which takes approx 3 minutes to save (while freezing your laptop obviously).. Too many figures.

What do you do?

- Solution 1: You choose to save your work every 3h.
  - Mid-afternoon of Day 3, your laptop crashes. You have lost 1.5h of work.

Finishing your PhD on time

In a hypothetical future, in a galaxy far far away, you decided to do a PhD. Even more hypothetical, you made it to the step where you are actually writing it!

- Thanks to the lack of budget in research, your laptop is very old: it crashes often.
- You chose to write your thesis using a WYSIWYG software which takes approx 3 minutes to save (while freezing your laptop obviously).. Too many figures.

What do you do?

- Solution 1: You choose to save your work every 3h.
  - Mid-afternoon of Day 3, your laptop crashes. You have lost 1.5h of work.
- Solution 2: You choose to save your work every half-hour.
  - No crash during the next three consecutive days.

Which solution is best?

Plan

Exascale platforms (courtesy Jack Dongarra)

Potential System Architecture with a cap of $200M and 20MW

Systems	2011 K computer	2019	Difference Today & 2019
System peak	**10.5 Pflop/s**	**1 Eflop/s**	**O(100)**
Power	**12.7 MW**	**~20 MW**	
System memory	1.6 PB	32 - 64 PB	O(10)
Node performance	128 GF	1,2 or 15TF	O(10) – O(100)
Node memory BW	64 GB/s	2 - 4TB/s	O(100)
Node concurrency	8	O(1k) or 10k	O(100) – O(1000)
Total Node Interconnect BW	20 GB/s	200-400GB/s	O(10)
System size (nodes)	88,124	O(100,000) or O(1M)	O(10) – O(100)
Total concurrency	705,024	O(billion)	O(1,000)
MTTI	days	O(1 day)	- O(10)

Exascale platforms (courtesy C. Engelmann & S. Scott)

Toward Exascale Computing (My Roadmap)

Based on proposed DOE roadmap with MTTI adjusted to scale linearly

Systems	2009	2011	2015	2018
System peak	2 Peta	20 Peta	100-200 Peta	1 Exa
System memory	0.3 PB	1.6 PB	5 PB	10 PB
Node performance	125 GF	200GF	200-400 GF	1-10TF
Node memory BW	25 GB/s	40 GB/s	100 GB/s	200-400 GB/s
Node concurrency	12	32	O(100)	O(1000)
Interconnect BW	1.5 GB/s	22 GB/s	25 GB/s	50 GB/s
System size (nodes)	18,700	100,000	500,000	O(million)
Total concurrency	225,000	3,200,000	O(50,000,000)	O(billion)
Storage	15 PB	30 PB	150 PB	300 PB
IO	0.2 TB/s	2 TB/s	10 TB/s	20 TB/s
MTTI	4 days	19 h 4 min	3 h 52 min	1 h 56 min
Power	6 MW	~10MW	~10 MW	~20 MW

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

- Failure-prone

MTBF – one node	1 year	10 years	120 years
MTBF – platform of 10^6 nodes	30sec	5mn	1h

More nodes ⇒ Shorter MTBF (Mean Time Between Failures)

- **Hierarchical**
  - 10^5 or 10^6 nodes
  - Each node equipped with ... or 10^3 cores

- **Failure-prone**

MTBF – one node	1 year	10 years	120 years
MTBF – platform of 10^6 nodes	30sec	5mn	1h

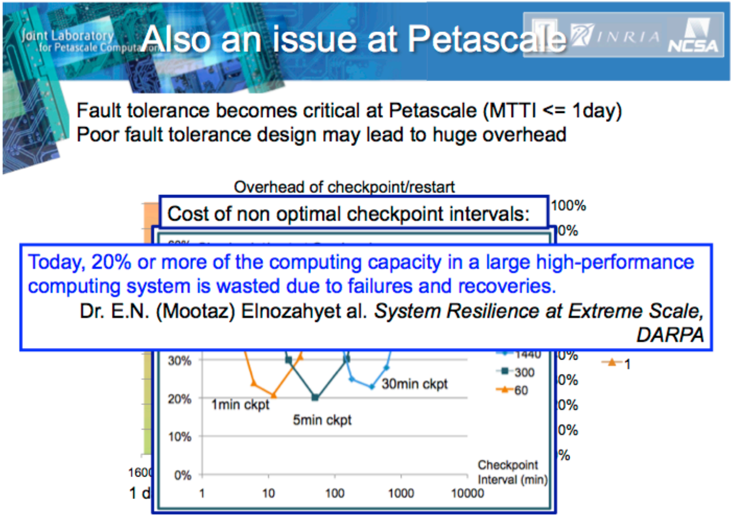
More no ... Failures)

Exascale

$\neq$ **Petascale** $\times 1000$

Plan

Even for today's platforms (courtesy F. Cappello)

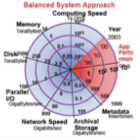


Today, 20% or more of the computing capacity in a large high-performance computing system is wasted due to failures and recoveries.

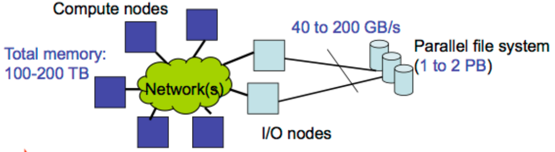
Dr. E.N. (Mootaz) Elnozahyet al. *System Resilience at Extreme Scale, DARPA*

Even for today's platforms (courtesy F. Cappello)

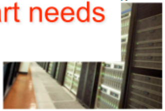
Classic approach for FT: Checkpoint-Restart



Typical "Balanced Architecture" for PetaScale Computers



RoadRunner



Without optimization, Checkpoint-Restart needs about 1h! (~30 minutes each)

Systems	Perf.	Ckpt time	Source
RoadRunner	1PF	~20 min.	Panasas
LLNL BG/L	500 TF	>20 min.	LLNL
LLNL Zeus	11TF	26 min.	LLNL
YYY BG/P	100 TF	~30 min.	YYY

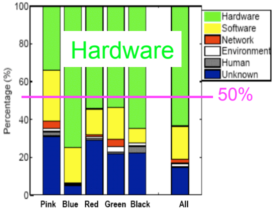
LLNL BG/L

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 faults that lead to application failures
- This includes all hardware faults, and some software ones
- Will use terms *fault* and *failure* interchangeably

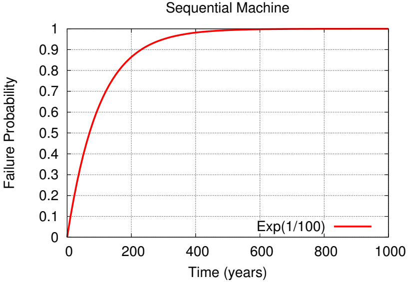
A few definitions

- Many types of faults: software error, hardware malfunction, memory corruption
- Many possible behaviors: silent, transient, unrecoverable
- Restrict to faults that lead to application failures
- This includes all hardware faults, and some software ones
- Will use terms *fault* and *failure* interchangeably

- First question: quantify the rate or frequency at which these faults strike!

Plan

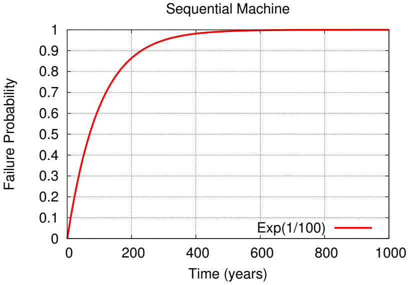
Failure distributions: (1) Exponential



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

- Probability density function (pdf): $f(t) = \lambda e^{-\lambda t} dt$ for $t \geq 0$
- Cumulative distribution function (cdf): $F(t) = 1 - e^{-\lambda t}$
- Mean: $\mu = \frac{1}{\lambda}$

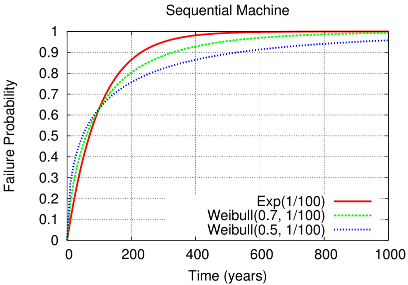
Failure distributions: (1) Exponential



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

- $\mathbb{P}(X \leq t) = 1 - e^{-\lambda t} dt$ (by definition)
- Memoryless property: $\mathbb{P}(X \geq t + s \,|\, X \geq s) = \mathbb{P}(X \geq t)$ (for all $t, s \geq 0$): at any instant, time to next failure does not depend upon time elapsed since last failure
- Mean Time Between Failures (MTBF) $\mu = \mathbb{E}(X) = \frac{1}{\lambda}$

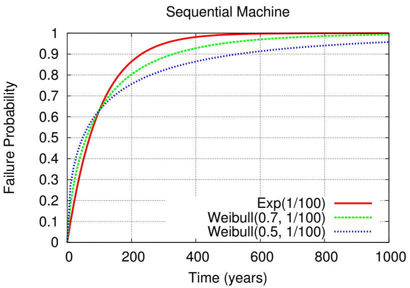
Failure distributions: (2) Weibull



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: $\mu = \frac{1}{\lambda}\Gamma(1 + \frac{1}{k})$

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: (3) with several processors

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

- If the MTBF is μ with one processor,
  what is its value with p processors?

Failure distributions: (3) with several processors

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

- If the MTBF is μ with one processor,
  what is its value 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 $Exp(\lambda)$:

$$\min\left(Exp(\lambda_1), Exp(\lambda_2)\right) = Exp(\lambda_1 + \lambda_2)$$

$$\mu = \frac{1}{\lambda} \Rightarrow \mu_p = \frac{\mu}{p}$$

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

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

$$\mu = \frac{1}{\lambda}\Gamma(1 + \frac{1}{k}) \Rightarrow \mu_p = \frac{\mu}{p^{1/k}}$$

Without rejuvenation (= real life)

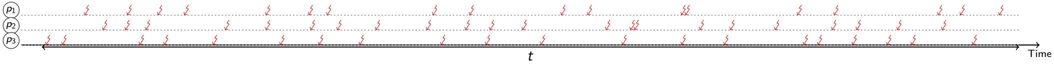
- Rebooting only faulty processor
- Platform failure distribution
  $\Rightarrow$ superposition of p IID processor distributions
  $\Rightarrow$ IID only for Exponential
- Define μ_p by

$$\lim_{F \to +\infty} \frac{n(F)}{F} = \frac{1}{\mu_p}$$

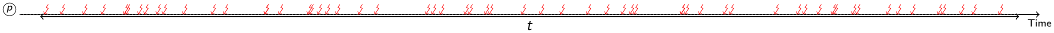
$n(F)$ = number of platform failures until time F is exceeded

Theorem: $\mu_p = \dfrac{\mu}{p}$ for arbitrary distributions

Intuition



If three processors have around 20 faults during a time t ($\mu = \frac{t}{20}$)...



...during the same time, the platform has around 60 faults ($\mu_p = \frac{t}{60}$)

Theorem: $\mu_p = \frac{\mu}{p}$ for arbitrary distributions

With one processor:

- $n(F)$ = number of failures until time F is exceeded
- X_i iid random variables for inter-arrival times, with $\mathbb{E}(X_i) = \mu$
- $\sum_{i=1}^{n(F)-1} X_i \leq F \leq \sum_{i=1}^{n(F)} X_i$
- Wald's equation: $(\mathbb{E}(n(F)) - 1)\mu \leq F \leq \mathbb{E}(n(F))\,\mu$
- $\lim_{F \to +\infty} \frac{\mathbb{E}(n(F))}{F} = \frac{1}{\mu}$

Theorem: $\mu_p = \frac{\mu}{p}$ for arbitrary distributions

With one processor:

- $n(F)$ = number of failures until time F is exceeded
- X_i iid random variables for inter-arrival times, with $\mathbb{E}(X_i) = \mu$
- $\sum_{i=1}^{n(F)-1} X_i \leq F \leq \sum_{i=1}^{n(F)} X_i$
- Wald's equation: $(\mathbb{E}(n(F)) - 1)\mu \leq F \leq \mathbb{E}(n(F))\,\mu$
- $\lim_{F\to+\infty} \frac{\mathbb{E}(n(F))}{F} = \frac{1}{\mu}$

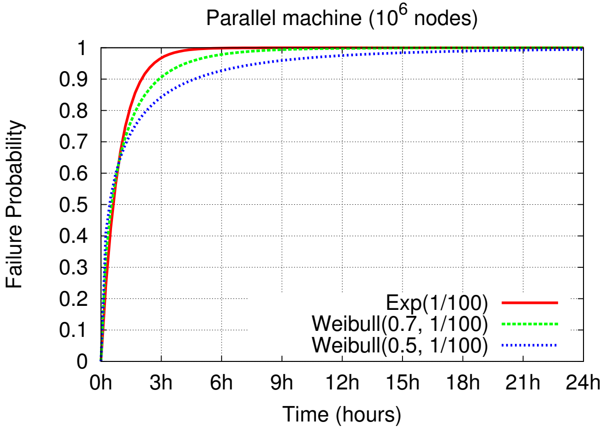
With p processors:

- $n(F)$ = number of platform failures until time F is exceeded
- $n_q(F)$ = number of those failures that strike processor q
- $n_q(F) + 1$ = number of failures on processor q until time F is exceeded (except for processor with last-failure)
- $\lim_{F\to+\infty} \frac{n_q(F)}{F} = \frac{1}{\mu}$ as above
- $\lim_{F\to+\infty} \frac{n(F)}{F} = \frac{1}{\mu_p}$ by definition
- Hence $\mu_p = \frac{\mu}{p}$ because $n(F) = \sum_{q=1}^{p} n_q(F)$

Values from the literature

- MTBF of one processor: between 1 and 125 years
- Shape parameters for Weibull: $k = 0.5$ or $k = 0.7$
- Failure trace archive from INRIA (http://fta.inria.fr)
- Computer Failure Data Repository from LANL (http://institutes.lanl.gov/data/fdata)

Does it matter?



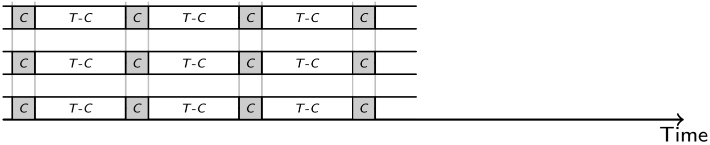
After infant mortality and before aging,
instantaneous failure rate of computer platforms is almost constant

Summary for the road

- MTBF key parameter and $\mu_p = \frac{\mu}{p}$ ☺
- Exponential distribution OK for most purposes ☺
- Assume failure independence while not (completely) true ☹

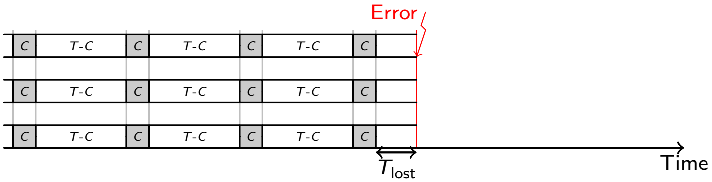
Plan

Periodic checkpointing, definitions



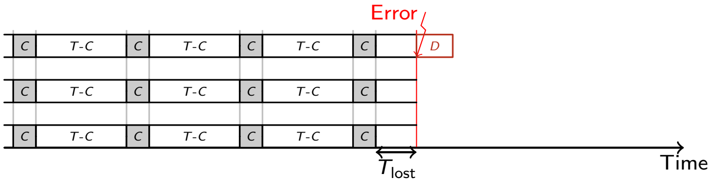
- Periodic checkpointing policy of period T
- Time to checkpoint C
- Time lost in case of a failure T_{lost}
- Downtime D
- Time for recovery R

Periodic checkpointing, definitions



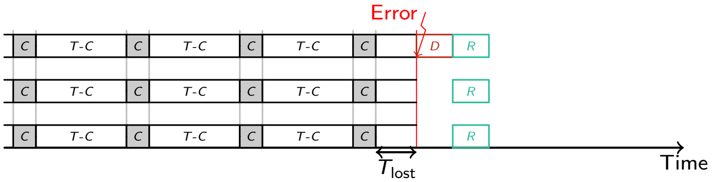
- Periodic checkpointing policy of period T
- Time to checkpoint C
- Time lost in case of a failure T_{lost}
- Downtime D
- Time for recovery R

Periodic checkpointing, definitions



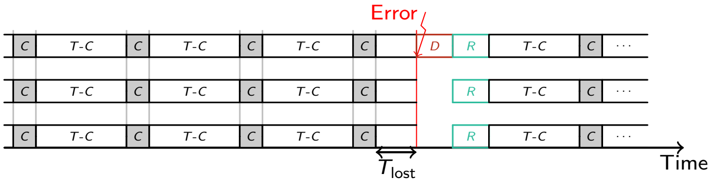
- Periodic checkpointing policy of period T
- Time to checkpoint C
- Time lost in case of a failure T_{lost}
- Downtime D
- Time for recovery R

Periodic checkpointing, definitions



- Periodic checkpointing policy of period T
- Time to checkpoint C
- Time lost in case of a failure T_{lost}
- Downtime D
- Time for recovery R

Periodic checkpointing, definitions



- Periodic checkpointing policy of period T
- Time to checkpoint C
- Time lost in case of a failure T_{lost}
- Downtime D
- Time for recovery R

Motivational example (1)

Strategies

1. Only one checkpoint at the end of the execution;
2. Three checkpoints during the execution, after every 10 minutes of work;
3. Five checkpoints during the execution, after every 6 minutes of work.

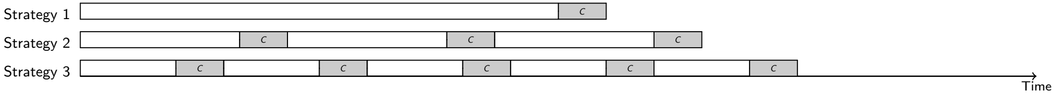
Scenarios

(A) A large time between faults (in this example, no fault during the execution);

(B) A medium time between faults (only one fault at the 19th minute);

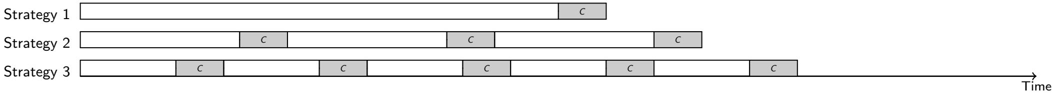
(C) A small time between faults (one fault at the 19th, 42th , and 62th minutes.).

Motivational example (2)

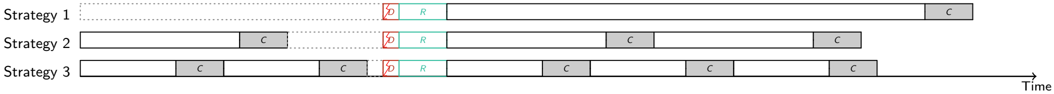


Large MTBF: there are no or very few faults. Checkpointing is too expensive. The first strategy wins.

Motivational example (2)

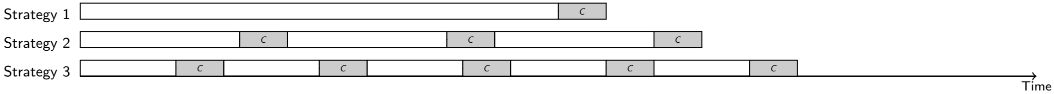


Large MTBF: there are no or very few faults. Checkpointing is too expensive. The first strategy wins.

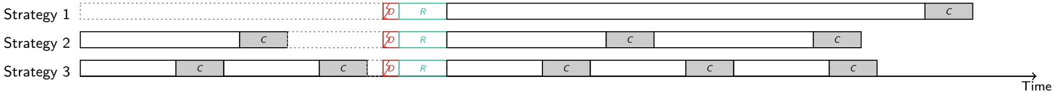


Medium MTBF: there are more faults. It is good to checkpoint, but not too frequently, because of the corresponding overhead. The second strategy wins.

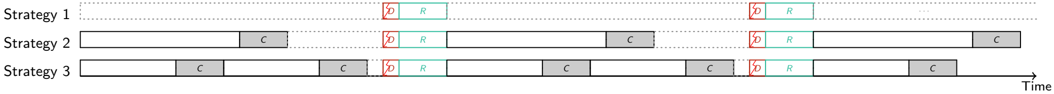
Motivational example (2)



Large MTBF: there are no or very few faults. Checkpointing is too expensive. The first strategy wins.



Medium MTBF: there are more faults. It is good to checkpoint, but not too frequently, because of the corresponding overhead. The second strategy wins.



Small MTBF: there are many faults. The cost of the checkpoints is paid off because the time lost due to faults is dramatically reduced. The third strategy wins.

Optimization objective

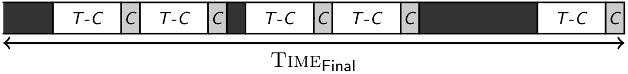
Waste: Fraction of time not spent for useful computations. If an application needs $\textsc{Time}_{\mathsf{base}}$ volume of compute, and the final execution time is $\textsc{Time}_{\mathsf{final}}$:

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

Equivalent to minimizing $\textsc{Time}_{\mathsf{Final}}$: $(1 - \textsc{Waste})\textsc{Time}_{\mathsf{Final}} = \textsc{Time}_{\mathsf{base}}$, but more convenient (get rid of notion of Time, and end of computation).

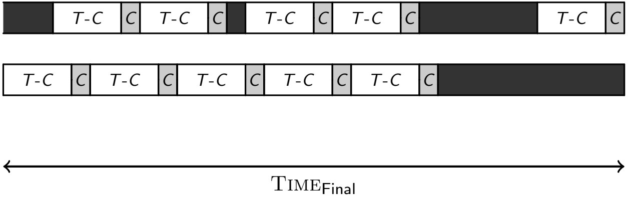
Plan

Waste?



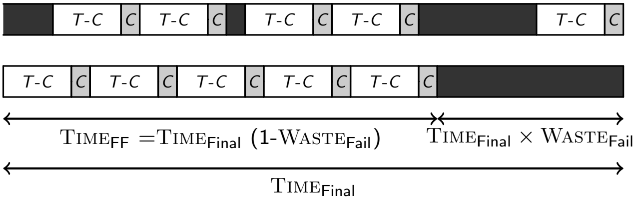
An execution. Black intervals correspond to work destroyed by faults, downtimes, and recoveries.

Waste?



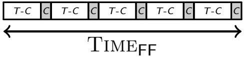
An execution. Black intervals correspond to work destroyed by faults, downtimes, and recoveries.

Waste?



An execution. Black intervals correspond to work destroyed by faults, downtimes, and recoveries.

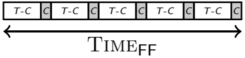
Waste in a fault-free execution



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

$$\text{TIME}_{\text{FF}} = \text{TIME}_{\text{base}} + \#checkpoints \times C$$

Waste in a fault-free execution



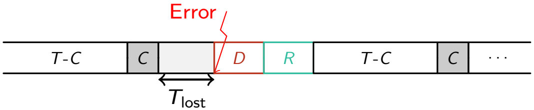
- $\textsc{Time}_{\mathsf{base}}$: application base time
- $\textsc{Time}_{\mathsf{FF}}$: with periodic checkpoints but failure-free

$$\textsc{Time}_{\mathsf{FF}} = \textsc{Time}_{\mathsf{base}} + \#checkpoints \times C$$

$$\#checkpoints = \left\lceil \frac{\textsc{Time}_{\mathsf{base}}}{T - C} \right\rceil \approx \frac{\textsc{Time}_{\mathsf{base}}}{T - C} \text{ (valid for large jobs)}$$

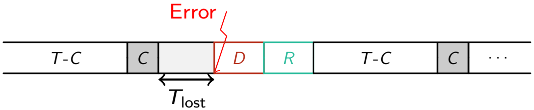
$$\textsc{Waste}_{\mathsf{FF}} = \frac{\textsc{Time}_{\mathsf{FF}} - \textsc{Time}_{\mathsf{base}}}{\textsc{Time}_{\mathsf{FF}}} = \frac{C}{T}$$

Waste due to failures



$$\text{TIME}_{\text{Final}} = \text{TIME}_{\text{FF}} + N_{\text{faults}}\,(T_{\text{lost}} + D + R)$$

Waste due to failures

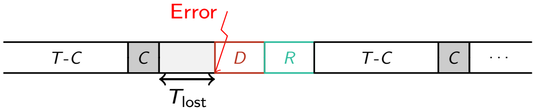


$$\text{TIME}_{\text{Final}} = \text{TIME}_{\text{FF}} + N_{\text{faults}}\,(T_{\text{lost}} + D + R)$$

$$\text{TIME}_{\text{Final}} = \text{TIME}_{\text{FF}} + \frac{\text{TIME}_{\text{Final}}}{\mu}\,(T/2 + D + R)$$

$\Rightarrow$ Instants when periods begin and failures strike are independent

Waste due to failures



$$\text{TIME}_{\text{Final}} = \text{TIME}_{\text{FF}} + N_{\text{faults}}\,(T_{\text{lost}} + D + R)$$

$$\text{TIME}_{\text{Final}} = \text{TIME}_{\text{FF}} + \frac{\text{TIME}_{\text{Final}}}{\mu}\,(T/2 + D + R)$$

$$\text{WASTE}_{\text{Fail}} = \frac{\text{TIME}_{\text{Final}} - \text{TIME}_{\text{FF}}}{\text{TIME}_{\text{Final}}} = \frac{1}{\mu}\,(T/2 + D + R)$$

Total waste

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

Reminder

- $\text{Time}_{\text{base}}$: application base time
- Time_{FF}: with periodic checkpoints but failure-free
- $\text{Time}_{\text{Final}}$: final time

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

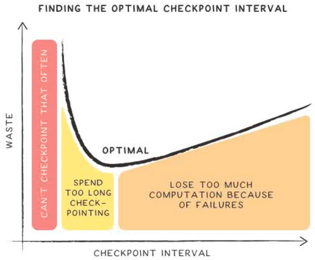
$$\text{Waste} = \frac{C}{T} + \left(1 - \frac{C}{T}\right)\frac{1}{\mu}\left(D + R + \frac{T}{2}\right)$$

Waste is minimized for

$$T = \sqrt{2(\mu - (D + R))C}$$

WRAP UP

- Capping periods, and enforcing a lower bound on MTBF $\Rightarrow$ mandatory for mathematical rigor ☹
- Not needed for practical purposes ☺
  - actual job execution uses optimal value
  - account for multiple faults by re-executing work until success
- Approach surprisingly robust ☺



Plan

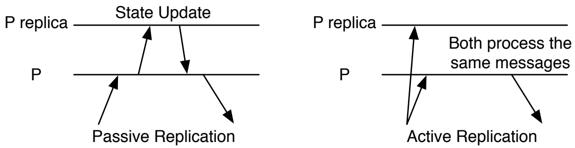
Your time to work

- Back to your thesis
- Saving periodically is a pain: you get interrupted in important paragraphs and then lose your train of thoughts.
- You would rather save at the end of sections.
- Let's assume you know how many sections you are going to write, and what their sizes are going to be.

- Propose a model for this new problem
- Solve it!

Plan

Replication

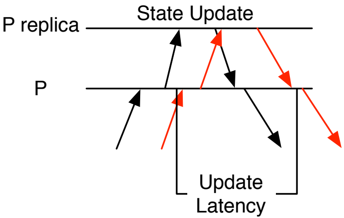


Idea

- Each process is replicated on a resource that has small chance to be hit by the same failure as its replica
- In case of failure, one of the replicas will continue working, while the other recovers
- Passive Replication / Active Replication

Passive Replication

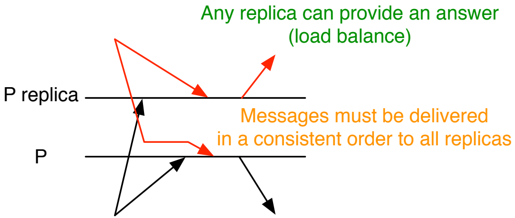
Replication



Challenges

- Passive replication: latency of state update
- Active replication: ordering of decision → internal additional communications

Replication

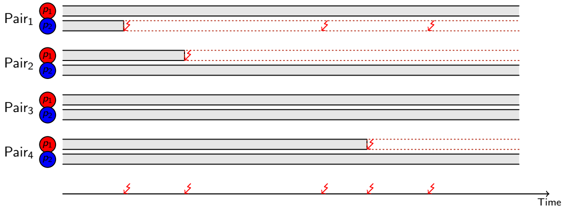


Challenges

- Passive replication: latency of state update
- Active replication: ordering of decision → internal additional communications

Plan

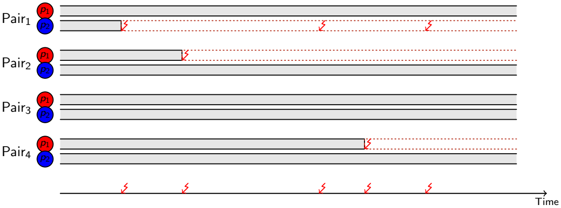
Modeling replication



Processor pairs for replication: each blue processor is paired with a red processor and they do the same work.

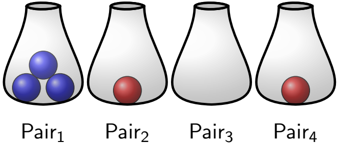
- How do you write the job model?
- How do you write the objective function?
- How do you compare to the checkpoint strategy?

Modeling replication



Processor pairs for replication: each blue processor is paired with a red processor and they do the same work.

- How do you write the job model?
- How do you write the objective function?
- How do you compare to the checkpoint strategy?



Modeling the state of the platform as a balls-into-bins problem. Colors of balls are important: $\neq$ birthday problem!

Conclusion

This was not a class about resilience but a class about scheduling ☺. If you are interested, still need to read about:

- Technical Protocols for resilience (dealing with messages etc)
- Different type of checkpointing (blocking v Asynchronous, coordinated v uncoordinated, hierarchical, in memory etc)
- Combining replication and checkpointing
- ABFT
- ...

You can see the Tutorial by Bosilca, Bouteiller, Hérault and Robert:
http://fault-tolerance.org/2018/11/09/sc18/