Optimization problems in the presence of failures on large-scale parallel systems

Anne Benoit

LIP, Ecole Normale Supérieure de Lyon, France

Anne.Benoit@ens-lyon.fr

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

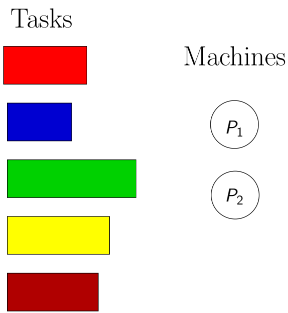
PDCO workshop, in conjunction with IPDPS
Rio de Janeiro, Brazil, May 20, 2019

Motivation

Optimization problems: focus on scheduling, i.e., allocating resources to applications to optimize some performance metrics

- Resources: Large-scale distributed systems with millions of components
- Applications: Parallel applications, expressed as a set of tasks, or divisible application with some work to complete
- Performance metrics: Of course we are concerned with the performance of the applications, but also with resilience and energy consumption

Classical scheduling problems

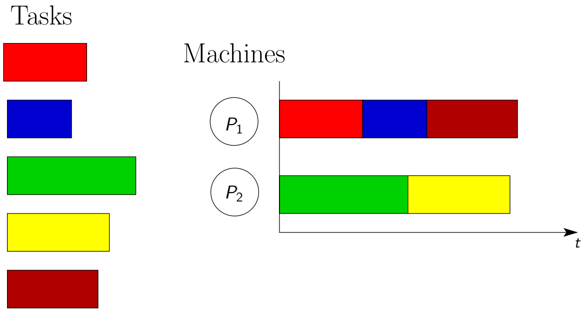


Objectives:

- Minimizing total execution time (C_{max})
- Minimizing weighted sum of execution times $\sum_i w_i C_i$

Results: NP-completeness, algorithms, approximation algorithms, (in-)approximation bounds

Classical scheduling problems

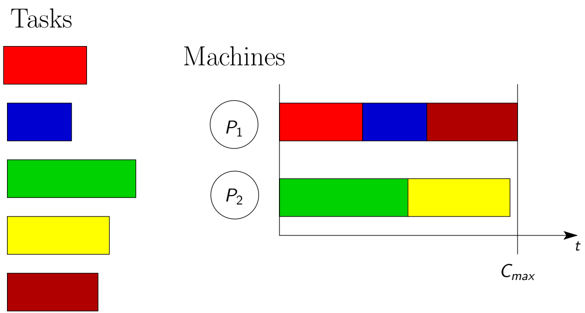


Objectives:

- Minimizing total execution time (C_{max})
- Minimizing weighted sum of execution times $\sum_i w_i C_i$

Results: NP-completeness, algorithms, approximation algorithms, (in-)approximation bounds

Classical scheduling problems

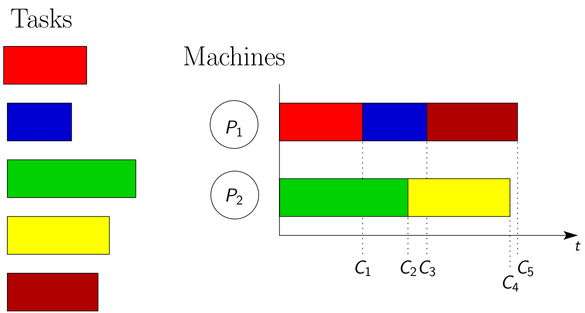


Objectives:

- Minimizing total execution time (C_{max})
- Minimizing weighted sum of execution times $\sum_i w_i C_i$

Results: NP-completeness, algorithms, approximation algorithms, (in-)approximation bounds

Classical scheduling problems

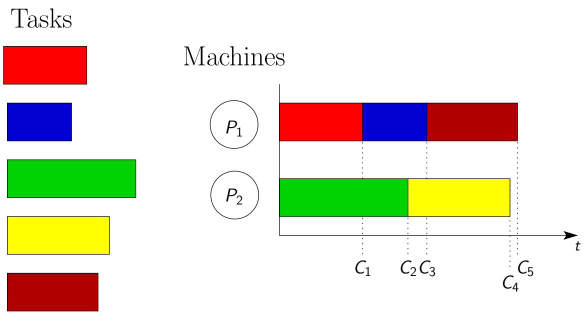


Objectives:

- Minimizing total execution time (C_{max})
- Minimizing weighted sum of execution times $\sum_i w_i C_i$

Results: NP-completeness, algorithms, approximation algorithms, (in-)approximation bounds

Classical scheduling problems



Objectives:

- Minimizing total execution time (C_{max})
- Minimizing weighted sum of execution times $\sum_i w_i C_i$

Results: NP-completeness, algorithms, approximation algorithms, (in-)approximation bounds

Dealing with failures

- Consider one processor (e.g. in your laptop)
  - Mean Time Between Failures (MTBF) = 100 years
  - (Almost) no failures in practice ☺

  Why bother about failures?

- **Theorem:** The MTBF decreases linearly with the number of processors!
  With 36500 processors:
  - MTBF = 1 day
  - A failure every day on average!

A large simulation can run for weeks, hence it will face failures ☹

Dealing with failures

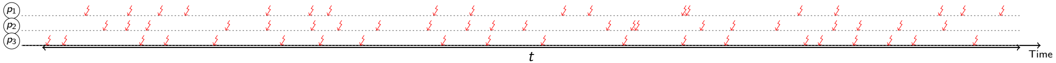
- Consider one processor (e.g. in your laptop)
  - Mean Time Between Failures (MTBF) = 100 years
  - (Almost) no failures in practice ☺

  Why bother about failures?

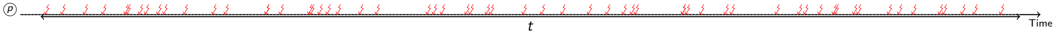
- **Theorem:** The MTBF decreases linearly with the number of processors!
  With 36500 processors:
  - MTBF = 1 day
  - A failure every day on average!

A large simulation can run for weeks, hence it will face failures ☹

Intuition



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

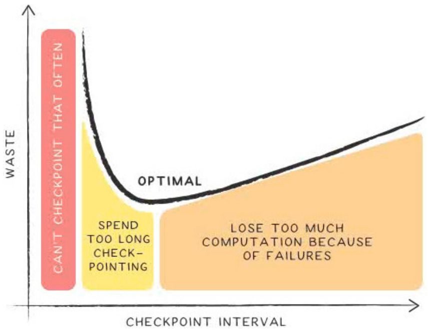


...during the same time, the platform has around 60 faults $(MTBF_{platform} = \frac{t}{60})$

So, how to deal with failures?

Failures usually handled by adding redundancy:

- Replicate the work (for instance, use only half of the processors, and the other half is used to redo the same computation)
- Checkpoint the application: Periodically save the state of the application on stable storage, so that we can restart in case of failure without loosing everything



Another crucial issue: Energy consumption

"The internet begins with coal"

- Nowadays: more than 90 billion kilowatt-hours of electricity a year; requires 34 giant (500 megawatt) coal-powered plants, and produces huge CO_2 emissions
- Explosion of artificial intelligence; AI is hungry for processing power! Need to double data centers in next four years
  $\rightarrow$ how to get enough power?
- Failures: Redundant work consumes even more energy

Energy and power awareness $\rightsquigarrow$ crucial for both environmental and economical reasons

Outline

1. Checkpointing for resilience
   - How to cope with errors?
   - Optimization objective and optimal period
   - Optimal period when accounting for energy consumption

2. Combining checkpoint with replication
   - Replication analysis
   - Simulations

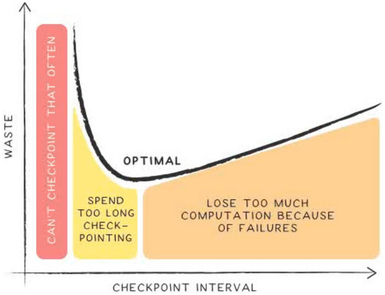
3. Back to task scheduling

4. Summary and need for trade-offs

Introduction to resilience

- Fail-stop errors:
  - Component failures (node, network, power, ...)
  - Application fails and data is lost
- Silent data corruptions:
  - Bit flip (Disk, RAM, Cache, Bus, ...)
  - Detection is not immediate, and we may get wrong results

How often should we checkpoint to minimize the waste, i.e., the time lost because of resilience techniques and failures?

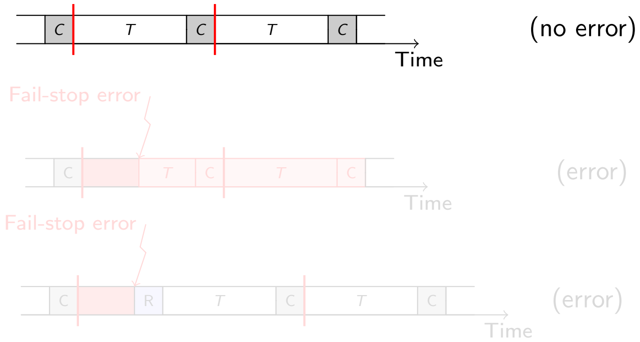


Outline

1. Checkpointing for resilience
   - How to cope with errors?
   - Optimization objective and optimal period
   - Optimal period when accounting for energy consumption
2. Combining checkpoint with replication
   - Replication analysis
   - Simulations
3. Back to task scheduling
4. Summary and need for trade-offs

Coping with fail-stop errors

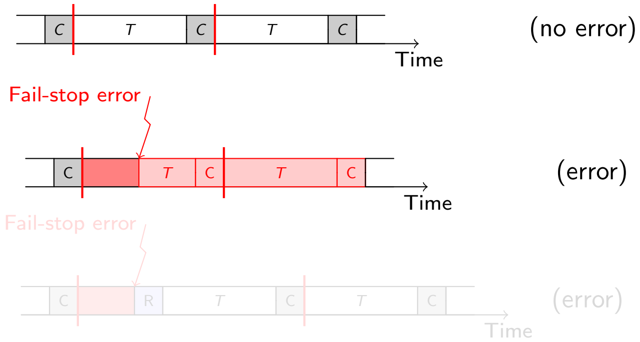
Periodic checkpoint, rollback, and recovery:



- Coordinated checkpointing (the platform is a giant macro-processor)
- Assume instantaneous interruption and detection
- Rollback to last checkpoint and re-execute

Coping with fail-stop errors

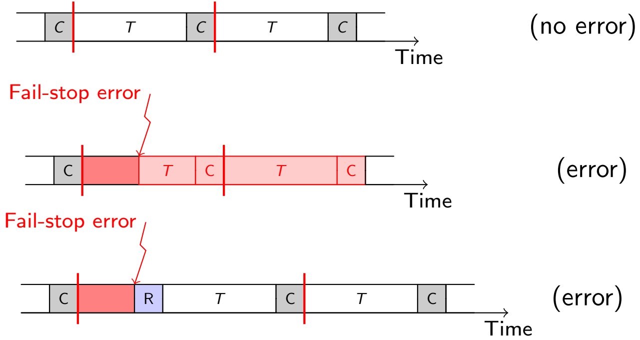
Periodic checkpoint, rollback, and recovery:



- Coordinated checkpointing (the platform is a giant macro-processor)
- Assume instantaneous interruption and detection
- Rollback to last checkpoint and re-execute

Coping with fail-stop errors

Periodic checkpoint, rollback, and recovery:



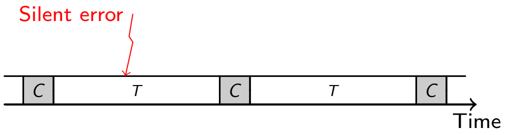
- Coordinated checkpointing (the platform is a giant macro-processor)
- Assume instantaneous interruption and detection
- Rollback to last checkpoint and re-execute

Coping with silent errors

Silent error = detection latency

Error is detected only when corrupted data is activated

Same approach?



Keep multiple checkpoints?

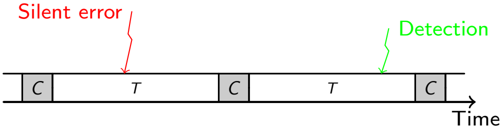
Which checkpoint to recover from?

Need an active method to detect silent errors!

Coping with silent errors

Silent error = detection latency
Error is detected only when corrupted data is activated

Same approach?



Keep multiple checkpoints?

Which checkpoint to recover from?

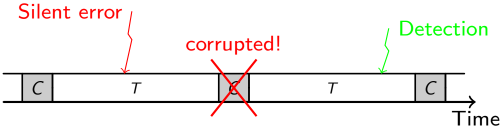
Need an active method to detect silent errors!

Coping with silent errors

Silent error = detection latency

Error is detected only when corrupted data is activated

Same approach?



Keep multiple checkpoints?

Which checkpoint to recover from?

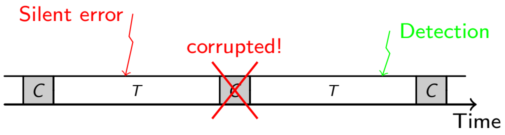
Need an active method to detect silent errors!

Coping with silent errors

Silent error = detection latency

Error is detected only when corrupted data is activated

Same approach?



Keep multiple checkpoints?

Which checkpoint to recover from?

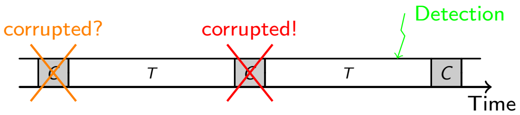
Need an active method to detect silent errors!

Coping with silent errors

Silent error = detection latency

Error is detected only when corrupted data is activated

Same approach?



Keep multiple checkpoints?

Which checkpoint to recover from?

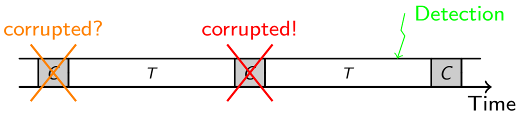
Need an active method to detect silent errors!

Coping with silent errors

Silent error = detection latency

Error is detected only when corrupted data is activated

Same approach?



Keep multiple checkpoints?

Which checkpoint to recover from?

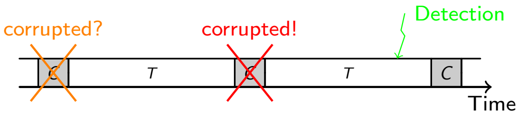
Need an active method to detect silent errors!

Coping with silent errors

Silent error = detection latency

Error is detected only when corrupted data is activated

Same approach?



Keep multiple checkpoints?

Which checkpoint to recover from?

Need an active method to detect silent errors!

Methods for detecting silent errors

General-purpose approaches

- Replication [*Fiala et al. 2012*] or triple modular redundancy and voting [*Lyons and Vanderkulk 1962*]

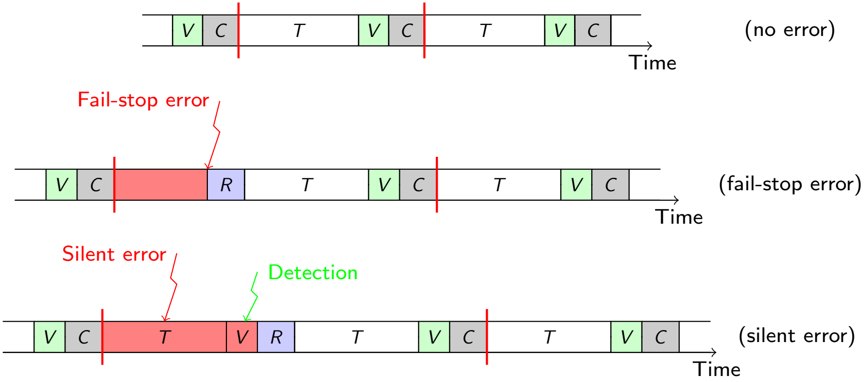
Application-specific approaches

- 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, Chen 2013*]

Data-analytics approaches

- 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*]

Coping with fail-stop and silent errors

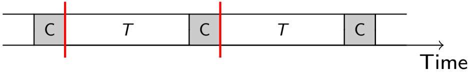


What is the optimal checkpointing period?

Outline

1. Checkpointing for resilience
   - How to cope with errors?
   - Optimization objective and optimal period
   - Optimal period when accounting for energy consumption
2. Combining checkpoint with replication
   - Replication analysis
   - Simulations
3. Back to task scheduling
4. Summary and need for trade-offs

Optimization objective (1/2)



- T is the **pattern length** (time without failures)
- C is the checkpoint cost
- $\mathbb{E}(T)$ is the expected execution time of the pattern

By definition, the overhead of the pattern is defined as:

$$\mathbb{H}(T) = \frac{\mathbb{E}(T)}{T} - 1$$

The overhead measures the fraction of **extra time** due to:

- Checkpoints
- Recoveries and re-executions (failures)

The goal is to minimize the quantity: $\mathbb{H}(T)$

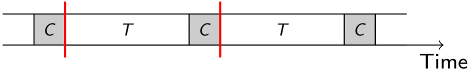
Optimization objective (2/2)

- Goal: Find the **optimal pattern length** T^*, so that the overhead is minimized
- Overhead: $\mathbb{H}(T) = \frac{\mathbb{E}(T)}{T} - 1$

1. Compute expected execution time $\mathbb{E}(T)$ (exact formula)
2. Compute overhead $\mathbb{H}(T)$ (first-order approximation)
3. Derive optimal T^*: fail-stop errors
4. Derive optimal T^*: silent errors
5. Derive optimal T^*: both

1. Expected execution time $\mathbb{E}(T)$

- T: Pattern length; C: Checkpoint time; R: Recovery time
- $\lambda^f = \frac{1}{\mu^f}$: Fail-stop error rate



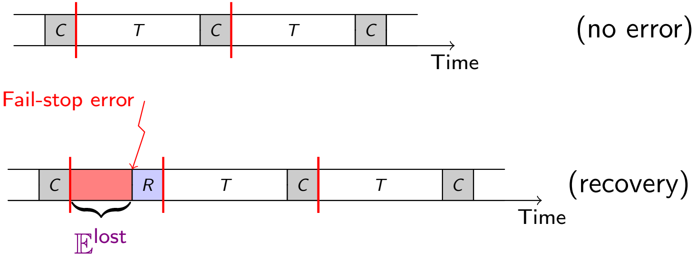
(no error)

$$\mathbb{E}(T) = \mathbb{P}_{no-error}\,(T + C)$$

$$+$$

1. Expected execution time $\mathbb{E}(T)$

- T: Pattern length; C: Checkpoint time; R: Recovery time
- $\lambda^f = \frac{1}{\mu^f}$: Fail-stop error rate



$$\mathbb{E}(T) = \mathbb{P}_{no-error}\,(T + C) + \mathbb{P}_{error}\left(\mathbb{E}^{lost} + R + \mathbb{E}(T)\right)$$

1. Expected execution time $\mathbb{E}(T)$

Assume that failures follow an **exponential distribution** $\text{Exp}(\lambda^f)$

- Independent errors (memoryless property)

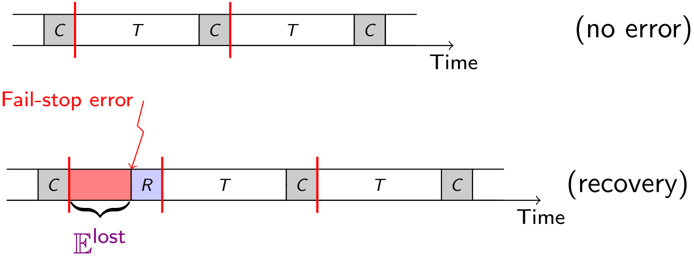
There is at least one error before time t with probability:

$$\mathbb{P}(X \leq t) = 1 - e^{-\lambda^f t} \quad \text{(cdf)}$$

Probability of failure / no-failure

- $\mathbb{P}_{error} = 1 - e^{-\lambda^f T}$
- $\mathbb{P}_{no-error} = e^{-\lambda^f T}$

1. Expected execution time $\mathbb{E}(T)$



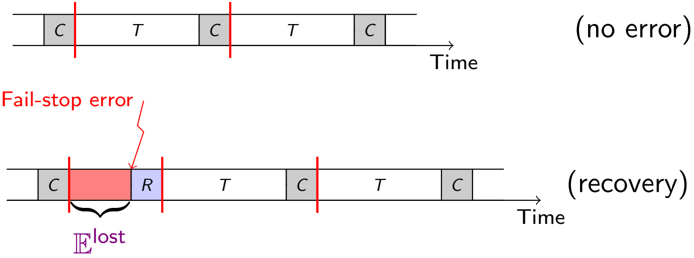
$$\mathbb{E}(T) = e^{-\lambda^f T}(T+C) + (1 - e^{-\lambda^f T})\left(\mathbb{E}^{\text{lost}} + R + \mathbb{E}(T)\right)$$

$$= T + C + (e^{\lambda^f T} - 1)\left(\mathbb{E}^{\text{lost}} + R\right)$$

$\mathbb{E}^{\text{lost}}$ is the time lost when the failure strikes:

$$\mathbb{E}^{\text{lost}} = \int_0^\infty t\mathbb{P}(X = t | X < T)dt = \frac{1}{\lambda^f} - \frac{T}{e^{\lambda^f T} - 1} = \frac{T}{2} + o(\lambda^f T)$$

1. Expected execution time $\mathbb{E}(T)$



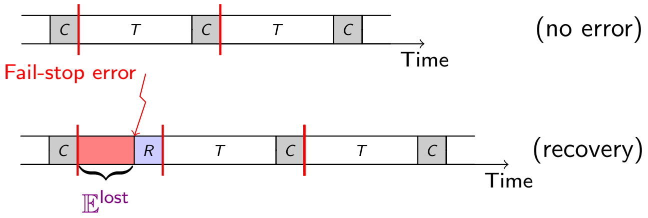
$$\mathbb{E}(T) = e^{-\lambda^f T}(T + C) + (1 - e^{-\lambda^f T})\left(\mathbb{E}^{\text{lost}} + R + \mathbb{E}(T)\right)$$

$$= T + C + (e^{\lambda^f T} - 1)\left(\mathbb{E}^{\text{lost}} + R\right)$$

$\mathbb{E}^{\text{lost}}$ is the time lost when the failure strikes:

$$\mathbb{E}^{\text{lost}} = \int_0^\infty t\mathbb{P}(X = t | X < T)dt = \frac{1}{\lambda^f} - \frac{T}{e^{\lambda^f T} - 1} = \frac{T}{2} + o(\lambda^f T)$$

2. Compute overhead $\mathbb{H}(T)$



We use Taylor series to approximate $e^{-\lambda^f T}$ up to first-order terms:

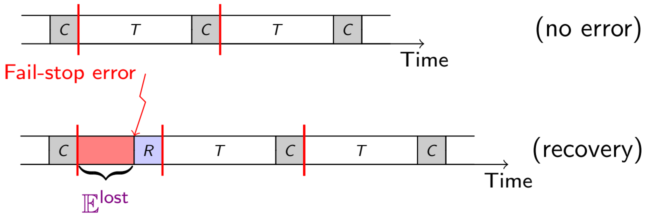
$$e^{-\lambda^f T} = 1 - \lambda^f T + o(\lambda^f T)$$

Works well provided that $\lambda^f << T, C, R$:

$$\mathbb{E}(T) = T + C + \lambda^f T \left(\frac{T}{2} + R\right) + o(\lambda^f T)$$

Finally, we get the overhead of the pattern: $\mathbb{H}(T) = \frac{C}{T} + \lambda^f \frac{T}{2} + o(\lambda^f T)$

2. Compute overhead $\mathbb{H}(T)$



We use Taylor series to approximate $e^{-\lambda^f T}$ up to first-order terms:

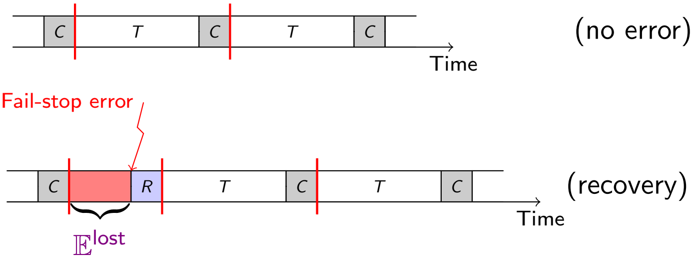
$$e^{-\lambda^f T} = 1 - \lambda^f T + o(\lambda^f T)$$

Works well provided that $\lambda^f << T, C, R$:

$$\mathbb{E}(T) = T + C + \lambda^f T \left(\frac{T}{2} + R \right) + o(\lambda^f T)$$

Finally, we get the overhead of the pattern: $\mathbb{H}(T) = \frac{C}{T} + \lambda^f \frac{T}{2} + o(\lambda^f T)$

3. Derive optimal T^*: Fail-stop errors



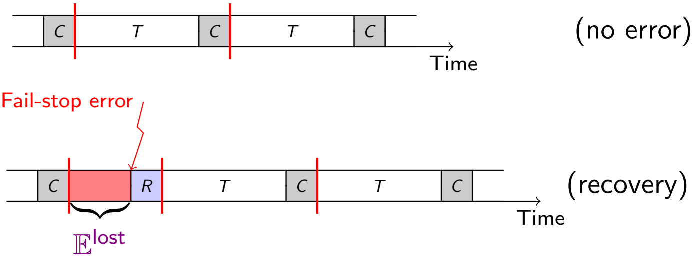
$$\mathbb{H}(T) = \frac{C}{T} + \lambda^f \frac{T}{2} + o(\lambda^f T)$$

We solve: $\frac{\partial \mathbb{H}(T)}{\partial T} = -\frac{C}{T^2} + \frac{\lambda^f}{2} = 0$

Finally, we retrieve:

$$T^* = \sqrt{\frac{2C}{\lambda^f}} = \sqrt{2\mu^f C}$$

3. Derive optimal T^*: Fail-stop errors



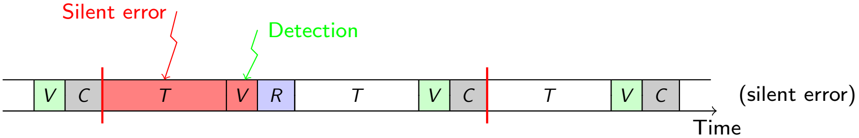
$$\mathbb{H}(T) = \frac{C}{T} + \lambda^f \frac{T}{2} + o(\lambda^f T)$$

We solve: $\frac{\partial \mathbb{H}(T)}{\partial T} = -\frac{C}{T^2} + \frac{\lambda^f}{2} = 0$

Finally, we retrieve:

$$T^* = \sqrt{\frac{2C}{\lambda^f}} = \sqrt{2\mu^f C}$$

4. Derive optimal T^*: Silent errors



Similar to fail-stop except:

- $\lambda^f \rightarrow \lambda^s$
- $\mathbb{E}^{\text{lost}} = T$
- V: verification time

Using the same approach:

$$\mathbb{H}(T) = \frac{C+V}{T} + \underbrace{\lambda^s T}_{\textit{silent}} + o(\lambda^s T)$$

5. Derive optimal T^*: Both errors

$$\mathbb{H}(T) = \frac{C+V}{T} + \underbrace{\lambda^f \frac{T}{2}}_{fail-stop} + \underbrace{\lambda^s T}_{silent} + o(\lambda T)$$

First-order approximations [Young 1974, Daly 2006, AB et al. 2016]

	Fail-stop errors	Silent errors	Both errors
Pattern	$T+C$	$T+V+C$	$T+V+C$
Optimal T^*	$\sqrt{\frac{C}{\frac{\lambda^f}{2}}}$	$\sqrt{\frac{V+C}{\lambda^s}}$	$\sqrt{\frac{V+C}{\lambda^s+\frac{\lambda^f}{2}}}$
Overhead $\mathbb{H}^*$	$2\sqrt{\frac{\lambda^f}{2}C}$	$2\sqrt{\lambda^s(V+C)}$	$2\sqrt{\left(\lambda^s+\frac{\lambda^f}{2}\right)(V+C)}$

Is this optimal for energy consumption?

5. Derive optimal T^*: Both errors

$$\mathbb{H}(T) = \frac{C+V}{T} + \underbrace{\lambda^f \frac{T}{2}}_{fail-stop} + \underbrace{\lambda^s T}_{silent} + o(\lambda T)$$

First-order approximations [Young 1974, Daly 2006, AB et al. 2016]

	Fail-stop errors	Silent errors	Both errors
Pattern	$T+C$	$T+V+C$	$T+V+C$
Optimal T^*	$\sqrt{\frac{C}{\frac{\lambda^f}{2}}}$	$\sqrt{\frac{V+C}{\lambda^s}}$	$\sqrt{\frac{V+C}{\lambda^s+\frac{\lambda^f}{2}}}$
Overhead $\mathbb{H}^*$	$2\sqrt{\frac{\lambda^f}{2}C}$	$2\sqrt{\lambda^s(V+C)}$	$2\sqrt{\left(\lambda^s+\frac{\lambda^f}{2}\right)(V+C)}$

Is this optimal for energy consumption?

Outline

1. Checkpointing for resilience
   - How to cope with errors?
   - Optimization objective and optimal period
   - Optimal period when accounting for energy consumption
2. Combining checkpoint with replication
   - Replication analysis
   - Simulations
3. Back to task scheduling
4. Summary and need for trade-offs

Energy model (1/2)

- Modern processors equipped with dynamic voltage and frequency scaling (DVFS) capability
- Power consumption of processing unit is $P_{idle} + \kappa\sigma^3$, where $\kappa > 0$ and σ is the processing speed
- Error rate: May also depend on processing speed
  - $\lambda(\sigma)$ follows a U-shaped curve
  - increases exponentially with decreased processing speed σ
  - increases also with increased speed because of high temperature

Energy model (2/2)

- Total power consumption depends on:
  - P_{idle}: static power dissipated when platform is on (even idle)
  - $P_{cpu}(\sigma)$: dynamic power spent by operating CPU at speed σ
  - P_{io}: dynamic power spent by I/O transfers (checkpoints and recoveries)

- Computation and verification: power depends upon σ (total time $T_{cpu}(\sigma)$)
- Checkpointing and recovering: I/O transfers (total time T_{io})
- Total energy consumption:

$$Energy(\sigma) = T_{cpu}(\sigma)(P_{idle} + P_{cpu}(\sigma)) + T_{io}(P_{idle} + P_{io})$$

  - Checkpoint: $E^C = C(P_{idle} + P_{io})$
  - Recover: $E^R = R(P_{idle} + P_{io})$
  - Verify at speed σ: $E^V(\sigma) = V(\sigma)(P_{idle} + P_{cpu}(\sigma))$

Bi-criteria problem

Linear combination of execution time and energy consumption:

$$a \cdot Time + b \cdot Energy$$

Theorem

Application subject to both fail-stop and silent errors

Minimize $a \cdot Time + b \cdot Energy$

The optimal checkpointing period is $T^*(\sigma) = \sqrt{\frac{2(V(\sigma)+C_e(\sigma))}{\lambda^f(\sigma)+2\lambda^s(\sigma)}}$,

where $C_e(\sigma) = \frac{a+b(P_{idle}+P_{io})}{a+b(P_{idle}+P_{cpu}(\sigma))} C$

Similar optimal period as without energy, but account for new parameters!

$$T^* = \sqrt{\frac{2(V+C)}{\lambda^f+2\lambda^s}}$$

Bi-criteria problem

Linear combination of execution time and energy consumption:

$$a \cdot Time + b \cdot Energy$$

Theorem

Application subject to both fail-stop and silent errors
Minimize $a \cdot Time + b \cdot Energy$
The optimal checkpointing period is $T^*(\sigma) = \sqrt{\frac{2(V(\sigma)+C_e(\sigma))}{\lambda^f(\sigma)+2\lambda^s(\sigma)}}$,
where $C_e(\sigma) = \frac{a+b(P_{idle}+P_{io})}{a+b(P_{idle}+P_{cpu}(\sigma))} C$

Similar optimal period as without energy, but account for new parameters!

$$T^* = \sqrt{\frac{2(V+C)}{\lambda^f+2\lambda^s}}$$

Outline

1. Checkpointing for resilience
   - How to cope with errors?
   - Optimization objective and optimal period
   - Optimal period when accounting for energy consumption
2. Combining checkpoint with replication
   - Replication analysis
   - Simulations
3. Back to task scheduling
4. Summary and need for trade-offs

When Amdahl meets Young/Daly

Error-free speedup with P processors and α sequential fraction:

Amdahl's Law: $S(P) = \frac{1}{\alpha + \frac{1-\alpha}{P}}$

- Bounded above by $1/\alpha$
- Strictly increasing function of P

Allocating more processors on an error-prone platform?

- Higher error-free speedup ☺
- More errors/faults ☹
  - More frequent checkpointing ☹
    - More resilience overhead ☹

We can compute optimal processor allocation and checkpointing interval!

How is replication used?

On a Q-processor platform, application is replicated n times:

- **Duplication**: each replica has $P = Q/2$ processors
- **Triplication**: each replica has $P = Q/3$ processors
- **General case**: each replica has $P = Q/n$ processors

Having more replicas on an error-prone platform?

- Lower error-free speedup ☹
- More resilient ☺
  - Smaller checkpointing frequency ☺
    - Less resilience overhead ☺

Optimal replication level, processor allocation per replica, and checkpointing interval?

How is replication used?

On a Q-processor platform, application is replicated n times:

- **Duplication**: each replica has $P = Q/2$ processors
- **Triplication**: each replica has $P = Q/3$ processors
- **General case**: each replica has $P = Q/n$ processors

Having more replicas on an error-prone platform?

- Lower error-free speedup ☹
- More resilient ☺
  - Smaller checkpointing frequency ☺
    - Less resilience overhead ☺

Optimal replication level, processor allocation per replica, and checkpointing interval?

How is replication used?

On a Q-processor platform, application is replicated n times:

- **Duplication**: each replica has $P = Q/2$ processors
- **Triplication**: each replica has $P = Q/3$ processors
- **General case**: each replica has $P = Q/n$ processors

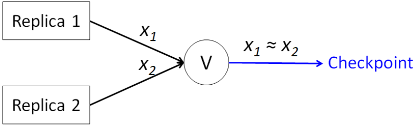
Having more replicas on an error-prone platform?

- Lower error-free speedup ☹
- More resilient ☺
  - Smaller checkpointing frequency ☺
    - Less resilience overhead ☺

Optimal replication level, processor allocation per replica, and checkpointing interval?

Why is replication useful?

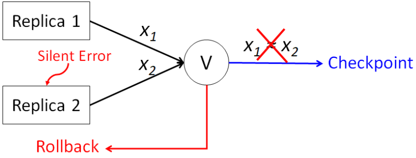
- **Error detection (duplication)**:



- **Error correction (triplication)**:

Why is replication useful?

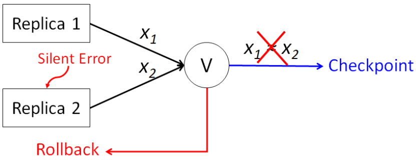
- **Error detection (duplication)**:



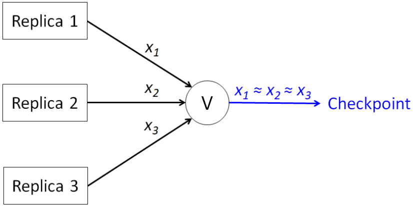
- **Error correction (triplication)**:

Why is replication useful?

- **Error detection (duplication)**:

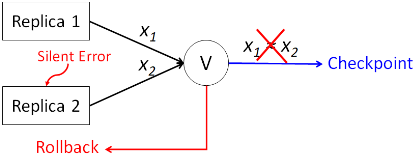


- **Error correction (triplication)**:

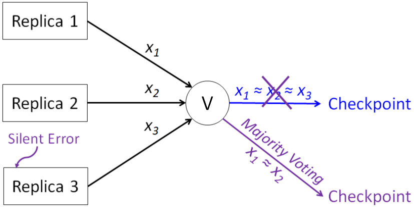


Why is replication useful?

- **Error detection (duplication)**:

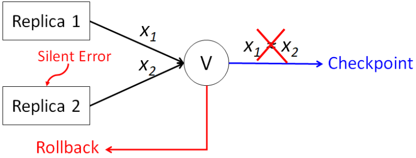


- **Error correction (triplication)**:

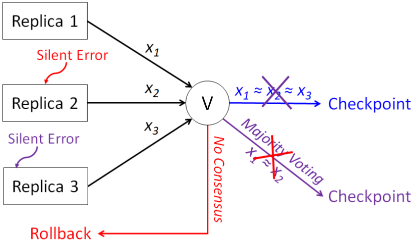


Why is replication useful?

- **Error detection (duplication)**:



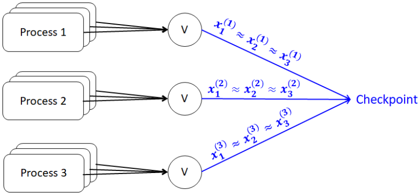
- **Error correction (triplication)**:



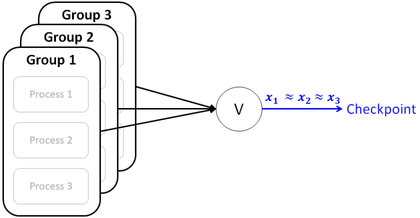
Outline

Two replication modes

- **Process replication**:

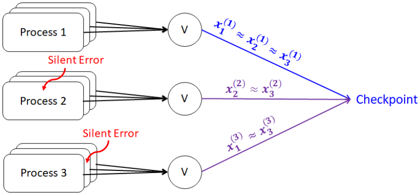


- **Group replication**:

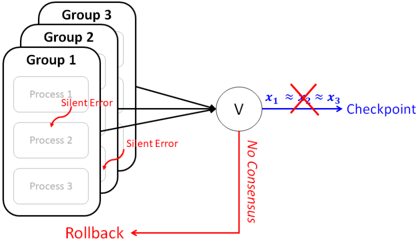


Two replication modes

- **Process replication**:



- **Group replication**:



Probability of failure

Independent process error distribution:

- Exponential $Exp(\lambda)$, $\lambda = 1/\mu$ (Memoryless)
- *Error probability of one process during T time of computation:*

$$\mathbb{P}(T) = 1 - e^{-\lambda T}$$

Process triplication:

- *Failure probability of any triplicated process:*

$$\mathbb{P}_3^{\text{prc}}(T,1) = \binom{3}{2}\Big(1 - \mathbb{P}(T)\Big)\mathbb{P}(T)^2 + \mathbb{P}(T)^3$$

$$= 3e^{-\lambda T}\left(1 - e^{-\lambda T}\right)^2 + \left(1 - e^{-\lambda T}\right)^3 = 1 - 3e^{-2\lambda T} + 2e^{-3\lambda T}$$

- *Failure probability of P-process application:*

$$\mathbb{P}_3^{\text{prc}}(T,P) = 1 - \mathbb{P}(\text{“No process fails”})$$

$$= 1 - \left(1 - \mathbb{P}_3^{\text{prc}}(T,1)\right)^P = 1 - \left(3e^{-2\lambda T} - 2e^{-3\lambda T}\right)^P$$

Probability of failure

Independent process error distribution:

- Exponential $Exp(\lambda)$, $\lambda = 1/\mu$ (Memoryless)
- *Error probability of one process during T time of computation:*

$$\mathbb{P}(T) = 1 - e^{-\lambda T}$$

Process triplication:

- *Failure probability of any triplicated process:*

$$\mathbb{P}_3^{\text{prc}}(T,1) = \binom{3}{2}\Big(1 - \mathbb{P}(T)\Big)\mathbb{P}(T)^2 + \mathbb{P}(T)^3$$

$$= 3e^{-\lambda T}\left(1 - e^{-\lambda T}\right)^2 + \left(1 - e^{-\lambda T}\right)^3 = 1 - 3e^{-2\lambda T} + 2e^{-3\lambda T}$$

- *Failure probability of P-process application:*

$$\mathbb{P}_3^{\text{prc}}(T,P) = 1 - \mathbb{P}(\text{“No process fails”})$$

$$= 1 - \left(1 - \mathbb{P}_3^{\text{prc}}(T,1)\right)^P = 1 - \left(3e^{-2\lambda T} - 2e^{-3\lambda T}\right)^P$$

Probability of failure

Independent process error distribution:

- Exponential $Exp(\lambda)$, $\lambda = 1/\mu$ (Memoryless)
- *Error probability of one process during T time of computation:*

$$\mathbb{P}(T) = 1 - e^{-\lambda T}$$

Process triplication:

- *Failure probability of any triplicated process:*

$$\mathbb{P}_3^{\text{prc}}(T,1) = \binom{3}{2}\Big(1 - \mathbb{P}(T)\Big)\mathbb{P}(T)^2 + \mathbb{P}(T)^3$$
$$= 3e^{-\lambda T}\left(1 - e^{-\lambda T}\right)^2 + \left(1 - e^{-\lambda T}\right)^3 = 1 - 3e^{-2\lambda T} + 2e^{-3\lambda T}$$

- *Failure probability of P-process application:*

$$\mathbb{P}_3^{\text{prc}}(T,P) = 1 - \mathbb{P}(\text{``No process fails''})$$
$$= 1 - \left(1 - \mathbb{P}_3^{\text{prc}}(T,1)\right)^P = 1 - \left(3e^{-2\lambda T} - 2e^{-3\lambda T}\right)^P$$

Probability of failure

Group triplication:

- *Failure probability of any P-process group:*

$$\mathbb{P}_1^{\text{grp}}(T,P) = 1 - \mathbb{P}(\text{“No process in group fails”}) = 1 - (1-\mathbb{P}(T))^P = 1 - e^{-\lambda PT}$$

- *Failure probability of three-group application:*

$$\mathbb{P}_3^{\text{grp}}(T,P) = \binom{3}{2}\left(1-\mathbb{P}_1^{\text{grp}}(T,1)\right)\mathbb{P}_1^{\text{grp}}(T,1)^2 + \mathbb{P}_1^{\text{grp}}(T,1)^3$$
$$= 3e^{-\lambda PT}\left(1-e^{-\lambda PT}\right)^2 + \left(1-e^{-\lambda PT}\right)^3$$
$$= 1 - 3e^{-2\lambda PT} + 2e^{-3\lambda PT}$$
$$> 1 - \left(3e^{-2\lambda T} - 2e^{-3\lambda T}\right)^P = \mathbb{P}_3^{\text{prc}}(T,P)$$

What about duplication? (any error kills both cases)

$$\mathbb{P}_2^{\text{prc}}(T,P) = \mathbb{P}_2^{\text{grp}}(T,P) = 1 - e^{-2\lambda PT}$$

Probability of failure

Group triplication:

- *Failure probability of any P-process group:*

$$\mathbb{P}_1^{\text{grp}}(T,P) = 1 - \mathbb{P}(\text{"No process in group fails"})$$
$$= 1 - (1-\mathbb{P}(T))^P = 1 - e^{-\lambda PT}$$

- *Failure probability of three-group application:*

$$\mathbb{P}_3^{\text{grp}}(T,P) = \binom{3}{2}\left(1-\mathbb{P}_1^{\text{grp}}(T,1)\right)\mathbb{P}_1^{\text{grp}}(T,1)^2 + \mathbb{P}_1^{\text{grp}}(T,1)^3$$
$$= 3e^{-\lambda PT}\left(1-e^{-\lambda PT}\right)^2 + \left(1-e^{-\lambda PT}\right)^3$$
$$= 1 - 3e^{-2\lambda PT} + 2e^{-3\lambda PT}$$
$$> 1 - \left(3e^{-2\lambda T} - 2e^{-3\lambda T}\right)^P = \mathbb{P}_3^{\text{prc}}(T,P)$$

What about duplication? (any error kills both cases)

$$\mathbb{P}_2^{\text{prc}}(T,P) = \mathbb{P}_2^{\text{grp}}(T,P) = 1 - e^{-2\lambda PT}$$

Probability of failure

Group triplication:

- *Failure probability of any P-process group:*

$$\mathbb{P}_1^{\text{grp}}(T,P) = 1 - \mathbb{P}(\text{“No process in group fails”}) = 1 - \left(1 - \mathbb{P}(T)\right)^P = 1 - e^{-\lambda P T}$$

- *Failure probability of three-group application:*

$$\mathbb{P}_3^{\text{grp}}(T,P) = \binom{3}{2}\left(1 - \mathbb{P}_1^{\text{grp}}(T,1)\right)\mathbb{P}_1^{\text{grp}}(T,1)^2 + \mathbb{P}_1^{\text{grp}}(T,1)^3$$
$$= 3e^{-\lambda P T}\left(1 - e^{-\lambda P T}\right)^2 + \left(1 - e^{-\lambda P T}\right)^3$$
$$= 1 - 3e^{-2\lambda P T} + 2e^{-3\lambda P T}$$
$$> 1 - \left(3e^{-2\lambda T} - 2e^{-3\lambda T}\right)^P = \mathbb{P}_3^{\text{prc}}(T,P)$$

What about duplication? (any error kills both cases)

$$\mathbb{P}_2^{\text{prc}}(T,P) = \mathbb{P}_2^{\text{grp}}(T,P) = 1 - e^{-2\lambda P T}$$

Probability of failure

Group triplication:

- *Failure probability of any P-process group:*

$$\mathbb{P}_1^{\text{grp}}(T,P) = 1 - \mathbb{P}(\text{“No process in group fails”}) = 1 - (1-\mathbb{P}(T))^P = 1 - e^{-\lambda PT}$$

- *Failure probability of three-group application:*

$$\mathbb{P}_3^{\text{grp}}(T,P) = \binom{3}{2}(1-\mathbb{P}_1^{\text{grp}}(T,1))\,\mathbb{P}_1^{\text{grp}}(T,1)^2 + \mathbb{P}_1^{\text{grp}}(T,1)^3$$
$$= 3e^{-\lambda PT}\left(1-e^{-\lambda PT}\right)^2 + \left(1-e^{-\lambda PT}\right)^3$$
$$= 1 - 3e^{-2\lambda PT} + 2e^{-3\lambda PT}$$
$$> 1 - \left(3e^{-2\lambda T} - 2e^{-3\lambda T}\right)^P = \mathbb{P}_3^{\text{prc}}(T,P)$$

What about duplication? (any error kills both cases)

$$\mathbb{P}_2^{\text{prc}}(T,P) = \mathbb{P}_2^{\text{grp}}(T,P) = 1 - e^{-2\lambda PT}$$

Two observations

Observation 1 (Implementation)

- **Process replication** is more resilient than group replication (assuming same overhead)
- **Group replication** is easier to implement by treating an application as a blackbox

Observation 2 (Analysis)

Following two scenarios are equivalent w.r.t. failure probability:

- **Group replication** with n replicas, where each replica has P processes and each process has error rate λ
- **Process replication** with one process, which has error rate λP and which is replicated n times

Benefit of analysis: $\text{Group}(n, P, \lambda) \rightarrow \text{Process}(n, 1, \lambda P)$

Two observations

Observation 1 (Implementation)

- **Process replication** is more resilient than group replication (assuming same overhead)
- **Group replication** is easier to implement by treating an application as a blackbox

Observation 2 (Analysis)

Following two scenarios are equivalent w.r.t. failure probability:

- **Group replication** with n replicas, where each replica has P processes and each process has error rate λ
- **Process replication** with one process, which has error rate λP and which is replicated n times

Benefit of analysis: Group$(n, P, \lambda) \rightarrow$ Process$(n, 1, \lambda P)$

Analysis steps

Maximize error-aware speedup

$$\mathbb{S}_n(T,P) = \frac{S(P)}{\mathbb{E}_n(T,P)/T}$$

1. Derive failure probability $\mathbb{P}_n^{\text{prc}}(T,P)$ or $\mathbb{P}_n^{\text{grp}}(T,P)$ — exact
2. Compute expected execution time $\mathbb{E}_n(T,P)$ — exact
3. Compute first-order approx. of error-aware speedup $\mathbb{S}_n(T,P)$
4. Derive optimal T_{opt}, P_{opt} and get $\mathbb{S}_n(T_{\text{opt}},P_{\text{opt}})$
5. Choose right replication level n

Analytical results

Duplication:
On a platform with Q processors and checkpointing cost C, the optimal resilience parameters for *process/group duplication* are:

$$P_{opt} = \min\left\{\frac{Q}{2}, \left(\frac{1}{2}\left(\frac{1-\alpha}{\alpha}\right)^2 \frac{1}{C\lambda}\right)^{\frac{1}{3}}\right\}$$

$$T_{opt} = \left(\frac{C}{2\lambda P_{opt}}\right)^{\frac{1}{2}}$$

$$\mathbb{S}_{opt} = \frac{S(P_{opt})}{1 + 2(2\lambda C P_{opt})^{\frac{1}{2}}}$$

Triplication & (n,k)-replication (k-out-of-n replica consensus):
similar results but different for process and group, less practical for $n > 3$

- For $\alpha > 0$, not necessarily use up all available Q processors
- Checkpointing interval T_{opt} nicely extends Young/Daly's result
- Error-aware speedup $\mathbb{S}_{opt}$ minimally affected for small λ

Analytical results

Duplication:
On a platform with Q processors and checkpointing cost C, the optimal resilience parameters for *process/group duplication* are:

$$P_{\text{opt}} = \min\left\{\frac{Q}{2}, \left(\frac{1}{2}\left(\frac{1-\alpha}{\alpha}\right)^2 \frac{1}{C\lambda}\right)^{\frac{1}{3}}\right\}$$

$$T_{\text{opt}} = \left(\frac{C}{2\lambda P_{\text{opt}}}\right)^{\frac{1}{2}}$$

$$\mathbb{S}_{\text{opt}} = \frac{S(P_{\text{opt}})}{1 + 2(2\lambda C P_{\text{opt}})^{\frac{1}{2}}}$$

Triplication & (n,k)-replication (k-out-of-n replica consensus):
similar results but different for process and group, less practical for $n > 3$

- For $\alpha > 0$, not necessarily use up all available Q processors
- Checkpointing interval T_{opt} nicely extends Young/Daly's result
- Error-aware speedup $\mathbb{S}_{\text{opt}}$ minimally affected for small λ

Analytical results

Duplication:
On a platform with Q processors and checkpointing cost C, the optimal resilience parameters for *process/group duplication* are:

$$P_{\text{opt}} = \min\left\{\frac{Q}{2}, \left(\frac{1}{2}\left(\frac{1-\alpha}{\alpha}\right)^2 \frac{1}{C\lambda}\right)^{\frac{1}{3}}\right\}$$

$$T_{\text{opt}} = \left(\frac{C}{2\lambda P_{\text{opt}}}\right)^{\frac{1}{2}}$$

$$\mathbb{S}_{\text{opt}} = \frac{S(P_{\text{opt}})}{1+2(2\lambda C P_{\text{opt}})^{\frac{1}{2}}}$$

Triplication & (n,k)-replication (k-out-of-n replica consensus):
similar results but different for process and group, less practical for $n > 3$

- For $\alpha > 0$, not necessarily use up all available Q processors
- Checkpointing interval T_{opt} nicely extends Young/Daly's result
- Error-aware speedup $\mathbb{S}_{\text{opt}}$ minimally affected for small λ

Results comparison

For fully parallel jobs, i.e., $\alpha = 0$ (similar for $\alpha > 0$)

- Duplication v.s. Process triplication

$$P_{\text{opt}} = \frac{Q}{2} \qquad P_{\text{opt}} = \frac{Q}{3} \qquad \text{(Processors} \downarrow\text{)}$$

$$T_{\text{opt}} = \sqrt{\frac{C}{\lambda Q}} \qquad T_{\text{opt}} = \sqrt[3]{\frac{C}{2\lambda^2 Q}} \qquad \text{(Chkpt interval} \uparrow\text{)}$$

$$\mathbb{S}_{\text{opt}} = \frac{Q/2}{1 + 2\sqrt{\lambda C Q}} \qquad \mathbb{S}_{\text{opt}} = \frac{Q/3}{1 + 3\sqrt[3]{\left(\frac{\lambda C}{2}\right)^2 Q}} \qquad \text{(Exp. speedup??)}$$

- Process triplication v.s. Group triplication

$$P_{\text{opt}} = \frac{Q}{3} \qquad P_{\text{opt}} = \frac{Q}{3} \qquad \text{(Processors} =\text{)}$$

$$T_{\text{opt}} = \sqrt[3]{\frac{C}{2\lambda^2 Q}} \qquad T_{\text{opt}} = \sqrt[3]{\frac{3C}{2(\lambda Q)^2}} \qquad \text{(Chkpt interval} \downarrow\text{)}$$

$$\mathbb{S}_{\text{opt}} = \frac{Q/3}{1 + 3\sqrt[3]{\left(\frac{\lambda C}{2}\right)^2 Q}} \qquad \mathbb{S}_{\text{opt}} = \frac{Q/3}{1 + 3\sqrt[3]{\frac{1}{3}\left(\frac{\lambda C Q}{2}\right)^2}} \qquad \text{(Exp. speedup} \downarrow\text{)}$$

Results comparison

For fully parallel jobs, i.e., $\alpha = 0$ (similar for $\alpha > 0$)

- Duplication v.s. Process triplication

Duplication	Process triplication	
$P_{opt} = \frac{Q}{2}$	$P_{opt} = \frac{Q}{3}$	(Processors ↓)
$T_{opt} = \sqrt{\frac{C}{\lambda Q}}$	$T_{opt} = \sqrt[3]{\frac{C}{2\lambda^2 Q}}$	(Chkpt interval ↑)
$\mathbb{S}_{opt} = \frac{Q/2}{1 + 2\sqrt{\lambda C Q}}$	$\mathbb{S}_{opt} = \frac{Q/3}{1 + 3\sqrt[3]{\left(\frac{\lambda C}{2}\right)^2 Q}}$	(Exp. speedup??)

- Process triplication v.s. Group triplication

Process triplication	Group triplication	
$P_{opt} = \frac{Q}{3}$	$P_{opt} = \frac{Q}{3}$	(Processors =)
$T_{opt} = \sqrt[3]{\frac{C}{2\lambda^2 Q}}$	$T_{opt} = \sqrt[3]{\frac{3C}{2(\lambda Q)^2}}$	(Chkpt interval ↓)
$\mathbb{S}_{opt} = \frac{Q/3}{1 + 3\sqrt[3]{\left(\frac{\lambda C}{2}\right)^2 Q}}$	$\mathbb{S}_{opt} = \frac{Q/3}{1 + 3\sqrt[3]{\frac{1}{3}\left(\frac{\lambda C Q}{2}\right)^2}}$	(Exp. speedup ↓)

Results comparison

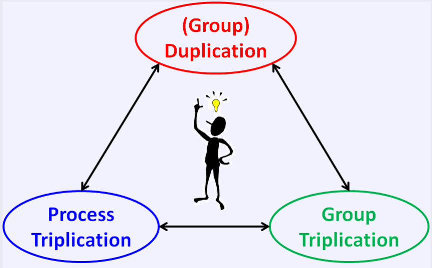
For fully parallel jobs, i.e., $\alpha = 0$ (similar for $\alpha > 0$)

- Duplication v.s. Process triplication

- Pro

Choosing right mode & level of replication

Based on analytical results, app. output structure and system/language support



$1 + 3\sqrt[3]{\left(\frac{\lambda C}{2}\right)^2 Q}$ $1 + 3\sqrt[3]{\frac{1}{3}\left(\frac{\lambda C Q}{2}\right)^2}$

Outline

Simulations

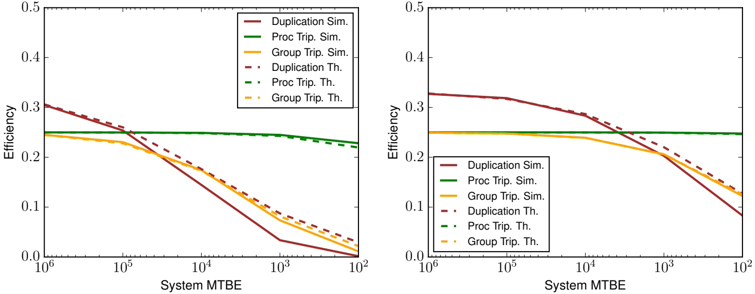
Consider a platform with $Q = 10^6$, and study

$$Efficiency = \frac{\mathbb{S}_{\text{opt}}}{Q}$$

- Impact of MTBE (Mean Time Between Errors – errors lead to failures) and checkpointing cost C
- Impact of sequential fraction α
- Impact of number of processes P

Impact of MTBE and checkpointing cost

$$\alpha = 10^{-6}$$

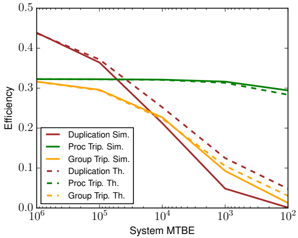


(a) $C = 1800s$ (b) $C = 60s$

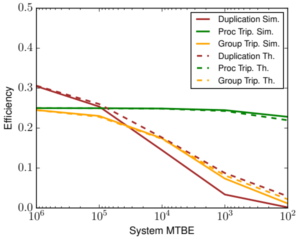
- First-order accurate except for duplication (where P is larger) and with small MTBE
- Duplication can be sufficient for large MTBE, especially for small checkpointing cost

Impact of sequential fraction

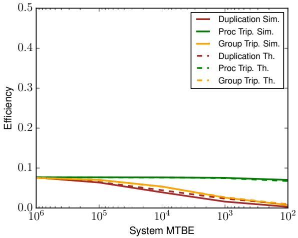
$C = 1800s$



(c) $\alpha = 10^{-7}$



(d) $\alpha = 10^{-6}$

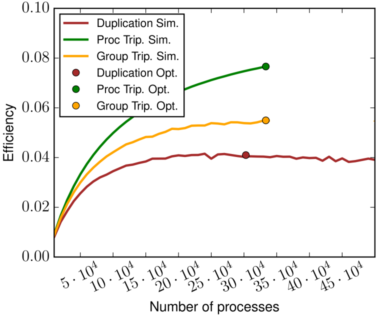


(e) $\alpha = 10^{-5}$

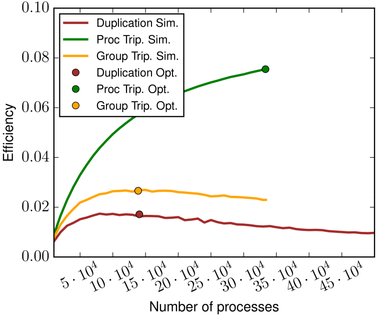
- Increased α reduces efficiency
- Increased α increases minimum MTBE for which duplication is sufficient

Impact of number of processes

$\alpha = 10^{-5}, C = 1800s$



(f) $MTBE = 10^4$



(g) $MTBE = 10^3$

- Efficiency/speedup not strictly increasing with P
- First-order P_{opt} close to actual optimum

What to remember

- "Replication + checkpointing" as a general-purpose fault- tolerance protocol for detecting/correcting silent errors in HPC
- Process replication is more resilient than group replication, but group replication is easier to implement
- Analytical solution for $P_{\text{opt}}, T_{\text{opt}}, and\ \mathbb{S}_{\text{opt}}$ and for choosing right replication mode and level

Outline

1. Checkpointing for resilience
   - How to cope with errors?
   - Optimization objective and optimal period
   - Optimal period when accounting for energy consumption

2. Combining checkpoint with replication
   - Replication analysis
   - Simulations

3. Back to task scheduling

4. Summary and need for trade-offs

Chains of tasks

- High-performance computing (HPC) application: chain of tasks $T_1 \rightarrow T_2 \rightarrow \cdots \rightarrow T_n$
- Parallel tasks executed on the whole platform
- For instance: tightly-coupled computational kernels, image processing applications, ...
- Goal: efficient execution, i.e., minimize total execution time
- Checkpoints can only be done after a task has completed

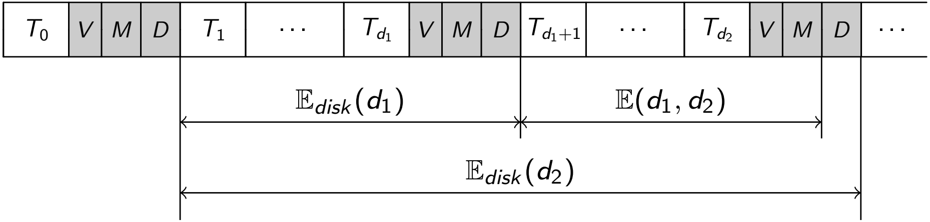
Chains of tasks

- High-performance computing (HPC) application:
  chain of tasks $T_1 \rightarrow T_2 \rightarrow \cdots \rightarrow T_n$
- Parallel tasks executed on the whole platform
- For instance: tightly-coupled computational kernels, image processing applications, ...

- Goal: efficient execution, i.e., minimize total execution time
- Checkpoints can only be done after a task has completed

Dynamic programming algorithm without replication

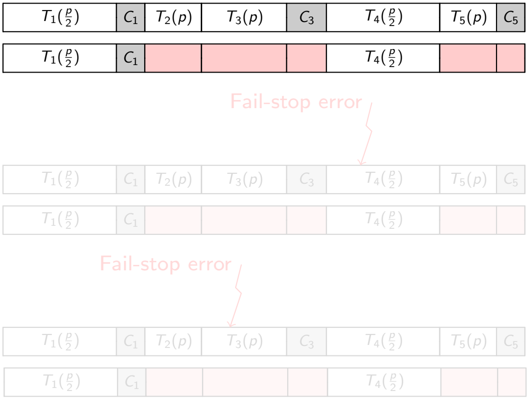
Possibility to add verification, memory checkpoint and disk checkpoint at the end of a task



$$\mathbb{E}_{disk}(d_2) = \min_{0 \le d_1 < d_2} \{\mathbb{E}_{disk}(d_1) + \mathbb{E}(d_1, d_2) + C_D\}$$

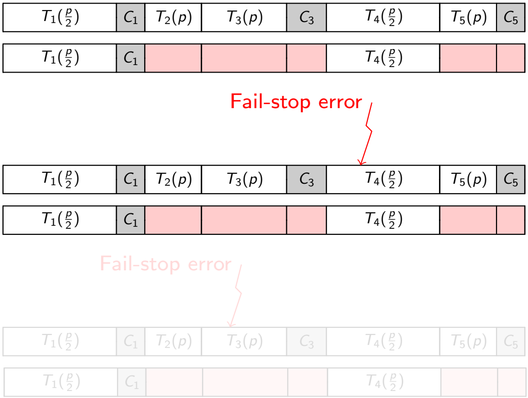
- Initialization: $\mathbb{E}_{disk}(0) = 0$
- Objective: Compute $\mathbb{E}_{disk}(n)$
- Compute $\mathbb{E}_{disk}(0), \mathbb{E}_{disk}(1), \mathbb{E}_{disk}(2), \ldots, \mathbb{E}_{disk}(n)$ in that order
- Complexity: $O(n^2)$

Coping with fail-stop errors with replication



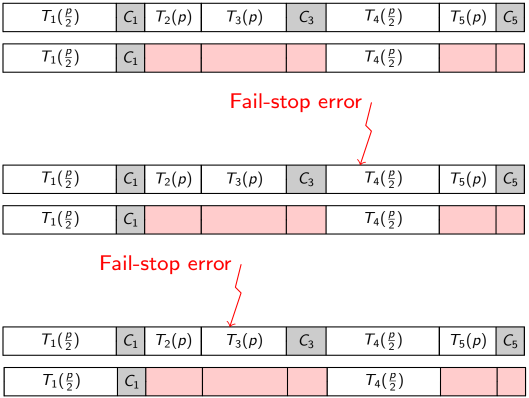
- The whole platform is used at all time, some tasks are replicated
- If failure hits a replicated task, no need to rollback
- Otherwise, rollback to last checkpoint and re-execute

Coping with fail-stop errors with replication



- The whole platform is used at all time, some tasks are replicated
- If failure hits a replicated task, no need to rollback
- Otherwise, rollback to last checkpoint and re-execute

Coping with fail-stop errors with replication



- The whole platform is used at all time, some tasks are replicated
- If failure hits a replicated task, no need to rollback
- Otherwise, rollback to last checkpoint and re-execute

Dynamic programming algorithm with replication

- Recursively computes expectation of optimal time required to execute tasks T_1 to T_i and then checkpoint T_i
- Distinguish whether T_i is replicated or not
- $T_{opt}^{rep}(i)$: knowing that T_i is replicated
- $T_{opt}^{norep}(i)$: knowing that T_i is not replicated
- Solution: $\min\left\{T_{opt}^{rep}(n) + C_n^{rep}, T_{opt}^{norep}(n) + C_n^{norep}\right\}$

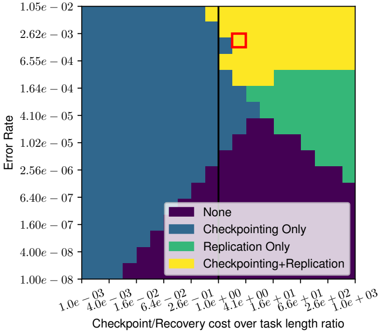
Computing $T_{opt}^{rep}(j)$: j is replicated

$$T_{opt}^{rep}(j) = \min_{1 \le i < j} \left\{ \begin{array}{l} T_{opt}^{rep}(i) + C_i^{rep} + T_{NC}^{rep,rep}(i+1,j), \\ T_{opt}^{rep}(i) + C_i^{rep} + T_{NC}^{norep,rep}(i+1,j), \\ T_{opt}^{norep}(i) + C_i^{norep} + T_{NC}^{rep,rep}(i+1,j), \\ T_{opt}^{norep}(i) + C_i^{norep} + T_{NC}^{norep,rep}(i+1,j), \\ R_1^{rep} + T_{NC}^{rep,rep}(1,j), \\ R_1^{norep} + T_{NC}^{norep,rep}(1,j) \end{array} \right\}$$

- T_i: last checkpointed task before T_j
- T_i can be replicated or not, T_{i+1} can be replicated or not
- $T_{NC}^{A,B}$: no intermediate checkpoint, first/last task replicated or not, previous task checkpointed: complicated formula but done in constant time
- Similar equation for $T_{opt}^{norep}(j)$
- Overall complexity: $O(n^2)$

Comparison to checkpoint only

- With identical tasks
- Reports occ. of checkpoints and replicas in optimal solution
- Checkpointing cost $\leq$ task length $\Rightarrow$ no replication



Summary

- Goal: Minimize execution time of linear workflows
- Decide which task to checkpoint and/or replicate
- Sophisticated dynamic programming algorithms: optimal solutions
- Even when accounting for energy: decide at which speed to execute each task
- Even with k different levels of checkpoints and partial verifications: algorithm in $O(n^{k+5})$
- Simulations: With replication, gain over checkpoint-only approach is quite significant, when checkpoint is costly and error rate is high

Outline

1. Checkpointing for resilience
   - How to cope with errors?
   - Optimization objective and optimal period
   - Optimal period when accounting for energy consumption
2. Combining checkpoint with replication
   - Replication analysis
   - Simulations
3. Back to task scheduling
4. Summary and need for trade-offs

Summary and need for trade-offs

- Two major challenges for Exascale systems:
  - Resilience: need to handle failures
  - Energy: need to reduce energy consumption
- The main optimization objective is often performance, such as execution time, but other criteria must be accounted for
- Many models for which we have the answer:
  - Optimal checkpointing period, with fail-stop / silent errors
  - Use of replication to detect and correct silent errors
  - When to checkpoint, replicate and verify for a chain of tasks?
- Still a lot of challenges to address, and techniques to be developed for many kinds of high-performance applications, making trade-offs between performance, reliability, and energy consumption

Summary and need for trade-offs

- Two major challenges for Exascale systems:
  - Resilience: need to handle failures
  - Energy: need to reduce energy consumption
- The main optimization objective is often performance, such as execution time, but other criteria must be accounted for
- Many models for which we have the answer:
  - Optimal checkpointing period, with fail-stop / silent errors
  - Use of replication to detect and correct silent errors
  - When to checkpoint, replicate and verify for a chain of tasks?
- Still a lot of challenges to address, and techniques to be developed for many kinds of high-performance applications, making trade-offs between performance, reliability, and energy consumption

Summary and need for trade-offs

- Two major challenges for Exascale systems:
  - Resilience: need to handle failures
  - Energy: need to reduce energy consumption
- The main optimization objective is often performance, such as execution time, but other criteria must be accounted for
- Many models for which we have the answer:
  - Optimal checkpointing period, with fail-stop / silent errors
  - Use of replication to detect and correct silent errors
  - When to checkpoint, replicate and verify for a chain of tasks?
- Still a lot of challenges to address, and techniques to be developed for many kinds of high-performance applications, making trade-offs between performance, reliability, and energy consumption

Thanks...

- ... to my co-authors
  - Valentin Le Fèvre, Aurélien Cavelan, Hongyang Sun, Yves Robert
  - Franck Cappello, Padma Raghavan, Florina M. Ciorba
- ... and to Didier El-Baz and Grégoire Danoy for their kind invitation!
- A few references:

  - A. Benoit, A. Cavelan, Y. Robert, H. Sun. Assessing general-purpose algorithms to cope with fail-stop and silent errors. TOPC, 2016
  - A. Benoit, A. Cavelan, F. Cappello, P. Raghavan, Y. Robert, H. Sun. Identifying the right replication level to detect and correct silent errors at scale. FTXS/HPDC, 2017.
  - A. Benoit, A. Cavelan, Y. Robert and H. Sun. Multi-level checkpointing and silent error detection for linear workflows. JoCS, 2017.
  - A. Benoit, A. Cavelan, F. Ciorba, V. Le Fèvre, Y. Robert. Combining checkpointing and replication for reliable execution of linear workflows with fail-stop and silent errors. IJNC, 2019.

Thanks...

- ... to my co-authors
  - Valentin Le Fèvre, Aurélien Cavelan, Hongyang Sun, Yves Robert
  - Franck Cappello, Padma Raghavan, Florina M. Ciorba
- ... and to Didier El-Baz and Grégoire Danoy for their kind invitation!
- A few references:

  - A. Benoit, A. Cavelan, Y. Robert, H. Sun. Assessing general-purpose algorithms to cope with fail-stop and silent errors. TOPC, 2016
  - A. Benoit, A. Cavelan, F. Cappello, P. Raghavan, Y. Robert, H. Sun. Identifying the right replication level to detect and correct silent errors at scale. FTXS/HPDC, 2017.
  - A. Benoit, A. Cavelan, Y. Robert and H. Sun. Multi-level checkpointing and silent error detection for linear workflows. JoCS, 2017.
  - A. Benoit, A. Cavelan, F. Ciorba, V. Le Fèvre, Y. Robert. Combining checkpointing and replication for reliable execution of linear workflows with fail-stop and silent errors. IJNC, 2019.