Fault tolerance techniques for high-performance computing Part 3

Anne Benoit

ENS Lyon

Anne.Benoit@ens-lyon.fr

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

CR02 - 2016/2017

Outline

1. Probabilistic models
2. In-memory checkpointing
   - Double checkpointing algorithm
   - Analysis
   - Triple checkpointing algorithm
   - Experiments
3. Probabilistic models for advanced methods
   - Failure prediction
   - Replication

Outline

Outline

Motivation

- Checkpoint transfer and storage
  $\Rightarrow$ critical issues of rollback/recovery protocols
- Stable storage: high cost
- Distributed in-memory storage:
  - Store checkpoints in local memory $\Rightarrow$ no centralized storage
    ☺ Much better scalability
  - Replicate checkpoints $\Rightarrow$ application survives single failure
    ☹ Still, risk of fatal failure in some (unlikely) scenarios

Outline

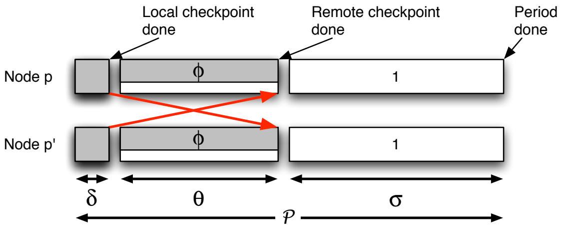
Double checkpoint algorithm

- Platform nodes partitioned into pairs
- Each node in a pair exchanges its checkpoint with its *buddy*
- Each node saves two checkpoints:
  - one locally: storing its own data
  - one remotely: receiving and storing its buddy's data

Two algorithms

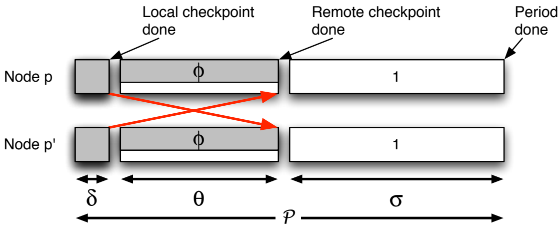
- blocking version by Zheng, Shi and Kalé
- non-blocking version by Ni, Meneses and Kalé

Non-blocking checkpoint algorithm



- Checkpoints taken periodically, with period $P = \delta + \theta + \sigma$
- Phase 1, length δ: local checkpoint, blocking mode. No work
- Phase 2, length θ: remote checkpoint. Overhead ϕ
- Phase 3, length σ: application at full speed 1

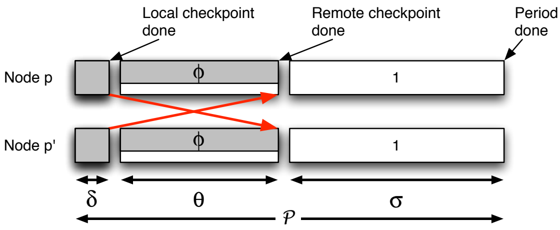
Non-blocking checkpoint algorithm



Work in failure-free period:

$$W = (\theta - \phi) + \sigma = P - \delta - \phi$$

Cost of overlap

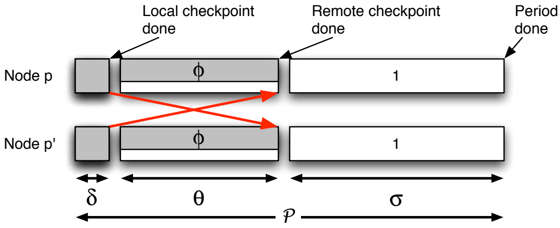


- Overlap computations and checkpoint file exchanges
- Large θ

  $\Rightarrow$ more flexibility to hide cost of file exchange

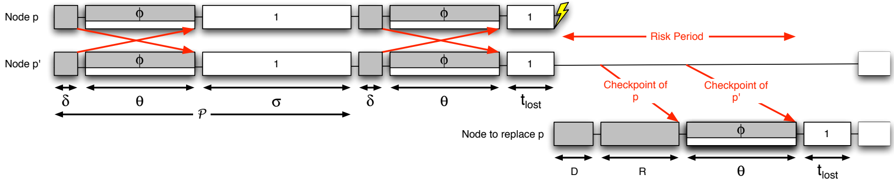
  $\Rightarrow$ smaller overhead ϕ

Cost of overlap



- $\theta = \theta_{\text{min}}$: fastest communication, fully blocking $\Rightarrow \phi = \theta_{\text{min}}$
- $\theta = \theta_{\text{max}}$: full overlap with computation $\Rightarrow \phi = 0$
- Linear interpolation $\theta(\phi) = \theta_{\text{min}} + \alpha(\theta_{\text{min}} - \phi)$
    - $\phi = 0$ for $\theta = \theta_{\text{max}} = (1 + \alpha)\theta_{\text{min}}$
    - α: rate of overhead decrease w.r.t. communication length

Assessing the risk



- After failure: downtime D and recovery from buddy node
- Two checkpoint files lost, must be re-sent to faulty processor
  1. Checkpoint of faulty node, needed for recovery
     $\Rightarrow$ sent as fast as possible, in time $R = \theta_{\text{min}}$
  2. Checkpoint of buddy node, needed in case buddy fails later on
     $\Rightarrow$ **??**
- Application at risk until complete reception of both messages

Checkpoint of buddy node

Scenario DoubleNBL

- File sent at same speed as in regular mode, in time $\theta(\phi)$
- Overhead ϕ
- Favors performance, at the price of higher risk

Scenario DoubleBoF

- File sent as fast as possible, in time $\theta_{\mathsf{min}} = R$
- Overhead R
- Favors risk reduction, at the price of higher overhead

Outline

Computing the waste

Waste
= fraction of time where nodes do not perform useful computations

- T_{base} base time without any overhead due to resilience
- Time for fault-free execution T_{ff}
  - Period $P \Rightarrow W = P - \delta - \phi$ work units
  - $T_{\text{ff}} = \frac{P}{W} T_{\text{base}}$
  - $\left(1 - \frac{\delta+\phi}{P}\right) T_{\text{ff}} = T_{\text{base}}$

Computing the waste

- T expectation of total execution time
  $\rightarrow$ single application
  $\rightarrow$ platform life (many jobs running concurrently)
- In average, failures occur every μ seconds
  $\rightarrow$ platform MTBF $\mu = \mu_{\text{ind}}/p$
- For each failure, $\mathcal{F}$ seconds are lost:

$$T = T_{\text{ff}} + \frac{T}{\mu}\mathcal{F}$$

$$\left(1 - \frac{\mathcal{F}}{\mu}\right)\left(1 - \frac{\delta + \phi}{P}\right) T = T_{\text{base}}$$

Computing the waste

$$\left(1 - \text{WASTE}\right) T = T_{\text{base}}$$

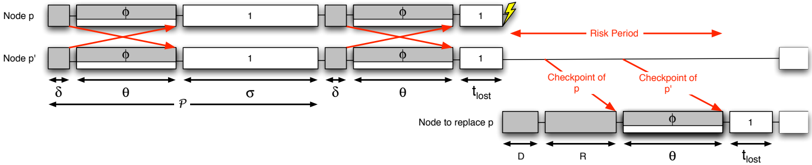
$$\text{WASTE} = 1 - \left(1 - \frac{\mathcal{F}}{\mu}\right)\left(1 - \frac{\delta + \phi}{P}\right)$$

Two sources of overhead:
$\text{WASTE}_{\text{ff}} = \frac{\delta+\phi}{P}$: checkpointing in a fault-free execution
$\text{WASTE}_{\text{fail}} = \frac{\mathcal{F}}{\mu}$: failures striking during execution

$$\text{WASTE} = \text{WASTE}_{\text{fail}} + \text{WASTE}_{\text{ff}} - \text{WASTE}_{\text{fail}}\text{WASTE}_{\text{ff}}$$

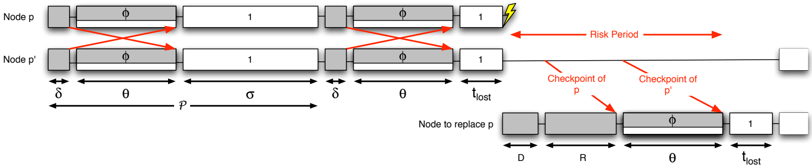
Time lost due to failures

Scenario DOUBLENBL



$$\mathcal{F}_{\text{nbl}} = D + R + \frac{\delta}{P}\mathcal{RE}_1 + \frac{\theta}{P}\mathcal{RE}_2 + \frac{\sigma}{P}\mathcal{RE}_3$$

Failure during third part of period



- No work during $D + R$
- Then re-execution of $W_{lost} = (\theta - \phi) + t_{lost}$
  - First θ seconds: overhead ϕ (receiving buddy checkpoint)
  - Then full speed
- $\mathbb{E}\left(t_{lost}\right) = \frac{\sigma}{2}$ (failures strike uniformly)

$$\mathcal{RE}_3 = \theta + \frac{\sigma}{2}$$

Waste minimization

Scenario DoubleNBL $\mathcal{F}_{\text{nbl}} = D + R + \theta + \frac{P}{2}$

$$\mathcal{TO}_{\text{nbl}} = \sqrt{2(\delta + \phi)(\mu - R - D - \theta)}$$

Scenario DoubleBoF $\mathcal{F}_{\text{bof}} = \mathcal{F}_{\text{nbl}} + R - \phi$

$$\mathcal{TO}_{\text{bof}} = \sqrt{2(\delta + \phi)(\mu - 2R - D - \theta + \phi)}$$

Not same δ as in Young/Daly for coordinated checkpointing on global remote storage ☺

Waste minimization

Scenario DoubleNBL $\mathcal{F}_{\text{nbl}} = D + R + \theta + \frac{P}{2}$

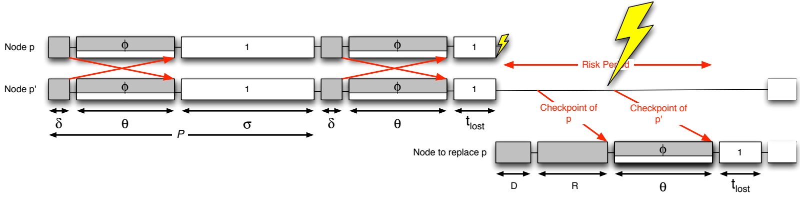
$$\mathcal{TO}_{\text{nbl}} = \sqrt{2(\delta+\phi)(\mu - R - D - \theta)}$$

Scenario DoubleBoF $\mathcal{F}_{\text{bof}} = \mathcal{F}_{\text{nbl}} + R - \phi$

$$\mathcal{TO}_{\text{bof}} = \sqrt{2(\delta+\phi)(\mu - 2R - D - \theta + \phi)}$$

Not same δ as in Young/Daly for coordinated checkpointing on global remote storage ☺

Risk



Application at risk until complete reception of both messages:

- Risk $= D + R + \theta$ for DOUBLENBL
- Risk $= D + 2R$ for DOUBLEBOF

Analysis:

- Failures strike with uniform distribution over time
- $\lambda = \frac{1}{n\mu}$ instantaneous processor failure rate

Success probability $\mathbb{P}_{\text{double}} = (1 - 2\lambda^2 T\text{Risk})^{n/2}$

Risk

Consider a pair made of one processor and its buddy:

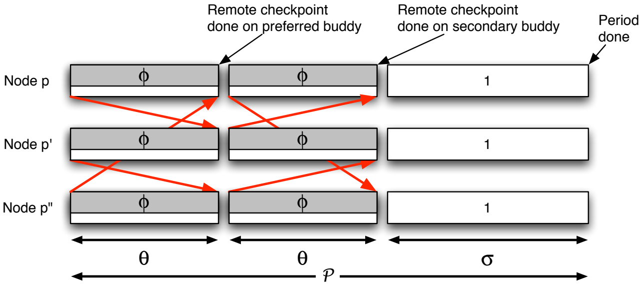
- Probability of first processor failing: λT,
- Probability of one failure in the pair : $1 - (1 - \lambda T)^2 \approx 2\lambda T$
- Probability of second failure within risk period: λRisk
- Probability of fatal failure in the pair: $(2\lambda T)(\lambda \text{Risk})$
- Probability of application fatal failure: $1 - (1 - 2\lambda^2 T \text{Risk})^{n/2}$

Success probability $\mathbb{P}_{\text{double}} = (1 - 2\lambda^2 T \text{Risk})^{n/2}$

compare to $\mathbb{P}_{\text{base}} = (1 - \lambda T_{\text{base}})^n$

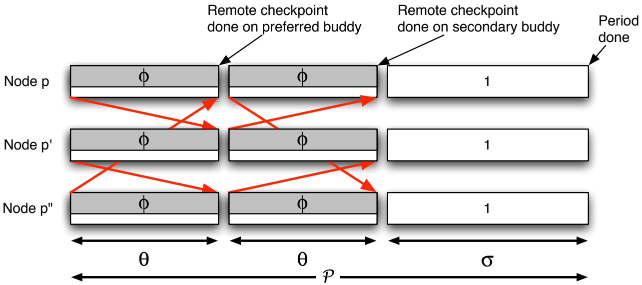
Outline

Principle



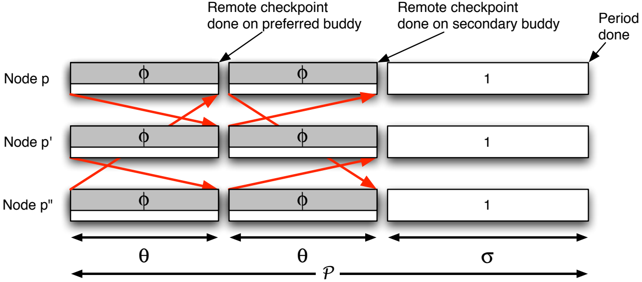
- Processors organized in triples
- Each processor has a preferred buddy and a secondary buddy
- Rotation of buddies

Principle



- Waste in fault-free execution tends to zero
- Application failure = three successive failures within a triple
  $\Rightarrow$ Smaller risk even for large θ
- Only need non-blocking version TRIPLE

Memory requirement



- Copy-on-write for local checkpoint file
- Same memory usage as double checkpointing algorithm

Analysis

Waste

- $\textsc{Waste}_{\mathsf{fail}}$ same as for $\textsc{DoubleNBL}$
- $\textsc{Waste}_{\mathsf{ff}} = \frac{2\phi}{P}$ instead of $\textsc{Waste}_{\mathsf{ff}} = \frac{\delta+\phi}{P}$ for $\textsc{DoubleNBL}$

Risk

- $\mathsf{Risk} = D + R + 2\theta$
- **Success probability** $\mathbb{P}_{\mathsf{triple}} = (1 - 6\lambda^3 T \mathsf{Risk}^2)^{n/3}$

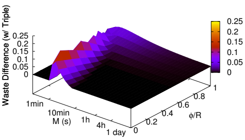
Outline

Scenarios

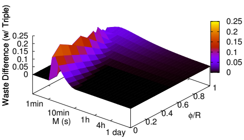
Scenario	D	δ	ϕ	R	α	n
Base	0	2	$0 \leq \phi \leq 4$	4	10	324×32
Exa	60	30	$0 \leq \phi \leq 60$	60	10	10^6

Exa corresponds to the Exa-Slim scenario

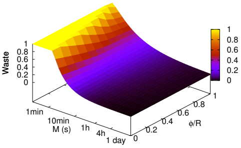
Waste for scenario *Base*



DoubleBoF



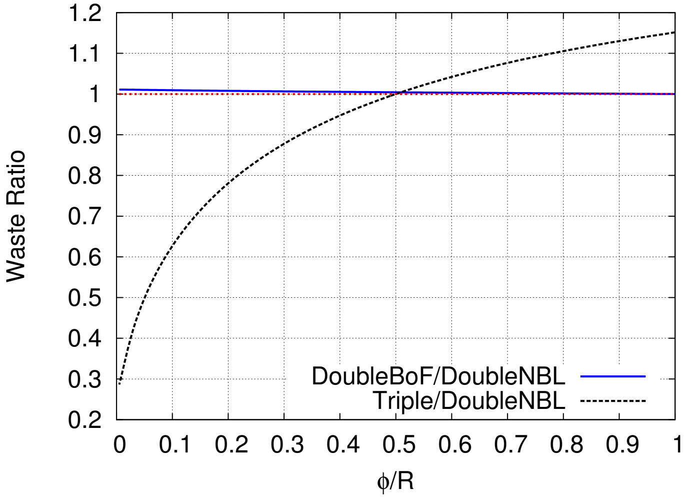
DoubleNBL



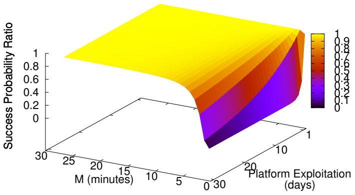
Triple

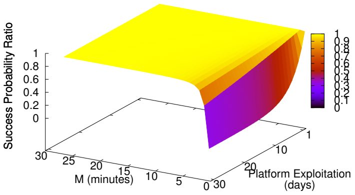
Waste as a function of ϕ/R and μ

Waste for scenario *Base* ($\mu = 7h$)



Success probability for scenario *Base*

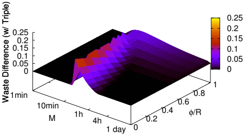




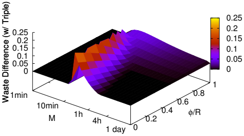
Ratio DoubleNBL/ DoubleBoF   Ratio DoubleBoF/ Triple

Relative success probability
function of μ and platform life T ($\theta = (\alpha + 1)R$)

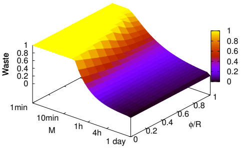
Waste for scenario *Exa*



DoubleBoF



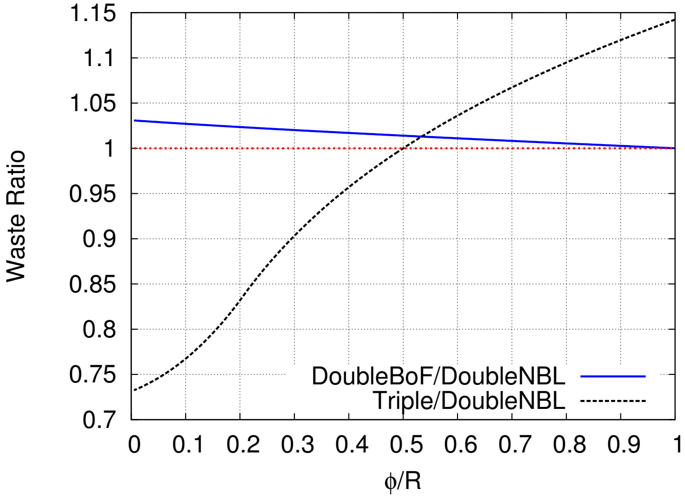
DoubleNBL



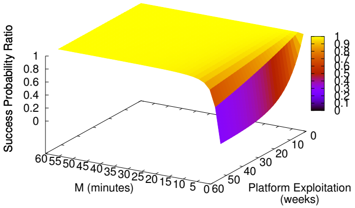
Triple

Waste as a function of ϕ/R and μ

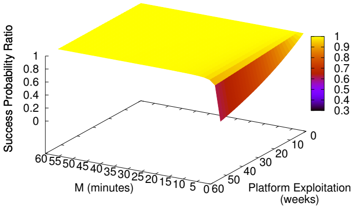
Waste for scenario *Exa* ($\mu = 7h$)



Success probability for scenario *Exa*



Ratio DOUBLENBL/ DOUBLEBOF



Ratio DOUBLEBOF/ TRIPLE

Relative success probability
function of μ and platform life T ($\theta = (\alpha + 1)R$)

Conclusion

Triple checkpointing

- Save checkpoint on two remote processes instead of one, without much more memory or storage requirements
- Excellent success probability, almost no failure-free overhead
- Assessment of performance and risk factors using unified mode
- Realistic scenarios conclude to superiority of TRIPLE

Future work

- Study real-life applications and propose refined values for α for a set of widely-used benchmarks
- Very small MTBF values on future exascale platforms
  $\Rightarrow$ combine distributed in-memory strategies
  with uncoordinated or hierarchical checkpointing protocols

Outline

Outline

Framework

Predictor

- Exact prediction dates (at least C seconds in advance)
- Recall r: fraction of faults that are predicted
- Precision p: fraction of fault predictions that are correct

Events

- *true positive*: predicted faults
- *false positive*: fault predictions that did not materialize as actual faults
- *false negative*: unpredicted faults

Fault rates

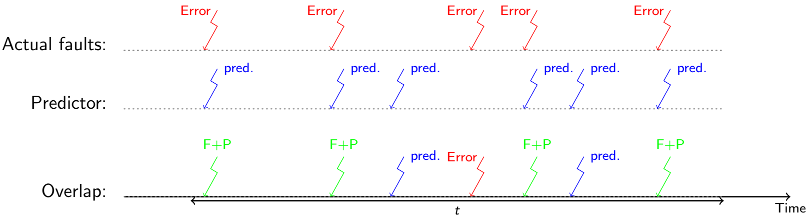
- μ: mean time between failures (MTBF)
- μ_P mean time between predicted events (both true positive and false positive)
- μ_{NP} mean time between unpredicted faults (false negative).
- μ_e: mean time between events (including three event types)

$$r = \frac{True_P}{True_P + False_N} \quad \text{and} \quad p = \frac{True_P}{True_P + False_P}$$

$$\frac{(1-r)}{\mu} = \frac{1}{\mu_{NP}} \quad \text{and} \quad \frac{r}{\mu} = \frac{p}{\mu_P}$$

$$\frac{1}{\mu_e} = \frac{1}{\mu_P} + \frac{1}{\mu_{NP}}$$

Example



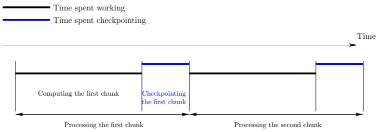
- Predictor predicts six faults in time t
- Five actual faults. One fault not predicted
- $\mu = \frac{t}{5}$, $\mu_P = \frac{t}{6}$, and $\mu_{NP} = t$
- Recall $r = \frac{4}{5}$ (green arrows over red arrows)
- Precision $p = \frac{4}{6}$ (green arrows over blue arrows)

Algorithm

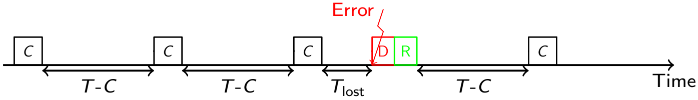
1. While no fault prediction is available:
   - checkpoints taken periodically with period T
2. When a fault is predicted at time t:
   - take a checkpoint ALAP (completion right at time t)
   - after the checkpoint, complete the execution of the period

Computing the waste

1. **Fault-free execution:** $\textsc{Waste}[FF] = \frac{C}{T}$



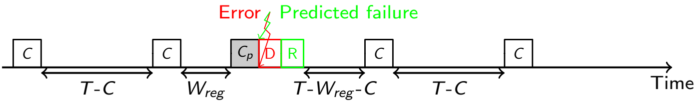
2. **Unpredicted faults:** $\frac{1}{\mu_{NP}}\left[D + R + \frac{T}{2}\right]$



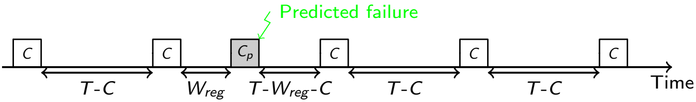
$$\textsc{Waste}[fail] = \frac{1}{\mu}\left[(1-r)\frac{T}{2} + D + R + \frac{r}{p}C\right] \Rightarrow T_{opt} \approx \sqrt{\frac{2\mu C}{1-r}}$$

Computing the waste

3. **Predictions:** $\frac{1}{\mu_P}\left[p(C+D+R)+(1-p)C\right]$



with actual fault (true positive)

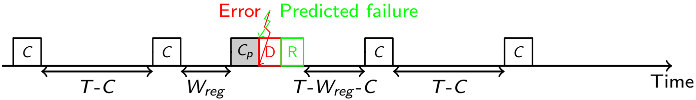


no actual fault (false negative)

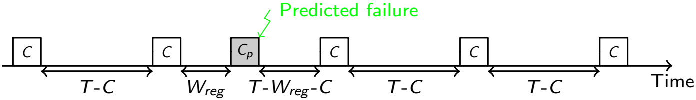
$$\textsc{Waste}[fail] = \frac{1}{\mu}\left[(1-r)\frac{T}{2} + D + R + \frac{r}{p}C\right] \Rightarrow T_{opt} \approx \sqrt{\frac{2\mu C}{1-r}}$$

Computing the waste

3. **Predictions:** $\frac{1}{\mu_P}\left[p(C+D+R)+(1-p)C\right]$



with actual fault (true positive)



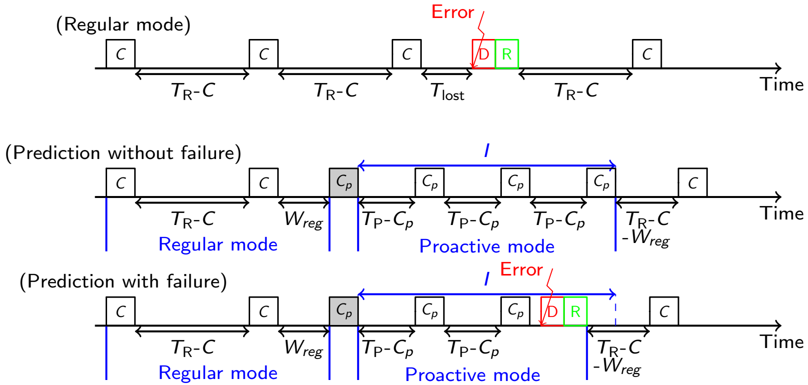
no actual fault (false negative)

$$\textsc{Waste}[fail] = \frac{1}{\mu}\left[(1-r)\frac{T}{2}+D+R+\frac{r}{p}C\right] \Rightarrow T_{opt} \approx \sqrt{\frac{2\mu C}{1-r}}$$

Refinements

- Use different value C_p for proactive checkpoints
- Avoid checkpointing too frequently for false negatives
  $\Rightarrow$ Only trust predictions with some fixed probability q
  $\Rightarrow$ Ignore predictions with probability $1 - q$
  Conclusion: trust predictor always or never ($q = 0$ or $q = 1$)
- Trust prediction depending upon position in current period
  $\Rightarrow$ Increase q when progressing
  $\Rightarrow$ Break-even point $\frac{C_p}{p}$

With prediction windows



Gets too complicated! ☹

Outline

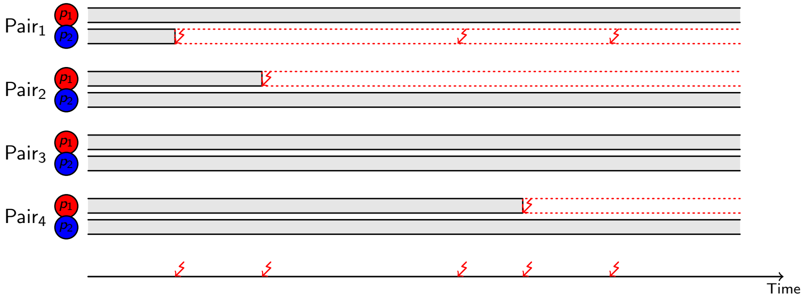
Replication

- Systematic replication: efficiency $< 50\%$
- Can replication+checkpointing be more efficient than checkpointing alone?
- Study by Ferreira et al. [SC'2011]: yes

Model by Ferreira et al. [SC' 2011]

- Parallel application comprising N processes
- Platform with $p_{total} = 2N$ processors
- Each process replicated $\rightarrow$ N *replica-groups*
- When a replica is hit by a failure, it is not restarted
- Application fails when both replicas in one replica-group have been hit by failures

Example



The birthday problem

Classical formulation

What is the probability, in a set of m people, that two of them have same birthday ?

Relevant formulation

What is the average number of people required to find a pair with same birthday?

$$Birthday(N) = 1 + \int_0^{+\infty} e^{-x}(1 + x/N)^{N-1}dx$$

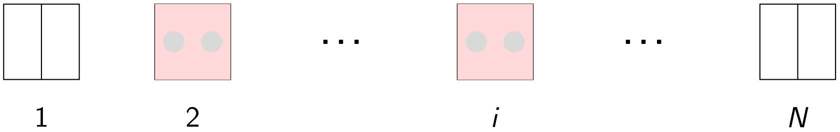
The analogy

Two people with same birthday

$\equiv$

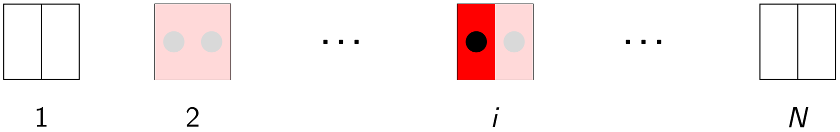
Two failures hitting same replica-group

Differences with birthday problem



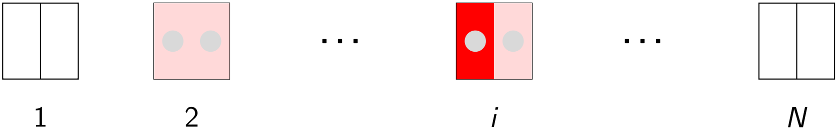
- N processes; each replicated twice
- Uniform distribution of failures
- First failure: each replica-group has probability $1/N$ to be hit
- Second failure

Differences with birthday problem



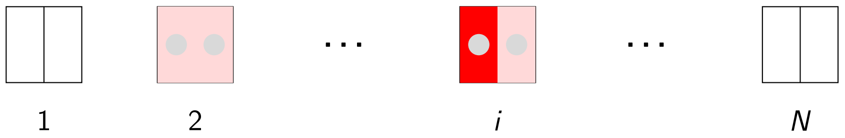
- N processes; each replicated twice
- Uniform distribution of failures
- First failure: each replica-group has probability $1/N$ to be hit
- Second failure

Differences with birthday problem



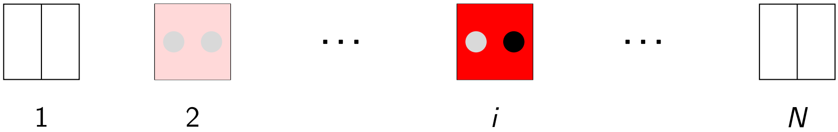
- N processes; each replicated twice
- Uniform distribution of failures
- First failure: each replica-group has probability $1/N$ to be hit
- Second failure: can failed PE be hit?

Differences with birthday problem



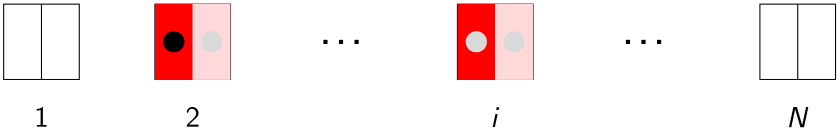
- N processes; each replicated twice
- Uniform distribution of failures
- First failure: each replica-group has probability $1/N$ to be hit
- Second failure cannot hit failed PE
  - Failure uniformly distributed over $2N-1$ PEs
  - Probability that replica-group i is hit by failure: $1/(2N-1)$
  - Probability that replica-group $\neq i$ is hit by failure: $2/(2N-1)$
  - Failure not uniformly distributed over replica-groups: this is not the birthday problem

Differences with birthday problem



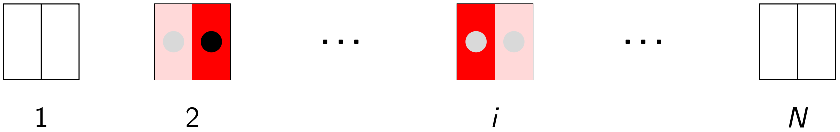
- N processes; each replicated twice
- Uniform distribution of failures
- First failure: each replica-group has probability $1/N$ to be hit
- Second failure cannot hit failed PE
    - Failure uniformly distributed over $2N-1$ PEs
    - Probability that replica-group i is hit by failure: $1/(2N-1)$
    - Probability that replica-group $\neq i$ is hit by failure: $2/(2N-1)$
    - Failure not uniformly distributed over replica-groups: this is not the birthday problem

Differences with birthday problem



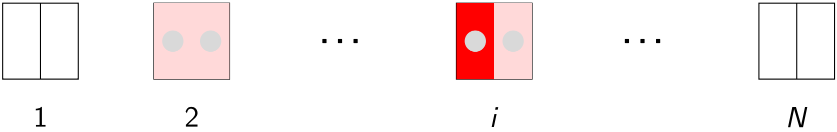
- N processes; each replicated twice
- Uniform distribution of failures
- First failure: each replica-group has probability $1/N$ to be hit
- Second failure cannot hit failed PE
  - Failure uniformly distributed over $2N - 1$ PEs
  - Probability that replica-group i is hit by failure: $1/(2N - 1)$
  - Probability that replica-group $\neq i$ is hit by failure: $2/(2N - 1)$
  - Failure not uniformly distributed over replica-groups: this is not the birthday problem

Differences with birthday problem



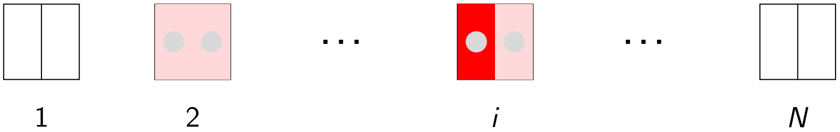
- N processes; each replicated twice
- Uniform distribution of failures
- First failure: each replica-group has probability $1/N$ to be hit
- Second failure cannot hit failed PE
  - Failure uniformly distributed over $2N-1$ PEs
  - Probability that replica-group i is hit by failure: $1/(2N-1)$
  - Probability that replica-group $\neq i$ is hit by failure: $2/(2N-1)$
  - Failure not uniformly distributed over replica-groups: this is not the birthday problem

Differences with birthday problem



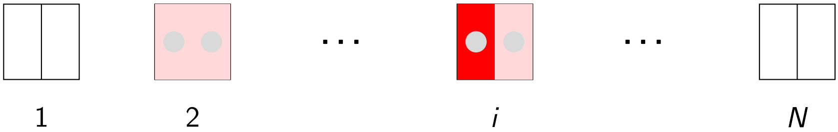
- N processes; each replicated twice
- Uniform distribution of failures
- First failure: each replica-group has probability $1/N$ to be hit
- Second failure cannot hit failed PE
  - Failure uniformly distributed over $2N - 1$ PEs
  - Probability that replica-group i is hit by failure: $1/(2N - 1)$
  - Probability that replica-group $\neq i$ is hit by failure: $2/(2N - 1)$
  - Failure not uniformly distributed over replica-groups: this is not the birthday problem

Differences with birthday problem



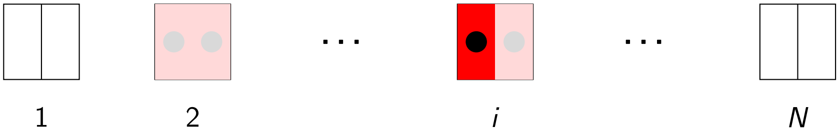
- N processes; each replicated twice
- Uniform distribution of failures
- First failure: each replica-group has probability $1/N$ to be hit
- Second failure can hit failed PE

Differences with birthday problem



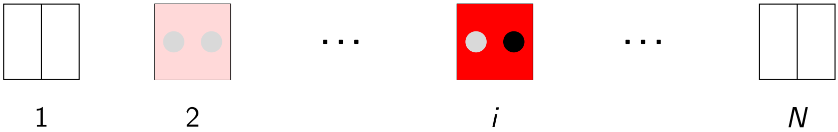
- N processes; each replicated twice
- Uniform distribution of failures
- First failure: each replica-group has probability $1/N$ to be hit
- Second failure can hit failed PE
  - Suppose failure hits replica-group i
  - If failure hits failed PE: application survives
  - If failure hits running PE: application killed
  - Not all failures hitting the same replica-group are equal: this is not the birthday problem

Differences with birthday problem



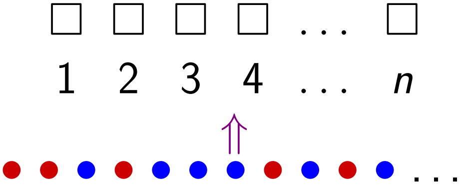
- N processes; each replicated twice
- Uniform distribution of failures
- First failure: each replica-group has probability $1/N$ to be hit
- Second failure can hit failed PE
  - Suppose failure hits replica-group i
  - If failure hits failed PE: application survives
  - If failure hits running PE: application killed
  - Not all failures hitting the same replica-group are equal: this is not the birthday problem

Differences with birthday problem



- N processes; each replicated twice
- Uniform distribution of failures
- First failure: each replica-group has probability $1/N$ to be hit
- Second failure can hit failed PE
  - Suppose failure hits replica-group i
  - If failure hits failed PE: application survives
  - If failure hits running PE: application killed
  - Not all failures hitting the same replica-group are equal: this is not the birthday problem

Correct analogy



N bins, red and blue balls

Mean Number of Failures to Interruption (bring down application)
$MNFTI$ = expected number of balls to throw
until one bin gets one ball of each color

Exponential failures

Theorem: $MNFTI = \mathbb{E}(NFTI|0)$ where

$$\mathbb{E}(NFTI|n_f) = \begin{cases} 2 & \text{if } n_f = N, \\ \frac{2N}{2N-n_f} + \frac{2N-2n_f}{2N-n_f}\,\mathbb{E}\left(NFTI|n_f+1\right) & \text{otherwise.} \end{cases}$$

$\mathbb{E}(NFTI|n_f)$: expectation of number of failures to kill application, knowing that

- application is still running
- failures have already hit n_f different replica-groups

How do we obtain this result?