Revisiting the double checkpointing algorithm a.k.a the buddy algorithm

Jack Dongarra[1], Thomas Hérault[1] and Yves Robert[1,2]

1. University of Tennessee Knoxville, USA
2. Ecole Normale Supérieure de Lyon & INRIA, France

{dongarra|herault}@eecs.utk.edu, yves.robert@ens-lyon.fr

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

ICL Lunch Talk

Motivation

buddy pronounce

buddy ☆ ['bʌdɪ]

1 noun forms

US ▸ **copain** m, ▸ **pote** ☆☆ m

esp US; [of Aids sufferer] ▸ **buddy** mf (bénévole accompagnant une personne atteinte du sida)

- hi there, buddy! : salut, mon pote ! ☆☆
- buddy movie or film : *film qui raconte l'histoire de deux amis*

2 compounds

◆ **buddy-buddy** ☆ adjective esp US

- Paul and Mark are very buddy-buddy, Paul is very buddy-buddy with Mark : Paul et Mark sont très copains or copains comme cochons
- a buddy-buddy movie : *un film qui a pour héros deux amis*

Motivation

buddy

410 up, **252** down

1: a good friend that you are confortable being around and sharing things with.

2: a condescending term used sarcastically to describe someone that you consider below yourself.

"I'm going to have some beers with my buddy tonight."

"Buddy just told me I can't park my car here."

Motivation

buddy

277 up, **160** down

A nice word that men use in presenting some sort of emotional affection towards other men.

Man #1: I'm really scared.

Man #2: Everything is going to be okay, buddy.

Motivation

buddy

7 up, **4** down

a douchebag, asshole, or someone else that annoys you/gets on your nerves. Usually another driver on the road.

"Move it, buddy!"

Motivation

- Checkpoint transfer and storage
  $\Rightarrow$ critical issues of rollback/recovery protocols

- Stable storage: high cost

- Distributed in-memory storage:
  ☺ Much better scalability
  ☹ Risk of non-recoverable failures

Outline

Outline

Main idea

- Store checkpoints in local memory $\Rightarrow$ no centralized storage
- Replicate checkpoints $\Rightarrow$ application survives single failure
- Still, risk of fatal failure in some (unlikely) scenarios

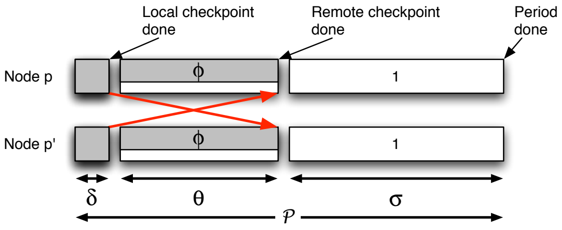
Double checkpoint algorithm

- 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

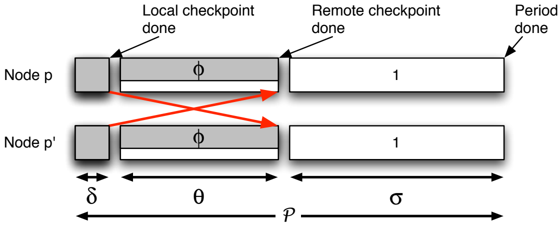
- blocking version by Zheng, Shi and Kalé
- non-blocking version by Ni, Meneses and Kalé

Non-blocking checkpoint algorithm



- Checkpoints taken periodically, with period $\mathcal{P} = \delta + \theta + \sigma$
- 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

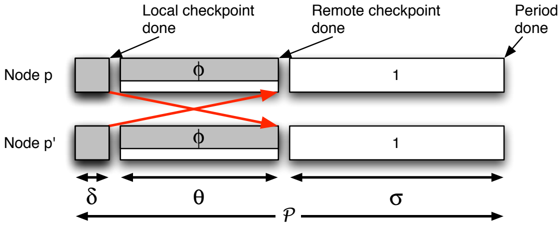
Non-blocking checkpoint algorithm



Work in failure-free period:

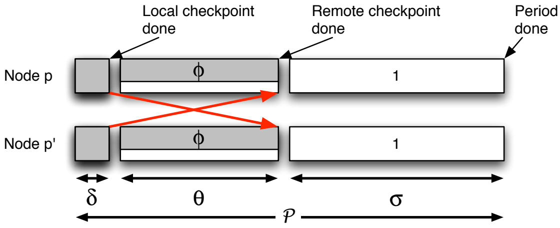
$$W = (\theta - \phi) + \sigma = \mathcal{P} - \delta - \phi$$

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 $\theta_{\text{min}} = R$
- 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 $\mathcal{P} \Rightarrow W = \mathcal{P} - \delta - \phi$ work units
  - $T_{\text{ff}} = \frac{\mathcal{P}}{W} T_{\text{base}}$
  - $\left(1 - \frac{\delta+\phi}{\mathcal{P}}\right) T_{\text{ff}} = T_{\text{base}}$

Computing the waste

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

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

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

Computing the waste

$$\left(1 - \textsc{Waste}\right) T = T_{\mathsf{base}}$$

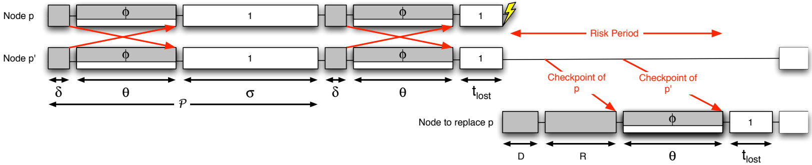
$$\textsc{Waste} = 1 - \left(1 - \frac{\mathcal{F}}{M}\right)\left(1 - \frac{\delta + \phi}{\mathcal{P}}\right)$$

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

$$\textsc{Waste} = \textsc{Waste}_{\mathsf{fail}} + \textsc{Waste}_{\mathsf{ff}} - \textsc{Waste}_{\mathsf{fail}}\textsc{Waste}_{\mathsf{ff}}$$

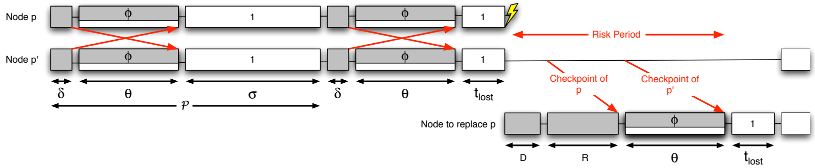
Time lost due to failures

Scenario DoubleNBL



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

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

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

Scenario DOUBLEBOF $\mathcal{F}_{\mathsf{bof}} = \mathcal{F}_{\mathsf{nbl}} + R - \phi$

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

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

Waste minimization

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

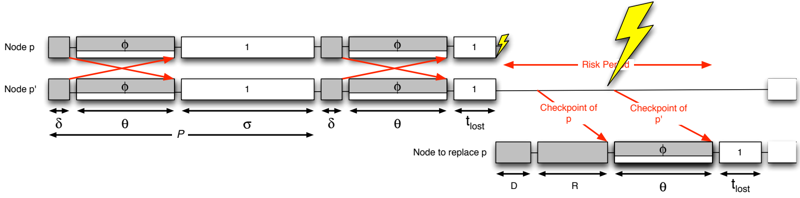
$$\mathcal{TO}_{\text{nbl}} = \sqrt{2(\delta + \phi)(M - R - D - \theta)}$$

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

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

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
- $\lambda = \frac{1}{nM}$ 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:

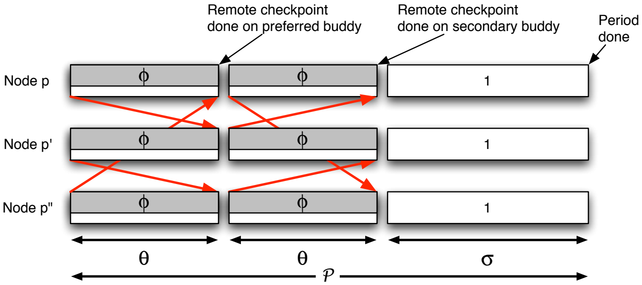
- Probability of first processor failing: λT,
- Probability of one failure in the pair : $1-(1-\lambda T)^2 \approx 2\lambda T$
- Probability of second failure within risk period: λRisk
- Probability of fatal failure in the pair: $(2\lambda T)(\lambda \text{Risk})$
- Probability of application fatal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

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

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

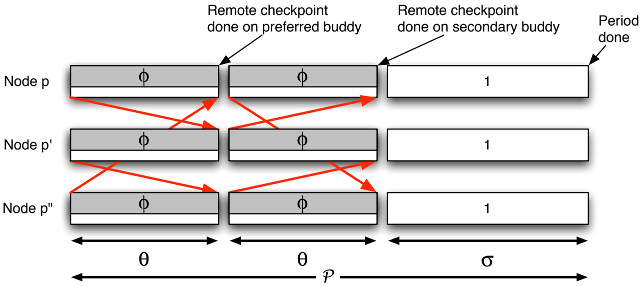
Outline

Principle



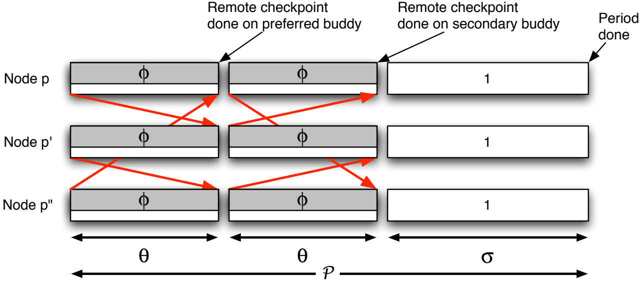
- Processors organized in triples
- Each processor has a preferred buddy and a secondary buddy
- Rotation of buddies

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

Risk

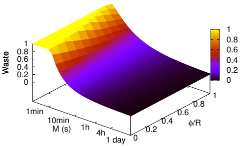
- $\mathsf{Risk} = D + R + 2\theta$
- **Success probability** $\mathbb{P}_{\mathsf{triple}} = (1 - 6\lambda^3 T \mathsf{Risk}^2)^{n/3}$

Outline

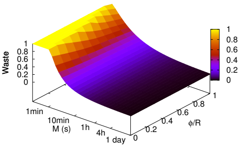
Scenarios

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

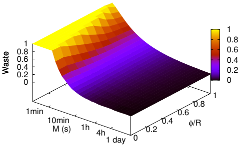
Waste for scenario *Base*



DOUBLEBOF



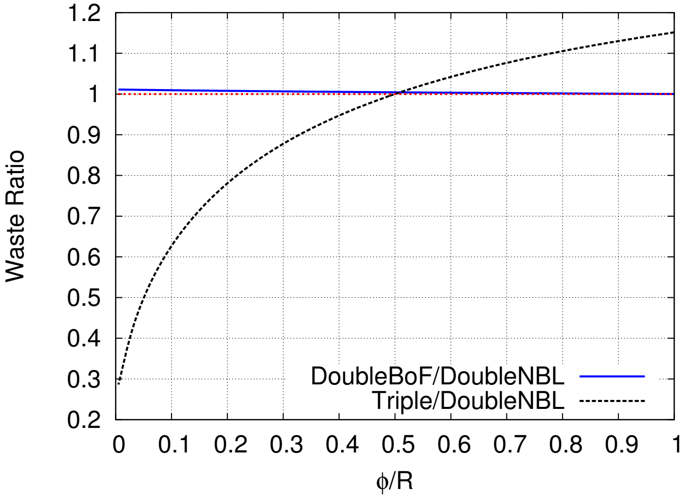
DOUBLENBL



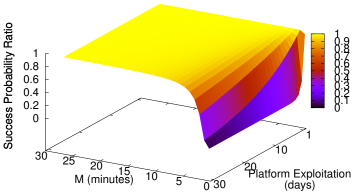
TRIPLE

Waste as a function of ϕ/R and M

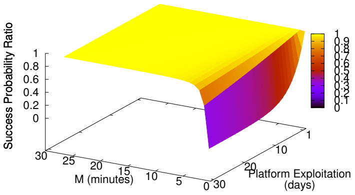
Waste for scenario *Base* ($M = 7h$)



Success probability for scenario *Base*



Ratio DOUBLENBL/ DOUBLEBOF

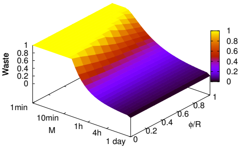


Ratio DOUBLEBOF/ TRIPLE

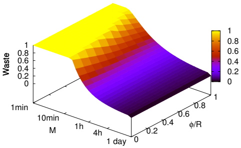
Relative success probability

function of M and platform life T ($\theta = (\alpha + 1)R$)

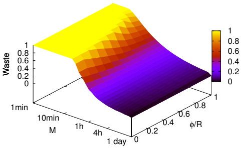
Waste for scenario *Exa*



DOUBLEBOF



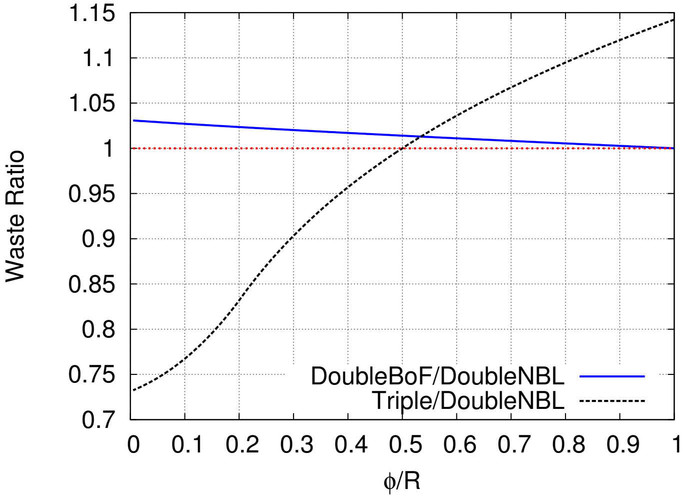
DOUBLENBL



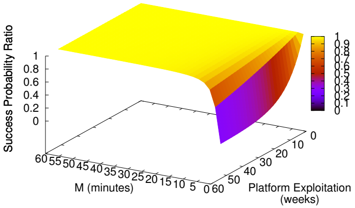
TRIPLE

Waste as a function of ϕ/R and M

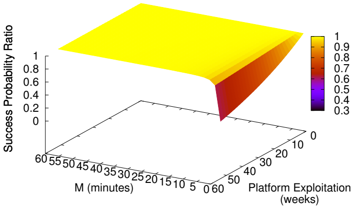
Waste for scenario *Exa* ($M = 7h$)



Success probability for scenario *Exa*



Ratio DOUBLENBL/ DOUBLEBOF



Ratio DOUBLEBOF/ TRIPLE

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

Conclusion

Double checkpointing

- Revisiting algorithms by Zheng, Shi and Kalé and by Ni, Meneses and Kalé
- New version DOUBLEBOF: reduce risk duration, at the cost of increasing failure overhead
- New parameter α for transfer cost overlap
- Unified model for performance/risk bi-criteria assessment

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

Conclusion

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

Tribute to Buddy Knox

https://www.youtube.com/watch?v=yGrVNpFJbLI