Models for fault-tolerance at very large scale

Yves Robert

ENS Lyon & Institut Universitaire de France
University of Tennessee Knoxville

yves.robert@ens-lyon.fr

http://graal.ens-lyon.fr/~yrobert/espas2013.pdf

ESPAS 2013

Thanks

INRIA & ENS Lyon

- Frédéric Vivien
- PhD students (Guillaume Aupy, Dounia Zaidouni)

UT Knoxville

- George Bosilca
- Aurélien Bouteiller
- Jack Dongarra
- Thomas Hérault

Others

- Franck Cappello, UIUC-Inria joint lab
- Henri Casanova, Univ. Hawai'i
- Amina Guermouche, Univ. Versailles Paris

Outline

Outline

Outline

Exascale platforms (courtesy Jack Dongarra)

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

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

Exascale platforms

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

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

More nodes $\Rightarrow$ Shorter MTBF (Mean Time Between Failures)

Exascale platforms

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

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

Exascale
$\neq$ Petascale $\times 1000$

More nodes = (between failures)

Scenario for 2015

- Phase-Change memory, read 100GB/sec, write 10GB/sec
- Checkpoint size 128GB
- C: checkpoint save time: $C = 12sec$
- R: checkpoint recovery time: $R = 1.2sec$
- D: down/reboot time: $D = 15sec$
- **Steady-state** job utilization of whole platform
- p: total number of (multicore) nodes: $p = 2^8$ to $p = 2^{20}$
- MTBF $\mu = 1$ week, 1 month, 1|10|100|1000 years (per node)

Platform throughput with optimal checkpointing period

μ =1 week	p	Throughput
	2^8	91.56%
	2^{11}	73.75%
	2^{14}	20.07%
	2^{17}	2.51%
	2^{20}	0.31%

μ =1 month	p	Throughput
	2^8	96.04%
	2^{11}	88.23%
	2^{14}	62.28%
	2^{17}	10.66%
	2^{20}	1.33%

μ =1 year	p	Throughput
	2^8	98.89%
	2^{11}	96.80%
	2^{14}	90.59%
	2^{17}	70.46%
	2^{20}	15.96%

μ =10 years	p	Throughput
	2^8	99.65%
	2^{11}	99.00%
	2^{14}	97.15%
	2^{17}	91.63%
	2^{20}	74.01%

μ =100 years	p	Throughput
	2^8	99.89%
	2^{11}	99.69%
	2^{14}	99.11%
	2^{17}	97.45%
	2^{20}	92.56%

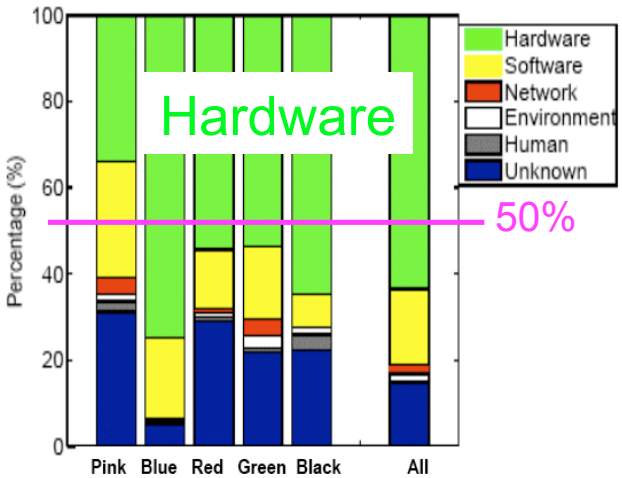
μ =1000 years	p	Throughput
	2^8	99.97%
	2^{11}	99.90%
	2^{14}	99.72%
	2^{17}	99.20%
	2^{20}	97.73%

Outline

Error sources (courtesy Franck Cappello)

Sources of failures

- Analysis of error and failure logs
- In 2005 (Ph. D. of CHARNG-DA LU) : "Software halts account for the most number of outages (59-84 percent), and take the shortest time to repair (0.6-1.5 hours). Hardware problems, albeit rarer, need 6.3-100.7 hours to solve."
- In 2007 (Garth Gibson, ICPP Keynote):



Relative frequency of root cause by system type.

- In 2008 (Oliner and J. Stearley, DSN Conf.):

Type	Raw		Filtered	
	Count	%	Count	%
Hardware	174,586,516	98.04	1,999	18.78
Software	144,899	0.08	6,814	64.01
Indeterminate	3,350,044	1.88	1,832	17.21

Software errors: Applications, OS bug (kernel panic), communication libs, File system error and other.

Hardware errors, Disks, processors, memory, network

Conclusion: Both Hardware and Software failures have to be considered

Faults and failures

A few definitions

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

Failure distributionss

- Exponential (memoryless)
- Weibull (account for infant mortality)

Failure distributions: with several processors

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

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

- Well, it depends whether with or without rejuvenation ☹

Failure distributions: with several processors

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

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

- Well, it depends whether with or without rejuvenation ☹

Without rejuvenation

- Rebooting only faulty processor
- Platform failure distribution
  $\Rightarrow$ superposition of p IID processor distributions
- Simple formula for arbitrary distributions:

$$\mu_p = \frac{\mu}{p}$$

  with p processors of MTBF μ
- Rejuvenation does not matter for Exponential
- Rejuvenation harmful for Weibull with $k < 1$

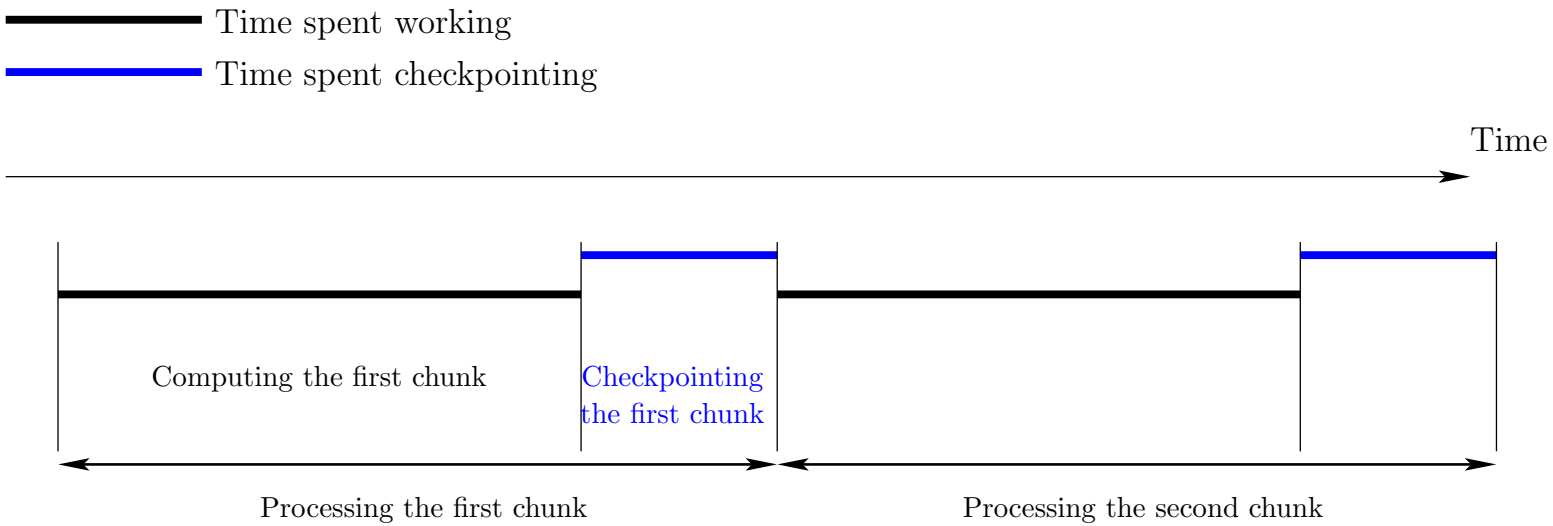
Values from the literature

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

Outline

Outline

Checkpointing cost



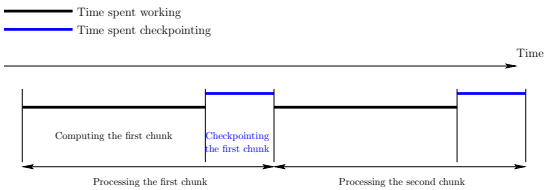
Blocking model: while a checkpoint is taken, no computation can be performed

Framework

- Periodic checkpointing policy of period T
- Independent and identically distributed failures
- Applies to a single processor with MTBF $\mu = \mu_{ind}$
- Applies to a platform with p processors with MTBF $\mu = \frac{\mu_{ind}}{p}$
  - coordinated checkpointing
  - tightly-coupled application
  - progress $\Leftrightarrow$ all processors available

Waste: fraction of time not spent for useful computations

Waste in fault-free execution



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

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

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

$$\text{TIME}_{\text{FF}} = \text{TIME}_{\text{base}} \frac{T}{T - C} \text{ and } \text{WASTE}[FF] = \frac{\text{TIME}_{\text{FF}} - \text{TIME}_{\text{base}}}{\text{TIME}_{\text{FF}}}$$

$$\text{WASTE}[FF] = \frac{C}{T}$$

Waste due to failures

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

$$\text{TIME}_{\text{final}} = \text{TIME}_{\text{FF}} + N_{faults} \times T_{\text{lost}}$$

N_{faults} number of failures during execution
T_{lost}: average time lost par failures

$$N_{faults} = \frac{\text{TIME}_{\text{final}}}{\mu}$$

T_{lost}?

Waste due to failures

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

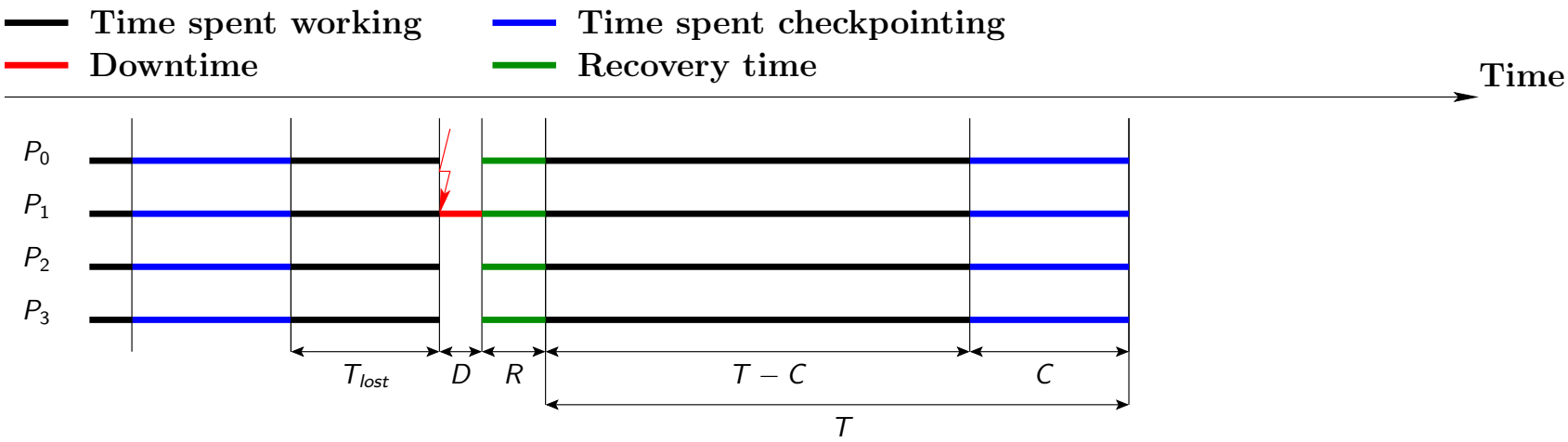
$$\text{TIME}_{\text{final}} = \text{TIME}_{\text{FF}} + N_{faults} \times T_{\text{lost}}$$

N_{faults} number of failures during execution
T_{lost}: average time lost par failures

$$N_{faults} = \frac{\text{TIME}_{\text{final}}}{\mu}$$

T_{lost}?

Computing T_{lost}



$$T_{\text{lost}} = D + R + \frac{T}{2}$$

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

$\Rightarrow$ Valid for all distribution laws, regardless of their particular shape

Waste due to failures

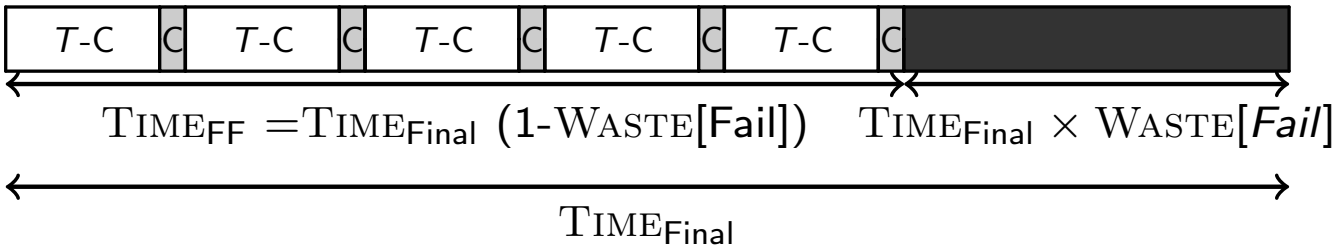
$$\text{TIME}_{\text{final}} = \text{TIME}_{\text{FF}} + N_{faults} \times T_{\text{lost}}$$

$$\text{TIME}_{\text{final}} = \text{TIME}_{\text{FF}} + \frac{\text{TIME}_{\text{final}}}{\mu} \times \left(D + R + \frac{T}{2} \right)$$

$$\text{WASTE}[fail] = \frac{\text{TIME}_{\text{final}} - \text{TIME}_{\text{FF}}}{\text{TIME}_{\text{final}}}$$

$$\text{WASTE}[fail] = \frac{1}{\mu} \left(D + R + \frac{T}{2} \right)$$

Total waste



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

$$(1 - \text{WASTE}[\mathit{fail}])(1 - \text{WASTE}[\mathit{FF}])\text{TIME}_{\text{final}} = \mathit{Time}[\mathit{base}]$$

$$1 - \text{WASTE} = (1 - \text{WASTE}[\mathit{FF}])(1 - \text{WASTE}[\mathit{fail}])$$

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

Waste minimization

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

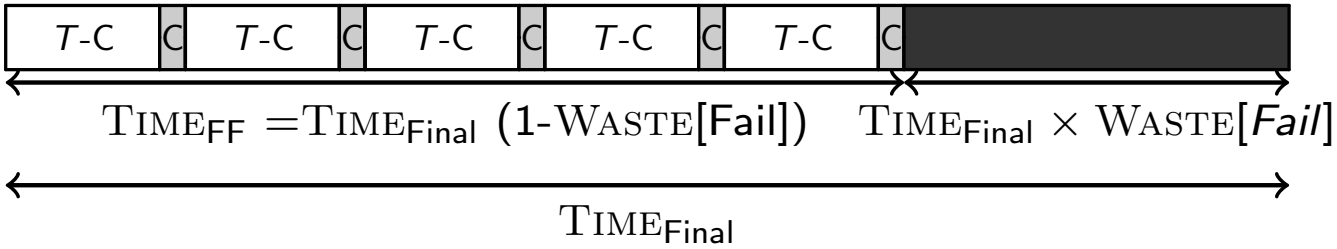
$$\text{WASTE} = \frac{u}{T} + v + wT$$

$$u = C\left(1 - \frac{D+R}{\mu}\right) \qquad v = \frac{D + R - C/2}{\mu} \qquad w = \frac{1}{2\mu}$$

WASTE minimized for $T = \sqrt{\frac{u}{w}}$

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

Comparison with Young/Daly



$$\big(1 - \textsc{Waste}[\mathit{fail}]\big)\,\textsc{Time}_{\mathsf{final}} = \textsc{Time}_{\mathsf{FF}}$$
$$\Rightarrow T = \sqrt{2(\mu - (D+R))C}$$

Daly: $\textsc{Time}_{\mathsf{final}} = \big(1 + \textsc{Waste}[\mathit{fail}]\big)\,\textsc{Time}_{\mathsf{FF}}$
$$\Rightarrow T = \sqrt{2(\mu + (D+R))C} + C$$

Young: $\textsc{Time}_{\mathsf{final}} = \big(1 + \textsc{Waste}[\mathit{fail}]\big)\,\textsc{Time}_{\mathsf{FF}}$ and $D = R = 0$
$$\Rightarrow T = \sqrt{2\mu C} + C$$

Validity of the approach (1/3)

Technicalities

- $\mathbb{E}(N_{faults}) = \frac{\text{TIME}_{\text{final}}}{\mu}$ and $\mathbb{E}(T_{\text{lost}}) = D + R + \frac{T}{2}$
  but expectation of product is not product of expectations
  (not independent RVs here)

- Enforce $C \leq T$ to get $\text{WASTE}[FF] \leq 1$

- Enforce $D + R \leq \mu$ and bound T to get $\text{WASTE}[fail] \leq 1$
  but $\mu = \frac{\mu_{ind}}{p}$ too small for large p, regardless of μ_{ind}

Validity of the approach (2/3)

Several failures within same period?

- WASTE[fail] accurate only when two or more faults do not take place within same period
- Cap period: $T \leq \gamma\mu$, where γ is some tuning parameter
  - Poisson process of parameter $\theta = \frac{T}{\mu}$
  - Probability of having $k \geq 0$ failures : $P(X = k) = \frac{\theta^k}{k!}e^{-\theta}$
  - Probability of having two or more failures: $\pi = P(X \geq 2) = 1 - (P(X = 0) + P(X = 1)) = 1 - (1 + \theta)e^{-\theta}$
  - $\gamma = 0.27 \Rightarrow \pi \leq 0.03$
    $\Rightarrow$ overlapping faults for only 3% of checkpointing segments

Validity of the approach (3/3)

- Enforce $T \leq \gamma\mu$, $C \leq \gamma\mu$, and $D + R \leq \gamma\mu$
- Optimal period $\sqrt{2(\mu - (D + R))C}$ may not belong to admissible interval $[C, \gamma\mu]$
- Waste is then minimized for one of the bounds of this admissible interval (by convexity)

Wrap up

- Capping periods, and enforcing a lower bound on MTBF
  $\Rightarrow$ mandatory for mathematical rigor ☹
  $\Rightarrow$ optimal strategy only known for Exp. distributions ☹

- Not needed for practical purposes ☺
  - actual job execution uses optimal value
  - account for multiple faults by re-executing work until success

- Approach surprisingly robust ☺

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

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

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 all three event types)

$$\frac{(1-r)}{\mu} = \frac{1}{\mu_{NP}}$$

$$\frac{r}{\mu} = \frac{p}{\mu_P}$$

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

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:** $\text{WASTE}[FF] = \frac{C}{T}$
2. **Unpredicted faults:** $\frac{1}{\mu_{NP}} \left[D + R + \frac{T}{2} \right]$
3. **Predictions:** $\frac{1}{\mu_P} [p(C + D + R) + (1 - p)C]$

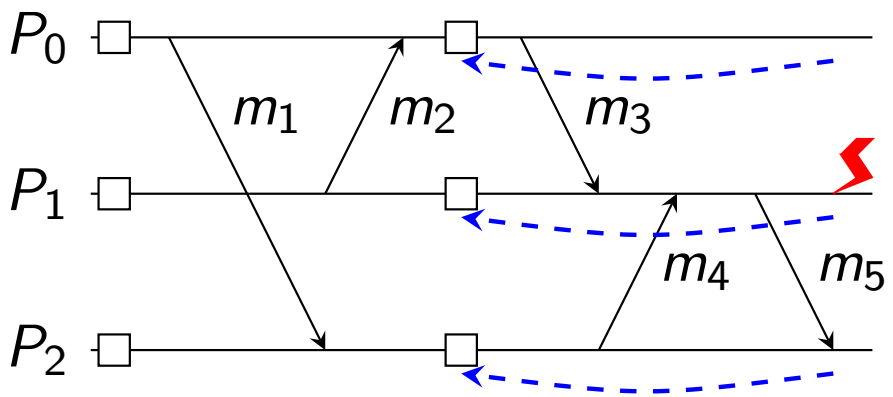
$$\text{WASTE}[fail] = \frac{1}{\mu} \left[(1 - r)\frac{T}{2} + D + R + \frac{r}{p}C \right]$$

$$T_{opt} \approx \sqrt{\frac{2\mu C}{1 - r}}$$

Outline

Background: coordinated checkpointing protocols

- Coordinated checkpoints over all processes
- Global restart after a failure



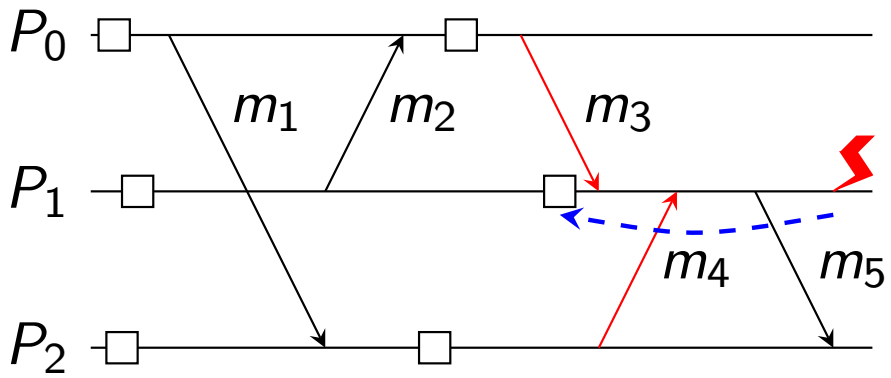
☺ No risk of cascading rollbacks

☺ No need to log messages

☹ All processors need to roll back

Background: message logging protocols

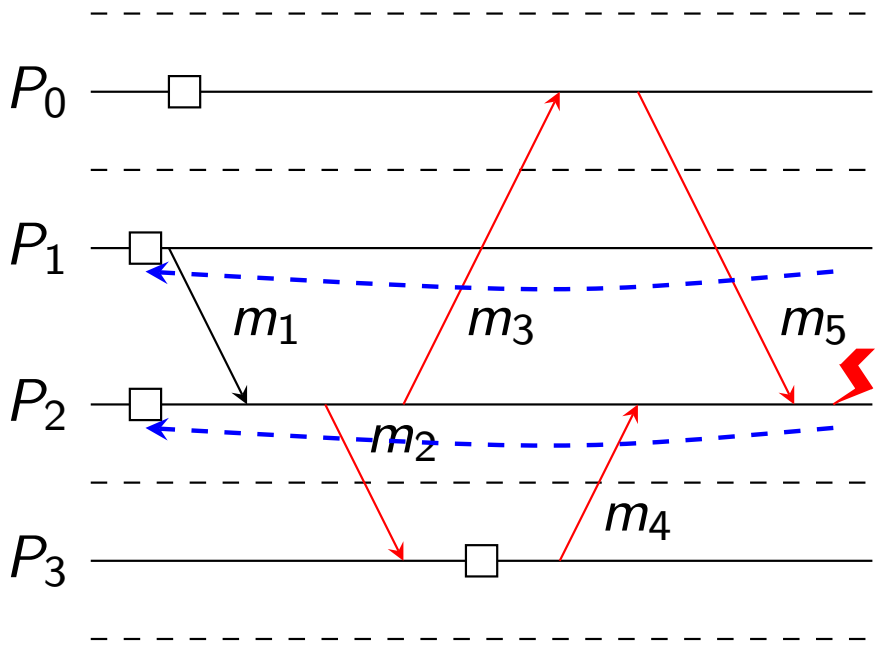
- Message content logging (sender memory)
- Restart of the failed process



- ☺ No cascading rollbacks
- ☺ Number of processes to roll back
- ☹ Memory occupation
- ☹ Overhead

Background: hierarchical protocols

- Clusters of processes
- Coordinated checkpointing protocol within clusters
- Message logging protocols between clusters
- Only processors from failed group need to roll back



☹ Need to log inter-groups messages
- Slowdowns failure-free execution
- Increases checkpoint size/time

☺ Faster re-execution with logged messages

Which checkpointing protocol to use?

Coordinated checkpointing

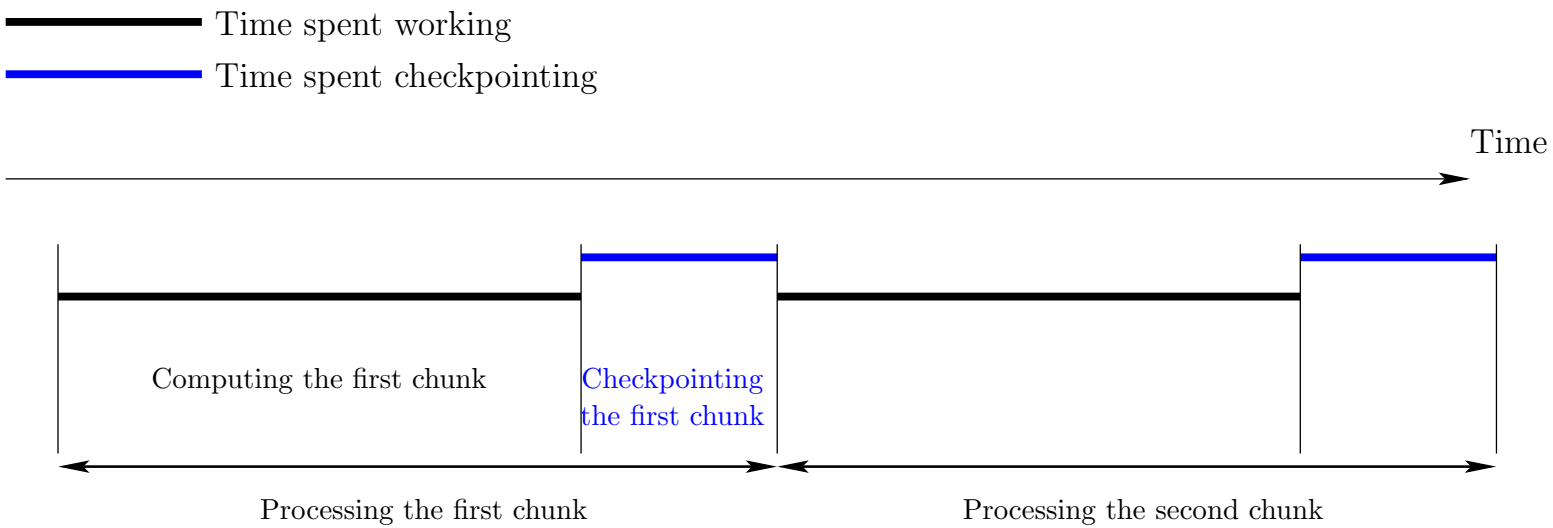
- ☺ No risk of cascading rollbacks
- ☺ No need to log messages
- ☹ All processors need to roll back
- ☹ Rumor: May not scale to very large platforms

Hierarchical checkpointing

- ☹ Need to log inter-groups messages
  - Slowdowns failure-free execution
  - Increases checkpoint size/time
- ☺ Only processors from failed group need to roll back
- ☺ Faster re-execution with logged messages
- ☺ Rumor: Should scale to very large platforms

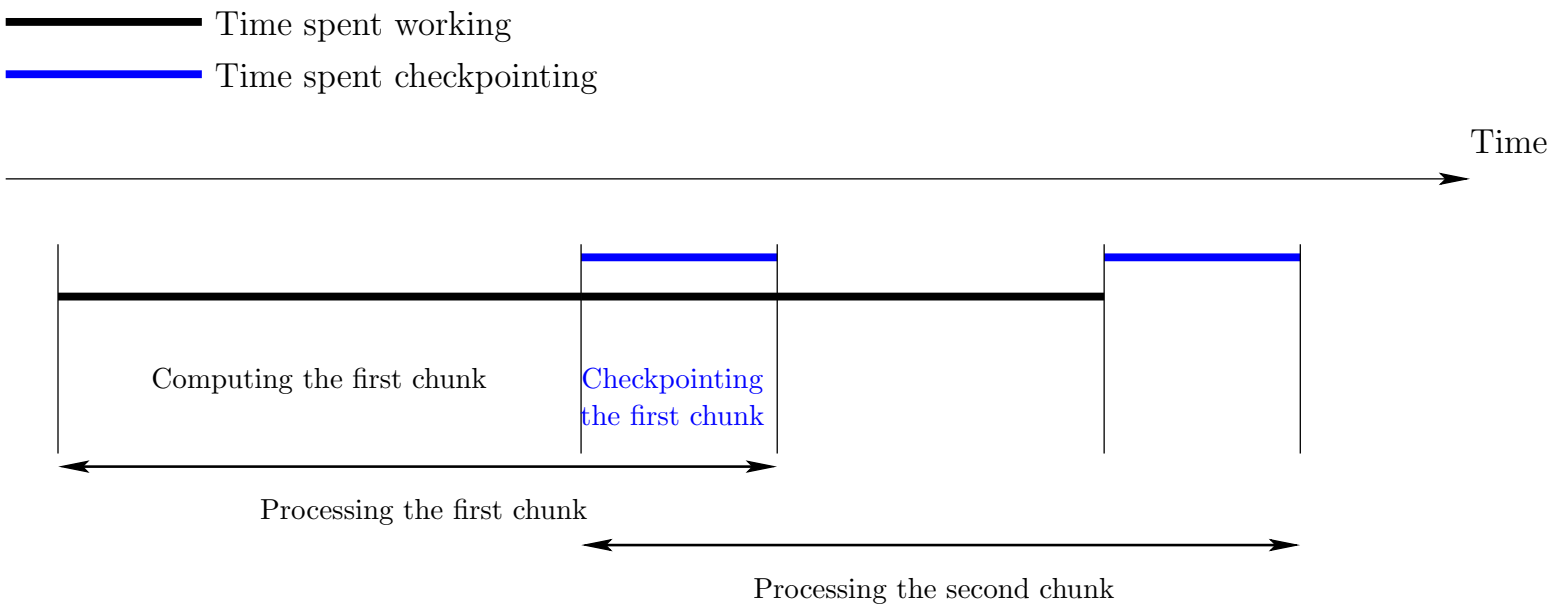
Outline

Checkpointing cost



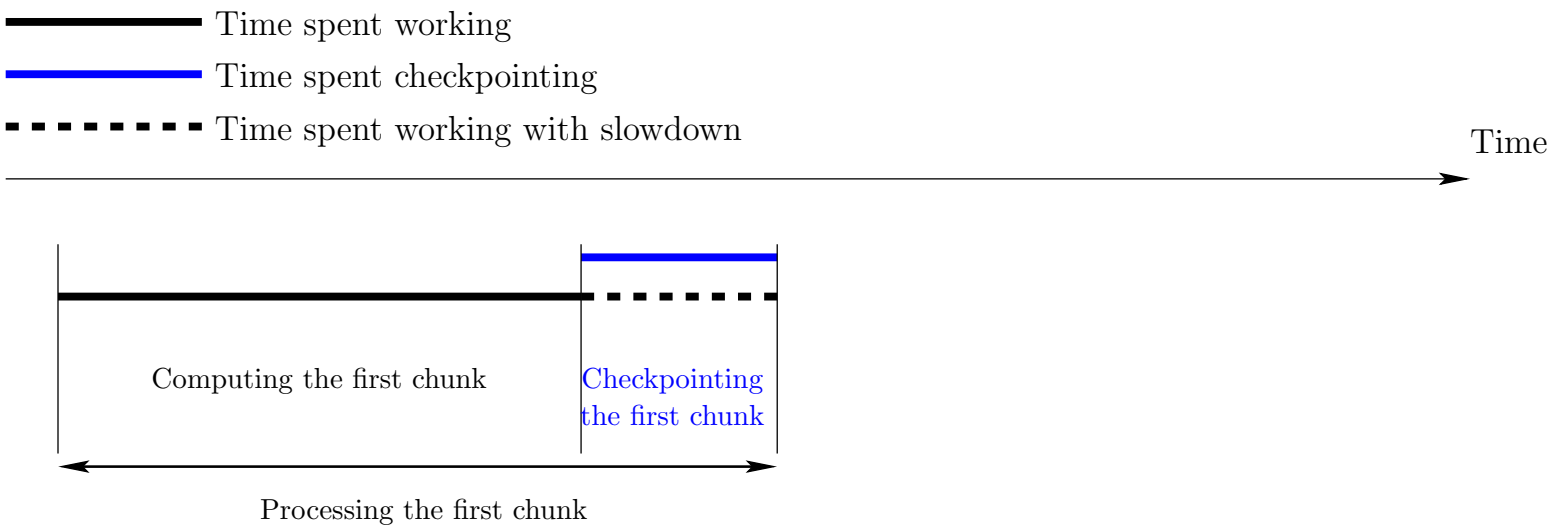
Blocking model: checkpointing blocks all computations

Checkpointing cost



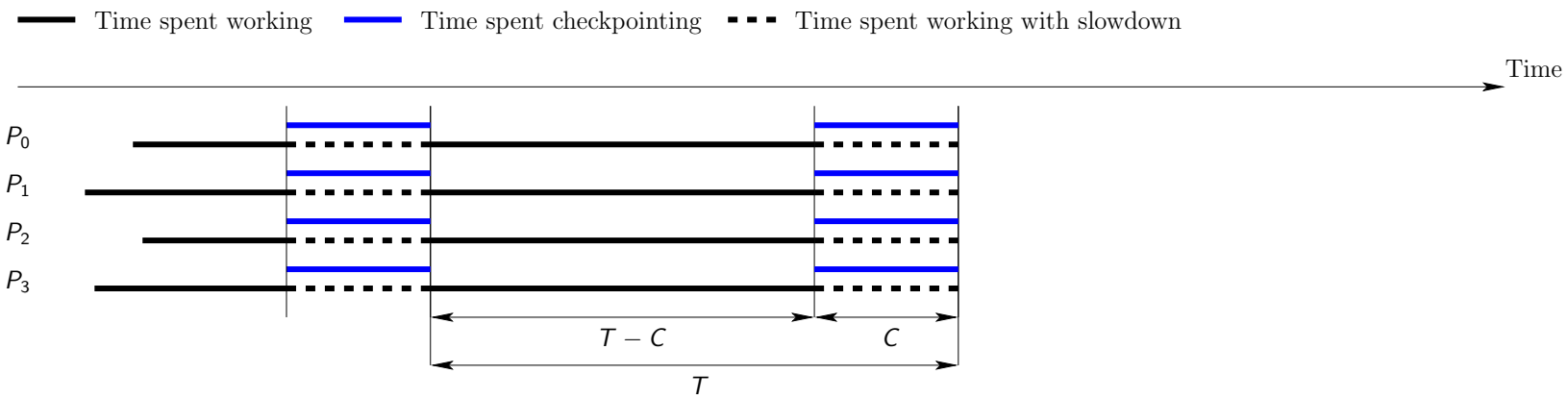
Non-blocking model: checkpointing has no impact on computations (e.g., first copy state to RAM, then copy RAM to disk)

Checkpointing cost



General model: checkpointing slows computations down: during a checkpoint of duration C, the same amount of computation is done as during a time αC without checkpointing ($0 \leq \alpha \leq 1$)

Waste in fault-free execution

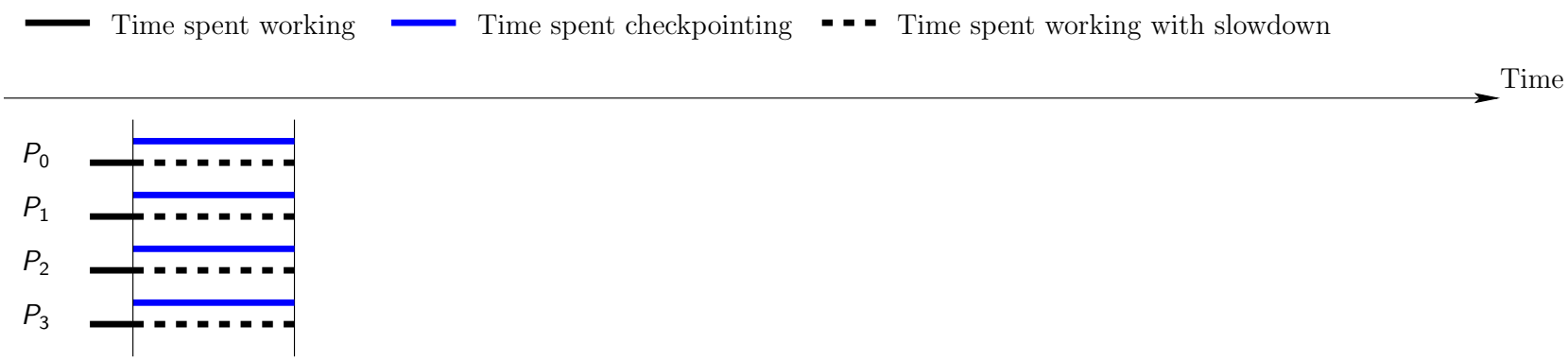


Time elapsed since last checkpoint: T

Amount of computations executed: $\text{WORK} = (T - C) + \alpha C$

$\text{WASTE}[FF] = \frac{T - \text{WORK}}{T}$

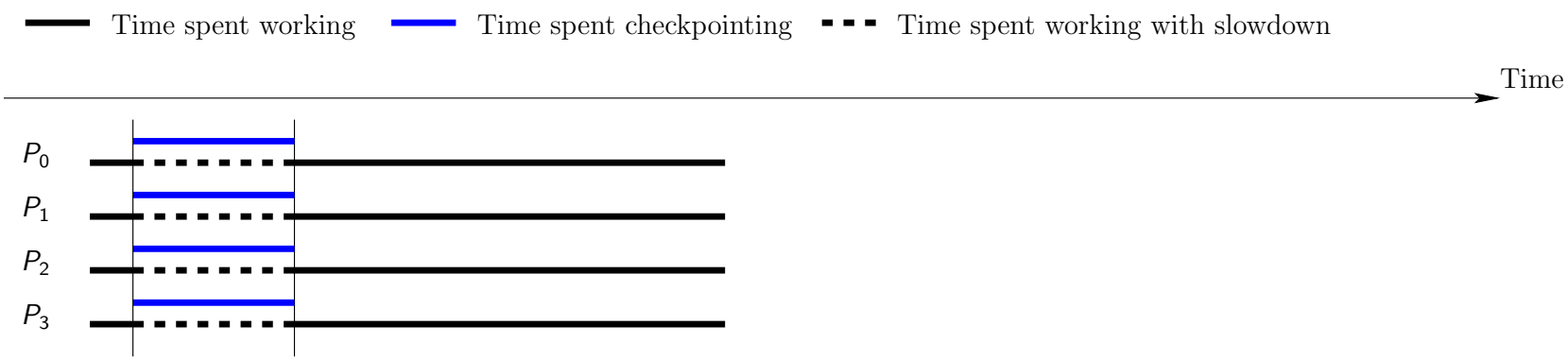
Waste due to failures



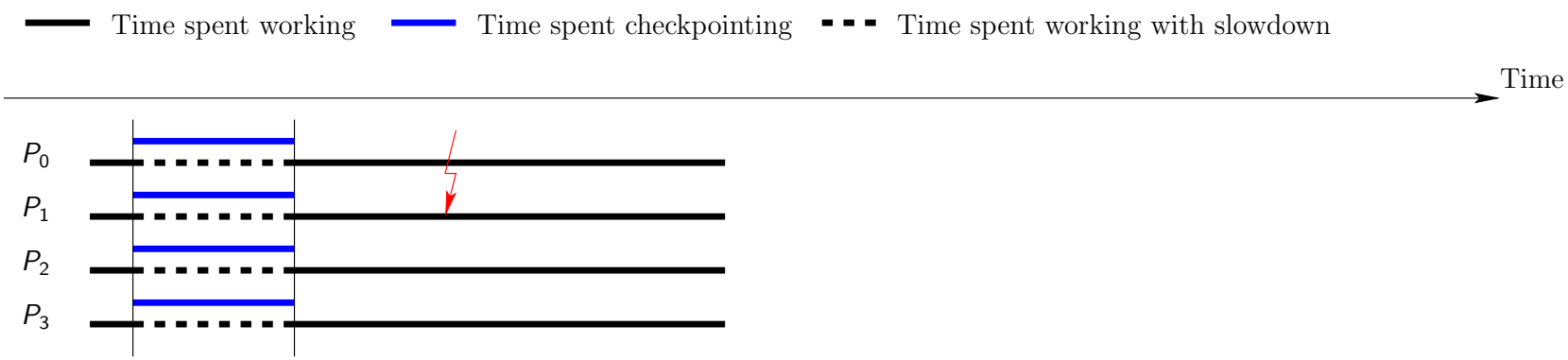
Failure can happen

1. During computation phase
2. During checkpointing phase

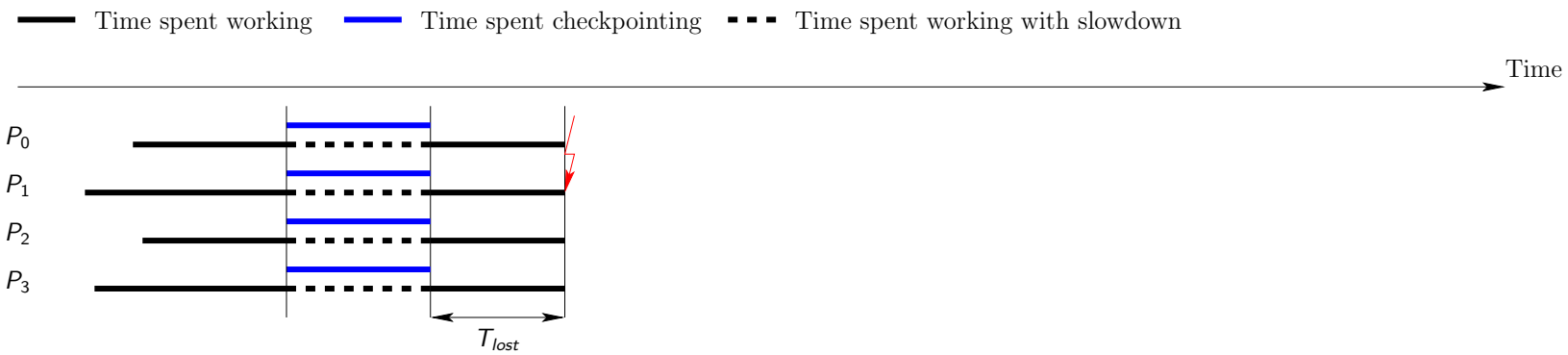
Waste due to failures



Waste due to failures

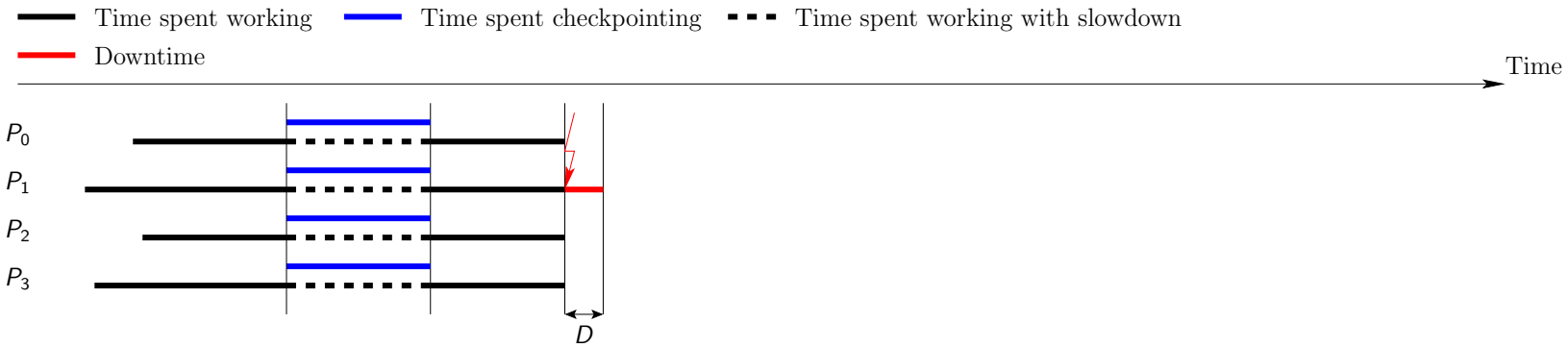


Waste due to failures

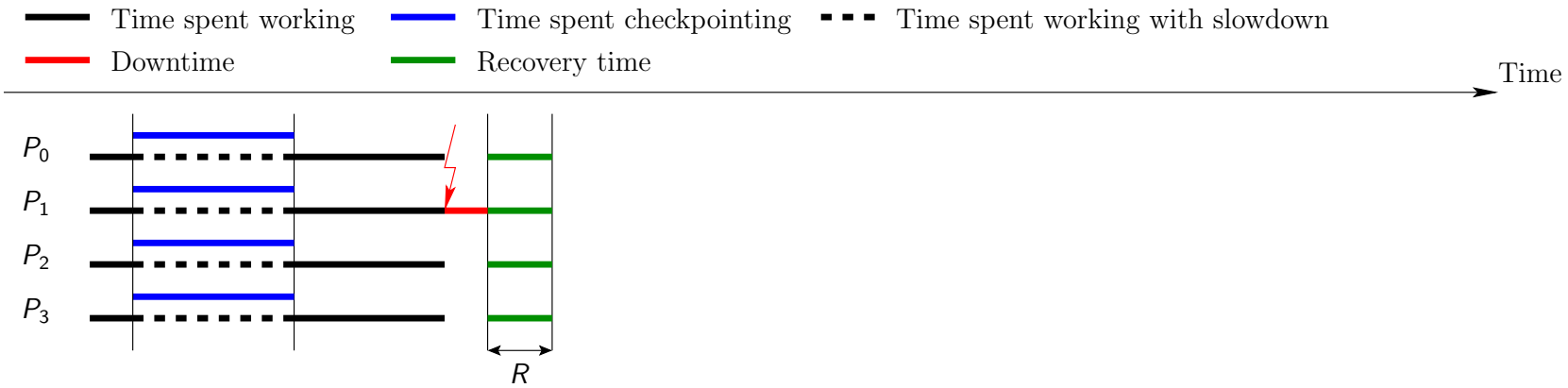


Coordinated checkpointing protocol: when one processor is victim of a failure, all processors lose their work and must roll back to last checkpoint

Waste due to failures in computation phase

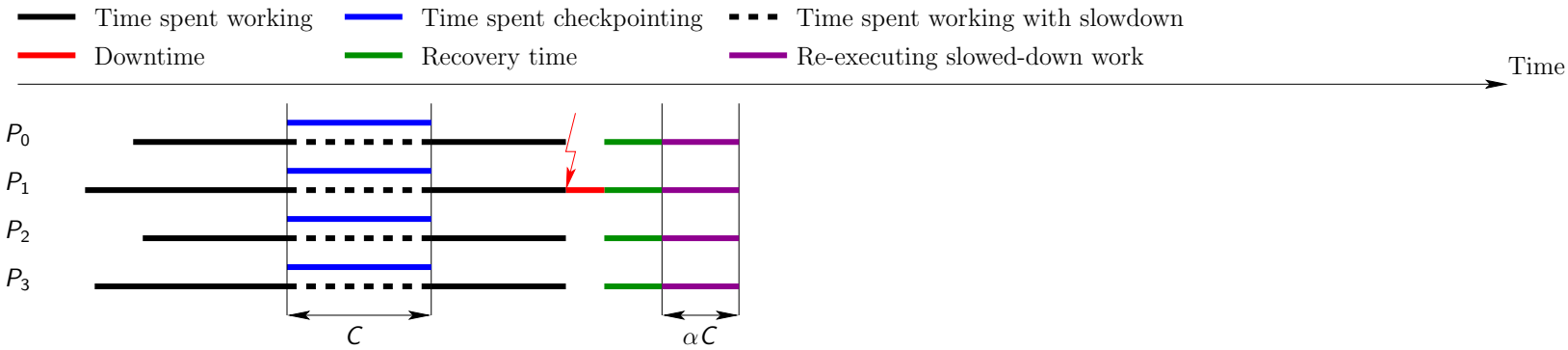


Waste due to failures in computation phase



Coordinated checkpointing protocol: all processors must recover from last checkpoint

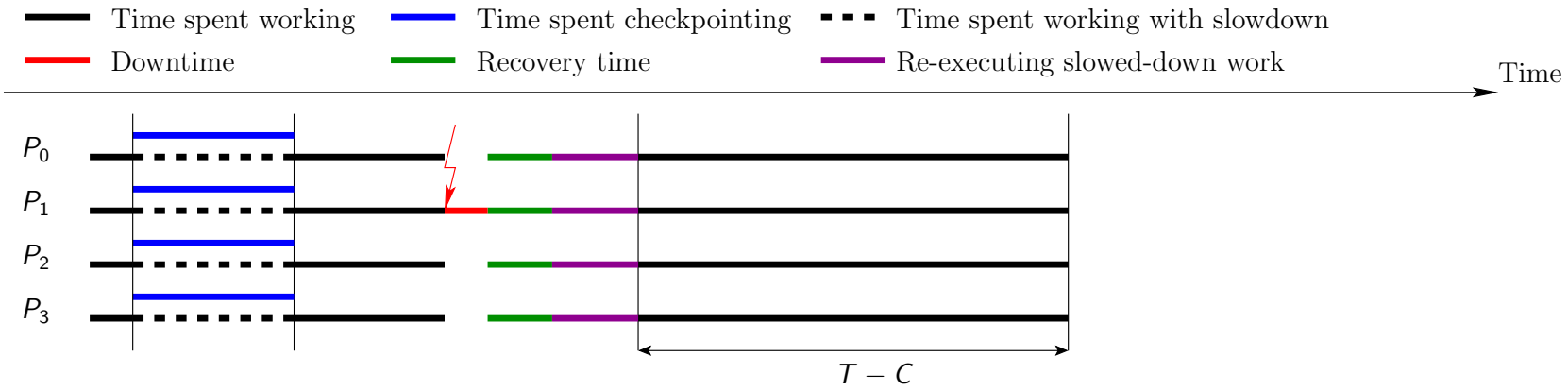
Waste due to failures in computation phase



Redo the work destroyed by the failure, that was done in the checkpointing phase before the computation phase

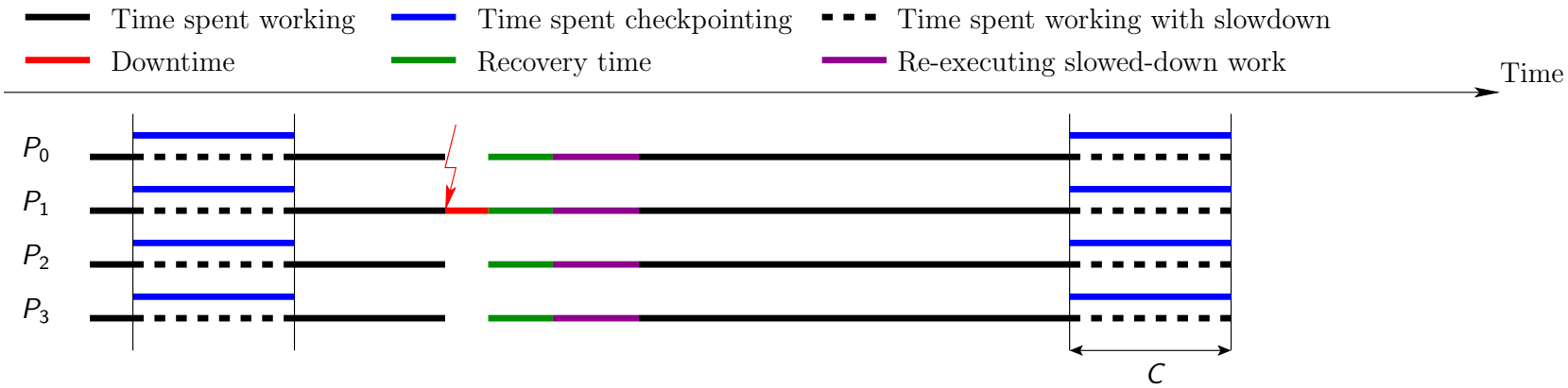
But no checkpoint is taken in parallel, hence this re-execution is faster than the original computation

Waste due to failures in computation phase



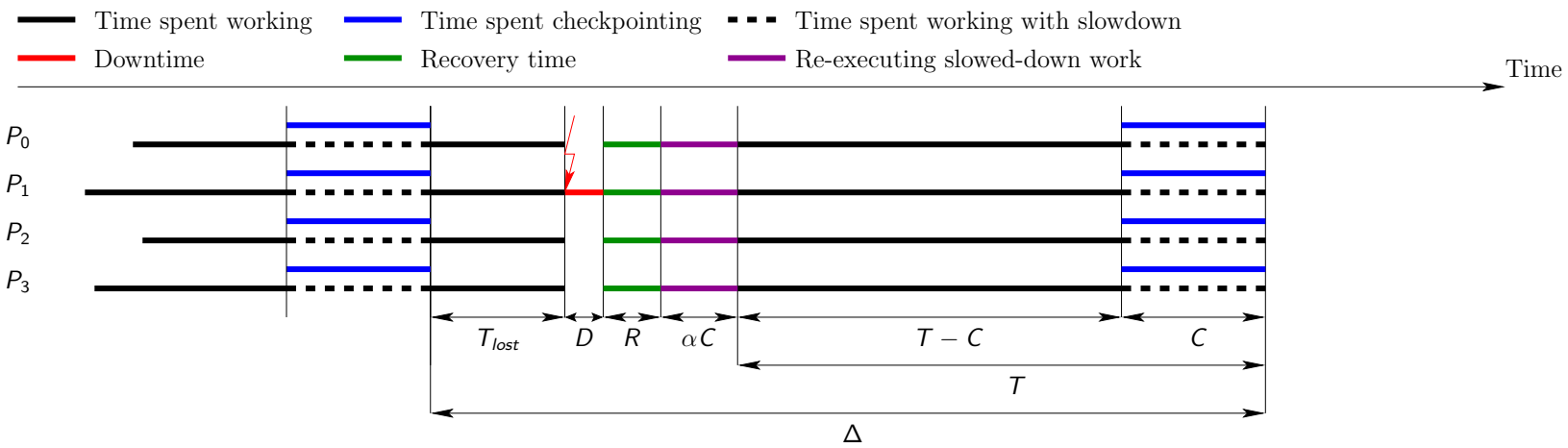
Re-execute the computation phase

Waste due to failures in computation phase



Finally, the checkpointing phase is executed

Total waste

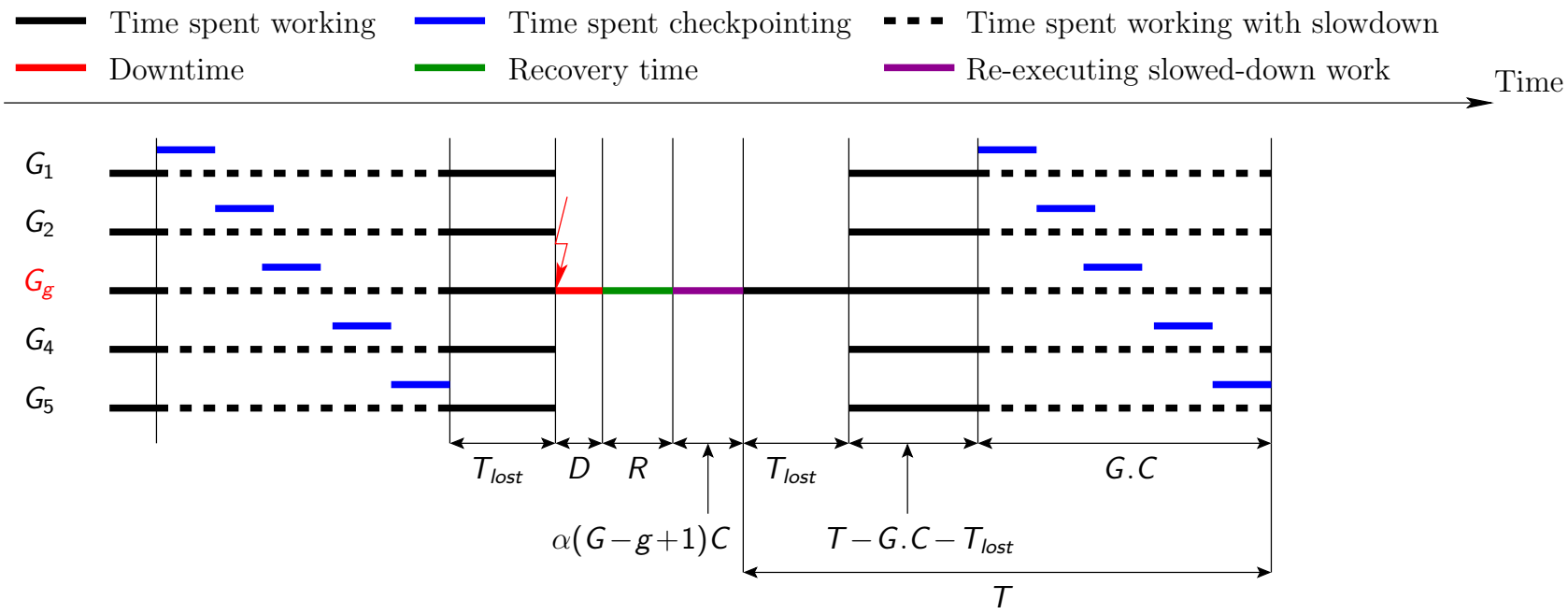


$$\text{WASTE}[\mathit{fail}] = \frac{1}{\mu}\left(D + R + \alpha C + \frac{T}{2}\right)$$

Optimal period $T_{\text{opt}} = \sqrt{2(1-\alpha)(\mu - (D+R))C}$

Outline

Hierarchical checkpointing



- Processors partitioned into G groups
- Each group includes q processors
- Inside each group: coordinated checkpointing in time $C(q)$
- Inter-group messages are logged

Accounting for message logging: Impact on work

- ☹ Logging messages slows down execution:
  $\Rightarrow$ WORK becomes λWORK, where $0 < \lambda < 1$
  Typical value: $\lambda \approx 0.98$
- ☺ Re-execution after a failure is faster:
  $\Rightarrow$ RE-EXEC becomes $\frac{\text{RE-EXEC}}{\rho}$, where $\rho \in [1..2]$
  Typical value: $\rho \approx 1.5$

$$\text{WASTE}[FF] = \frac{T - \lambda \text{WORK}}{T}$$

$$\text{WASTE}[fail] = \frac{1}{\mu}\left(D(q) + R(q) + \frac{\text{RE-EXEC}}{\rho}\right)$$

Accounting for message logging: Impact on checkpoint size

- Inter-groups messages logged continuously
- Checkpoint size increases with amount of work executed before a checkpoint
- $C_0(q)$: Checkpoint size of a group without message logging

$$C(q) = C_0(q)(1 + \beta \text{WORK}) \Leftrightarrow \beta = \frac{C(q) - C_0(q)}{C_0(q)\text{WORK}}$$

$$\text{WORK} = \lambda(T - (1 - \alpha)GC(q))$$

$$C(q) = \frac{C_0(q)(1 + \beta\lambda T)}{1 + GC_0(q)\beta\lambda(1 - \alpha)}$$

Three case studies

Coord-IO

Coordinated approach: $C = C_{\text{Mem}} = \frac{\text{Mem}}{\text{b}_{io}}$
where Mem is the memory footprint of the application

Hierarch-IO

Several (large) groups, *I/O-saturated*
$\Rightarrow$ groups checkpoint sequentially

$$C_0(q) = \frac{C_{\text{Mem}}}{G} = \frac{\text{Mem}}{G\text{b}_{io}}$$

Hierarch-Port

Very large number of smaller groups, *port-saturated*
$\Rightarrow$ some groups checkpoint in parallel
Groups of q_{min} processors, where $\text{q}_{\text{min}}\text{b}_{port} \geq \text{b}_{io}$

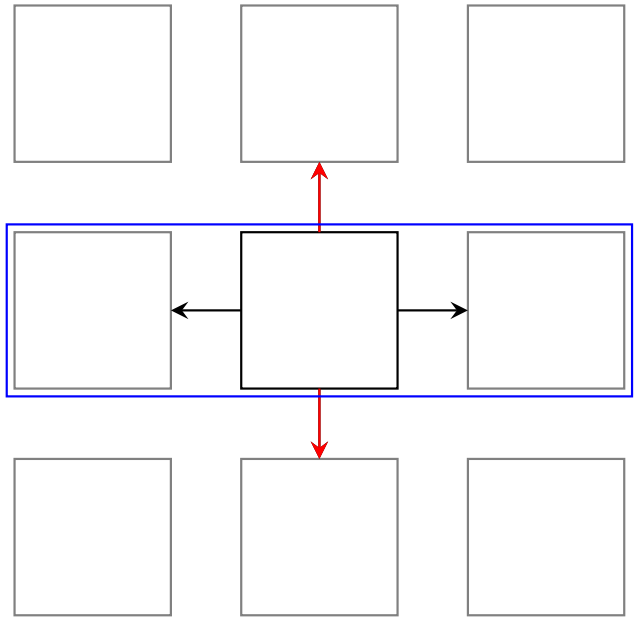
Three applications

1. 2D-stencil
2. Matrix product
3. 3D-Stencil
   - Plane
   - Line

Computing β for 2D-Stencil

$$C(q) = C_0(q) + Logged\_Msg = C_0(q)(1 + \beta \text{WORK})$$

Real $n \times n$ matrix and $p \times p$ grid
$Work = \frac{9b^2}{s_p}$, $b = n/p$
Each process sends a block to its 4 neighbors



HIERARCH-IO:

- 1 group = 1 grid row
- 2 out of the 4 messages are logged
- $\beta = \frac{2s_p}{9b^3}$

HIERARCH-PORT:

- β doubles

Four platforms: basic characteristics

Name	Number of cores	Number of processors p_{total}	Number of cores per processor	Memory per processor	I/O Network Bandwidth (b_{io}) Read	Write	I/O Bandwidth (b_{port}) Read/Write per processor
Titan	299,008	16,688	16	32GB	300GB/s	300GB/s	20GB/s
K-Computer	705,024	88,128	8	16GB	150GB/s	96GB/s	20GB/s
Exascale-Slim	1,000,000,000	1,000,000	1,000	64GB	1TB/s	1TB/s	200GB/s
Exascale-Fat	1,000,000,000	100,000	10,000	640GB	1TB/s	1TB/s	400GB/s

Name	Scenario	G ($C(q)$)	β for 2D-Stencil	β for Matrix-Product
Titan	Coord-IO	1 (2,048s)	/	/
	Hierarch-IO	136 (15s)	0.0001098	0.0004280
	Hierarch-Port	1,246 (1.6s)	0.0002196	0.0008561
K-Computer	Coord-IO	1 (14,688s)	/	/
	Hierarch-IO	296 (50s)	0.0002858	0.001113
	Hierarch-Port	17,626 (0.83s)	0.0005716	0.002227
Exascale-Slim	Coord-IO	1 (64,000s)	/	/
	Hierarch-IO	1,000 (64s)	0.0002599	0.001013
	Hierarch-Port	200,0000 (0.32s)	0.0005199	0.002026
Exascale-Fat	Coord-IO	1 (64,000s)	/	/
	Hierarch-IO	316 (217s)	0.00008220	0.0003203
	Hierarch-Port	33,3333 (1.92s)	0.00016440	0.0006407

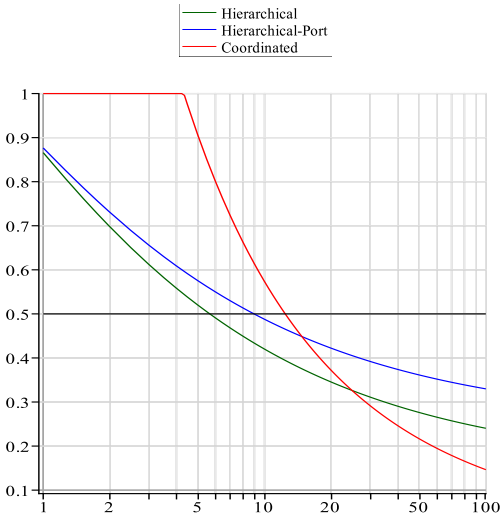
Checkpoint time

Name	C
K-Computer	14,688s
Exascale-Slim	64,000
Exascale-Fat	64,000

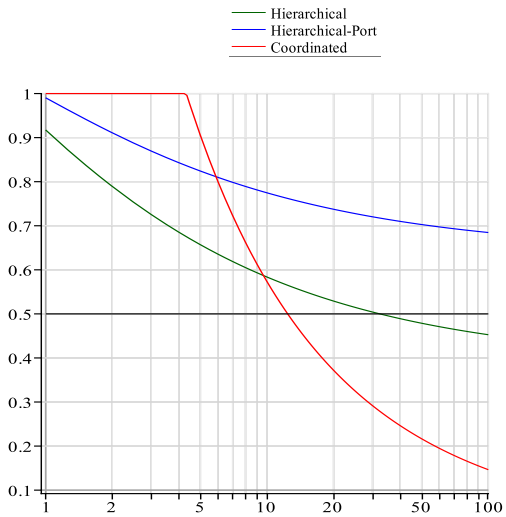
- Large time to dump the memory
- Using $1\%C$
- Comparing with $0.1\%C$ for exascale platforms
- $\alpha = 0.3$, $\lambda = 0.98$ and $\rho = 1.5$

Plotting formulas – Platform: Titan

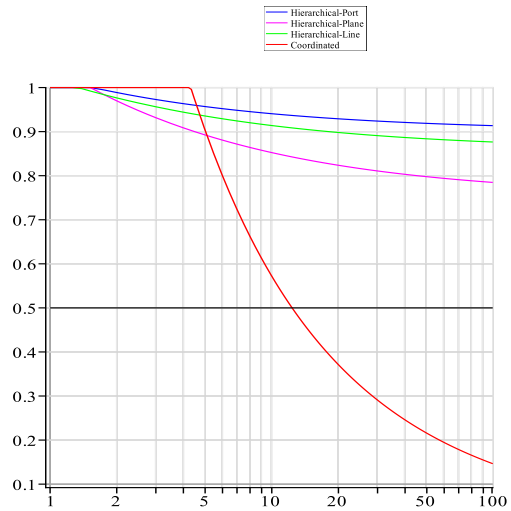
Stencil 2D



Matrix product



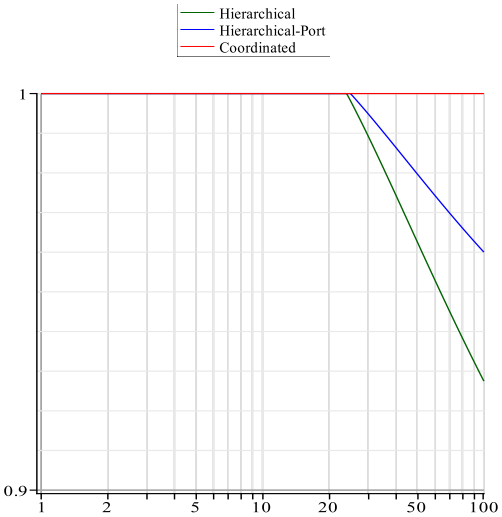
Stencil 3D



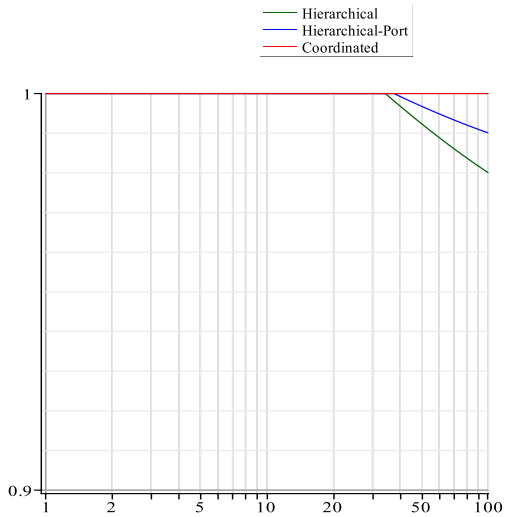
Waste as a function of processor MTBF μ

Platform: K-Computer

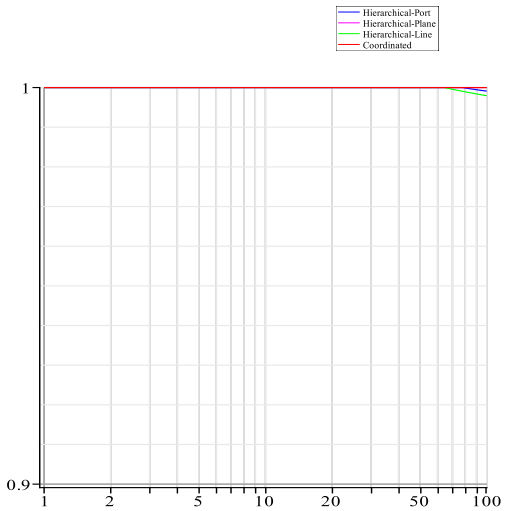
Stencil 2D



Matrix product



Stencil 3D



Waste as a function of processor MTBF μ

Plotting formulas – Platform: Exascale

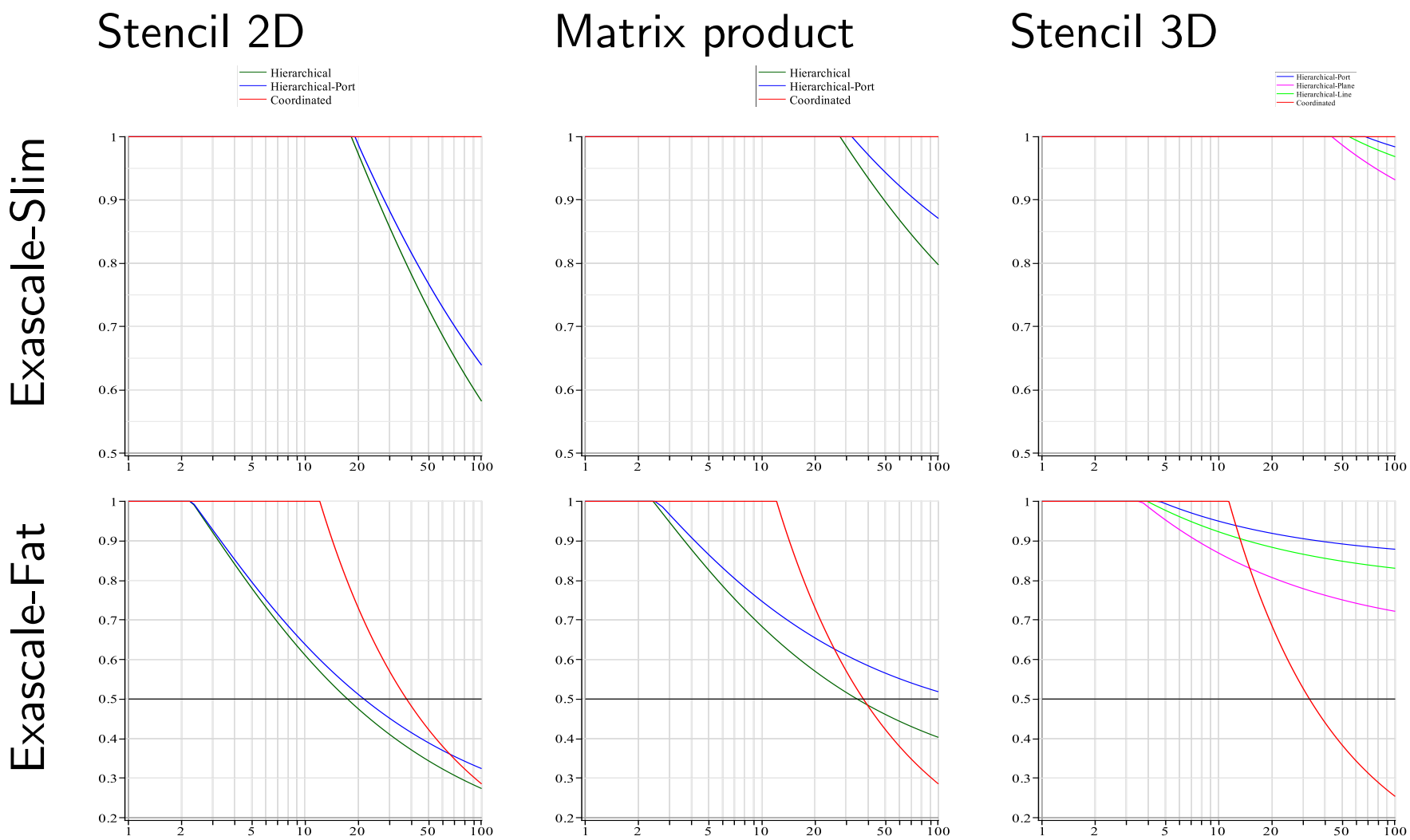
$\textsc{Waste} = 1$ for all scenarios!!!

Plotting formulas – Platform: Exascale

WASTE = ... for all s...narios!!!

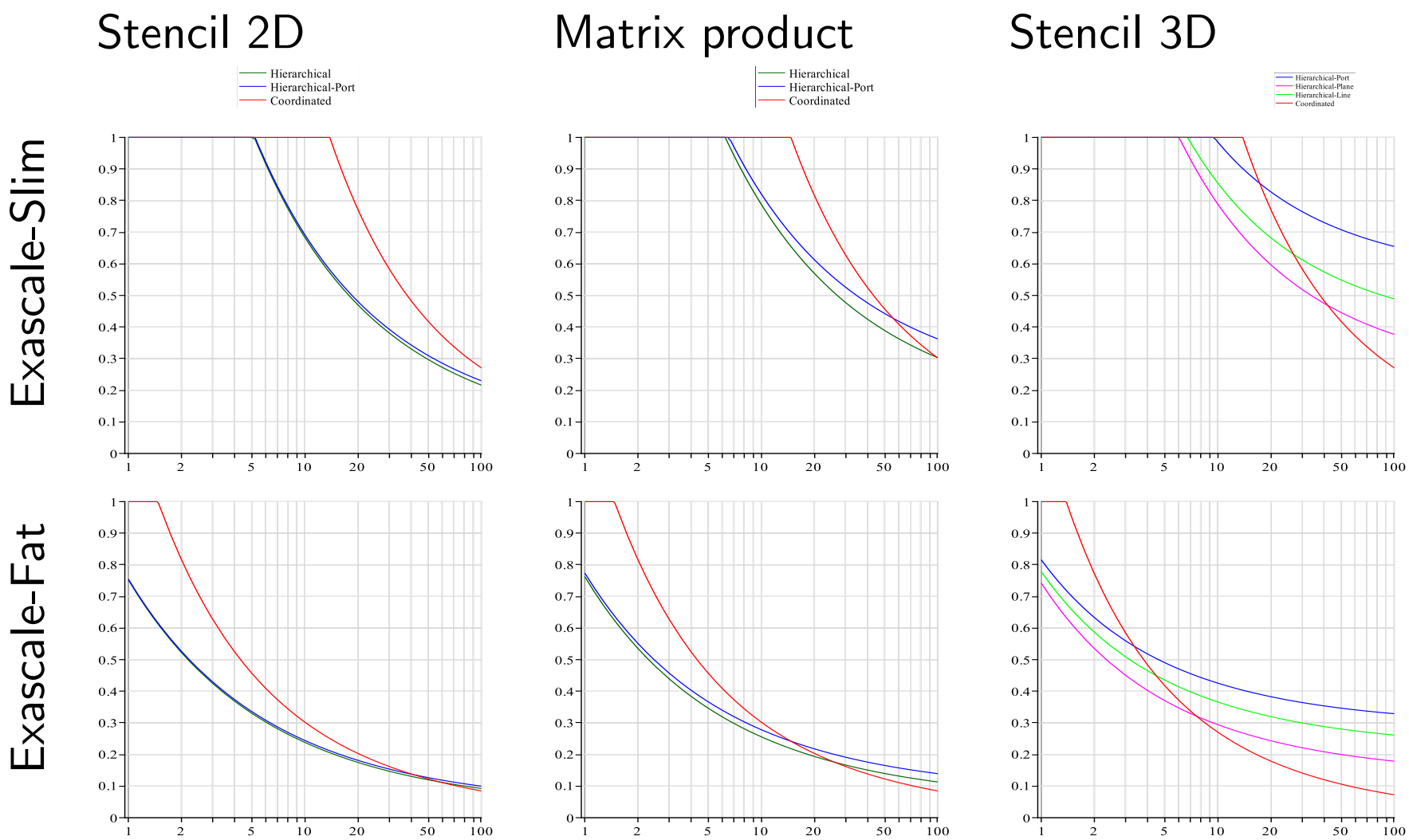
Goodbye Exascale?!

Plotting formulas – Platform: Exascale with $C = 1,000$



Waste as a function of processor MTBF μ, $C = 1,000$

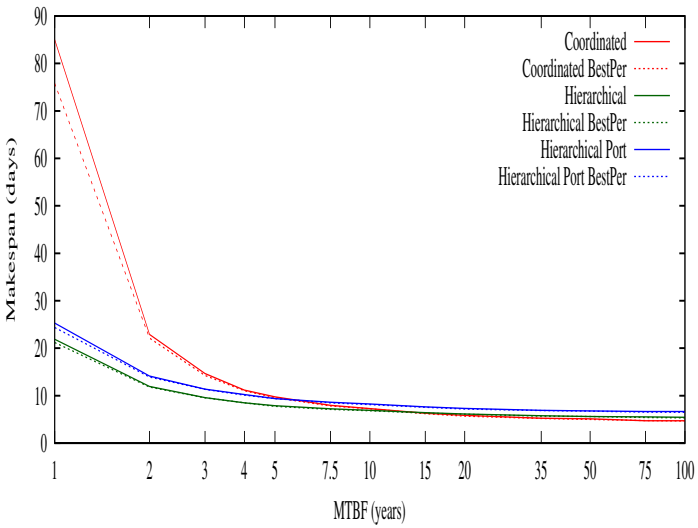
Plotting formulas – Platform: Exascale with $C = 100$



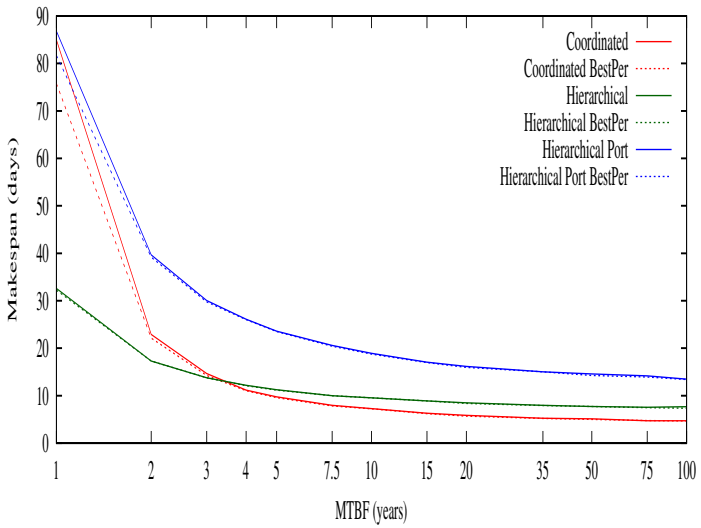
Waste as a function of processor MTBF μ, $C = 100$

Simulations – Platform: Titan

Stencil 2D



Matrix product



Makespan (in days) as a function of processor MTBF μ

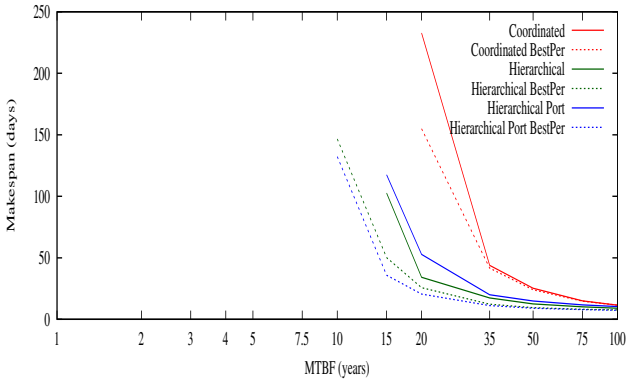
Simulations – Platform: Exascale with $C = 1,000$

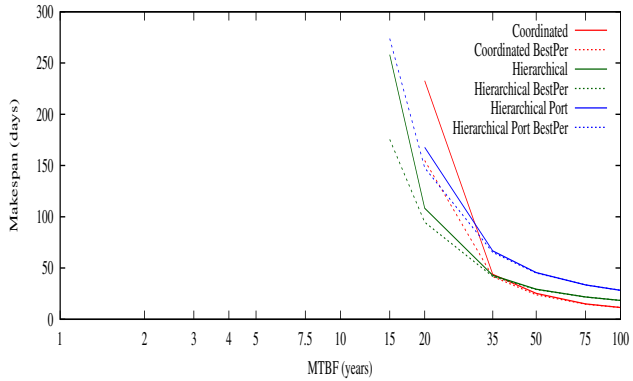
Stencil 2D

Matrix product

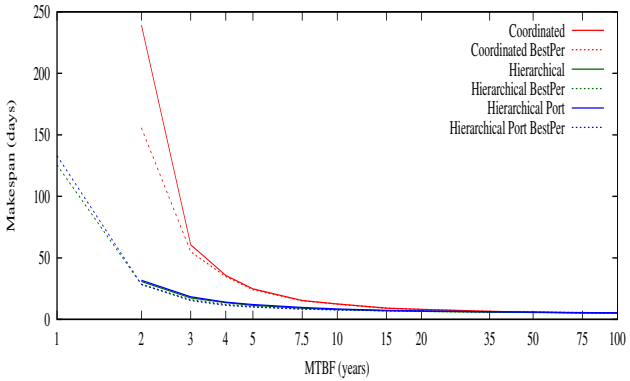
Coordinated
Coordinated BestPer
Hierarchical
Hierarchical BestPer
Hierarchical Port
Hierarchical Port BestPer

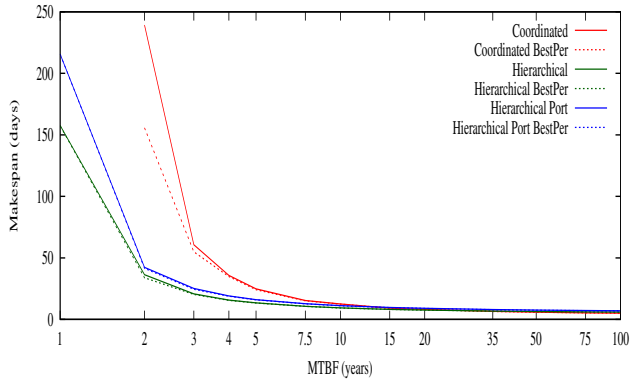
Exascale-Slim





Exascale-Fat





Makespan (in days) as a function of processor MTBF μ, $C = 1,000$

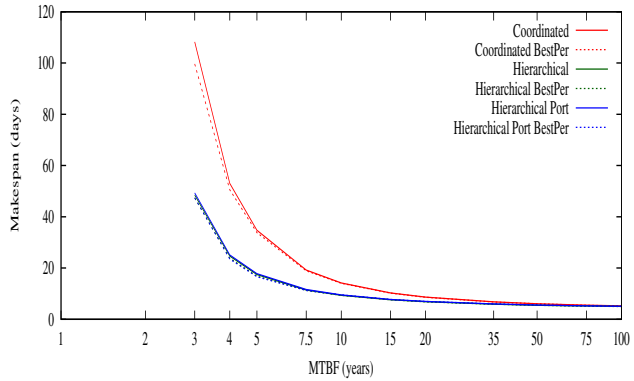
Simulations – Platform: Exascale with $C = 100$

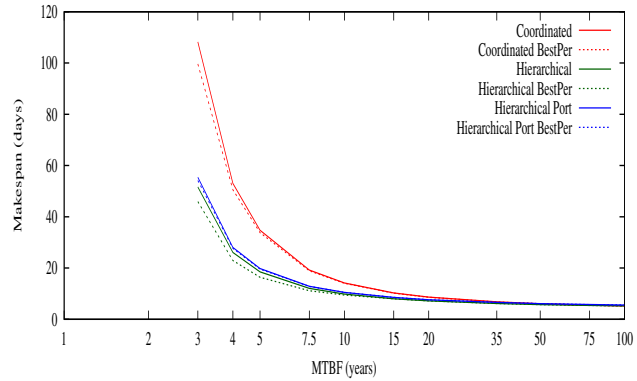
Stencil 2D

Matrix product

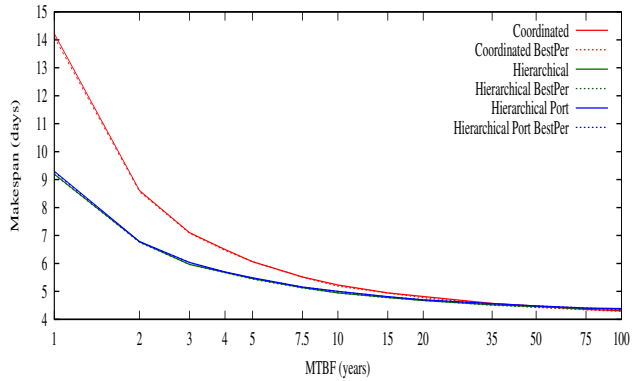
Coordinated
Coordinated BestPer
Hierarchical
Hierarchical BestPer
Hierarchical Port
Hierarchical Port BestPer

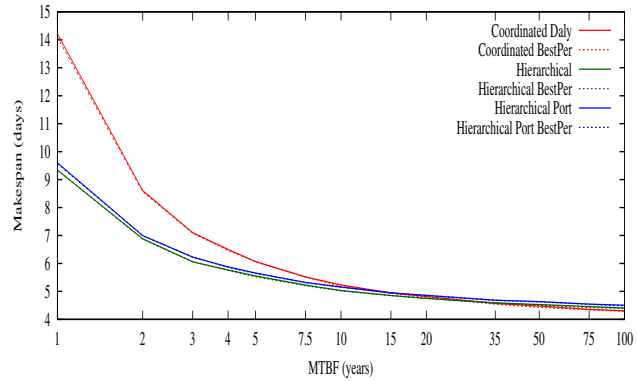
Exascale-Slim





Exascale-Fat

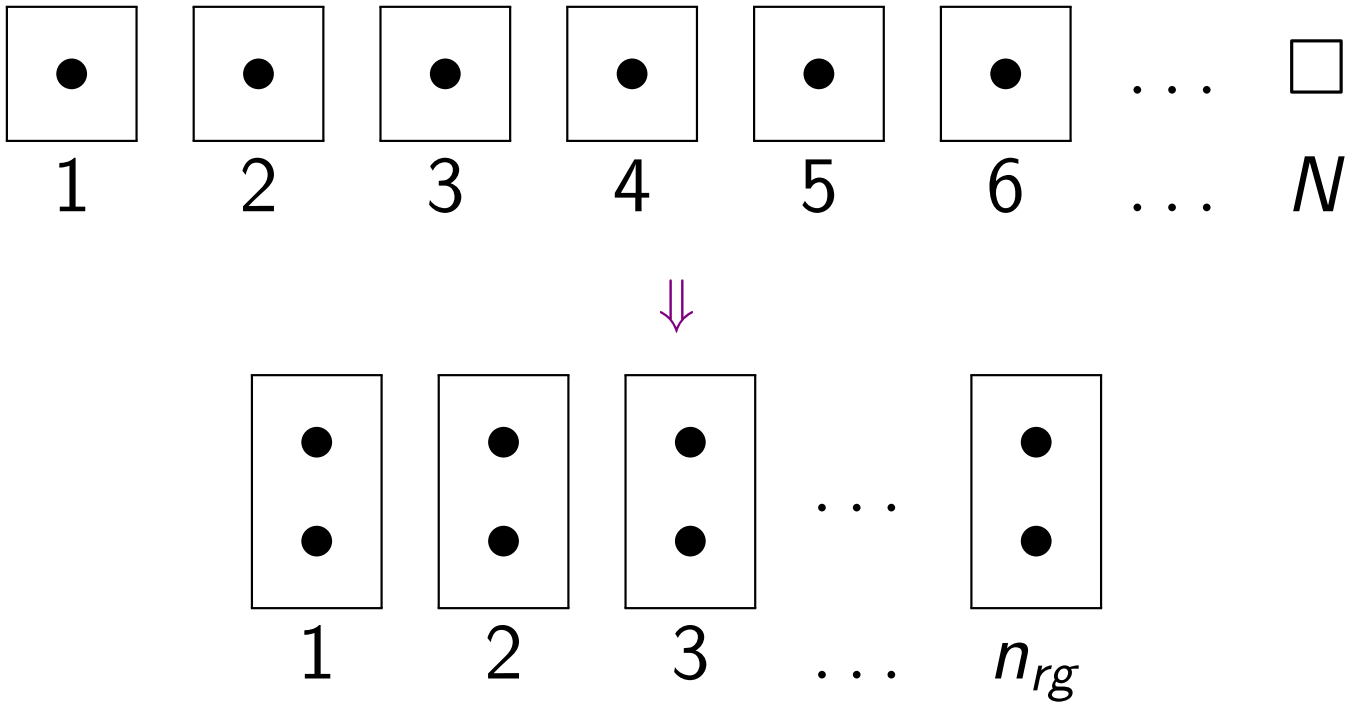




Makespan (in days) as a function of processor MTBF μ, $C = 100$

Outline

Process replication



- Each process replicated $g \geq 2$ times $\rightarrow$ *replica-group*
- n_{rg} = number of replica-groups ($g \times n_{rg} = N$)
- Study for $g = 2$ by Ferreira et al., SC'2011

Number of failures to bring down application

- $MNFTI^{\text{ah}}$ Count each failure hitting any of the $g \cdot n_{rg}$ initial processors, including those *already hit* by a failure
- $MNFTI^{\text{rp}}$ Count failures that hit *running processors*, and thus effectively kill replicas.

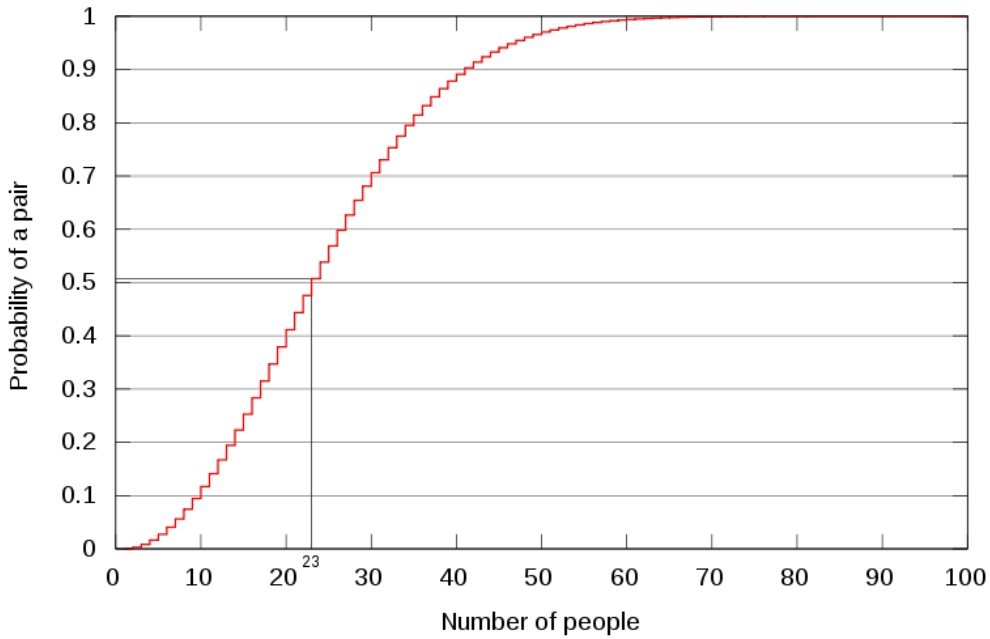
$$MNFTI^{\text{ah}} = 1 + MNFTI^{\text{rp}}$$

Number of failures to bring down application

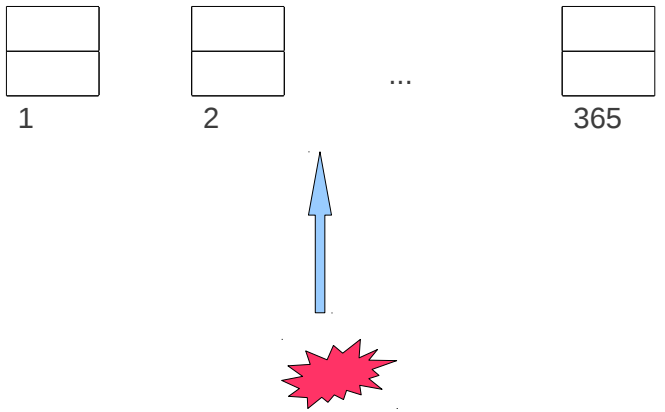
- $MNFTI^{\text{ah}}$ Count each failure hitting any of the $g \cdot n_{rg}$ initial processors, including those *already hit* by a failure
- $MNFTI^{\text{rp}}$ Count failures that hit *running processors*, and thus effectively kill replicas.

$$MNFTI^{\text{ah}} = 1 + MNFTI^{\text{rp}}$$

Analogy with birthday problem

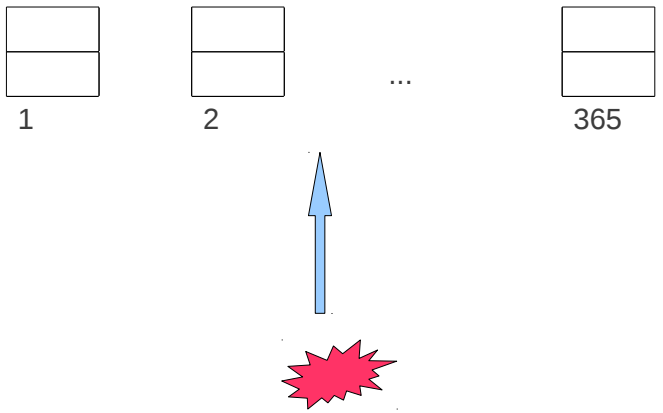


Analogy with birthday problem



$n = n_{rg}$ bins, throw balls until one bin gets two balls

Analogy with birthday problem

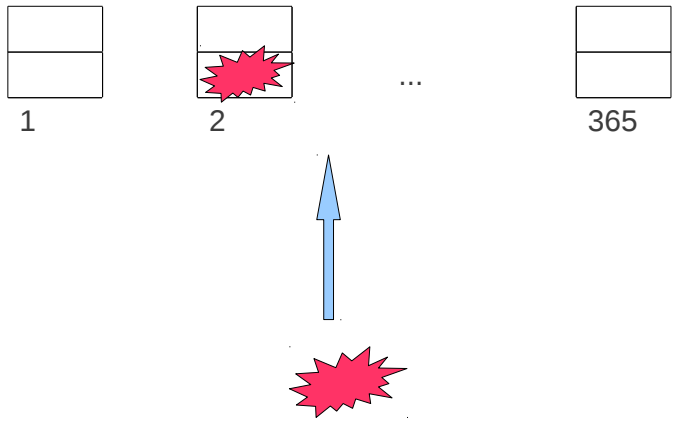


$n = n_{rg}$ bins, throw balls until one bin gets two balls

Expected number of balls to throw:

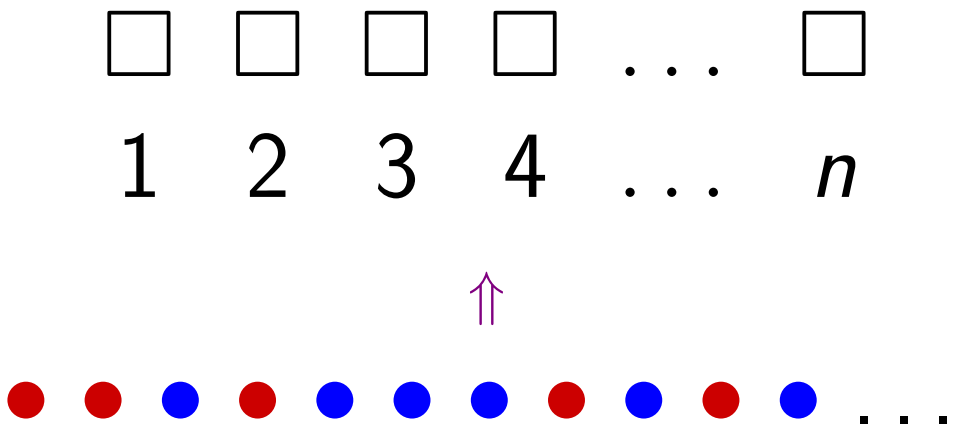
$$Birthday(n) = 1 + \int_0^{+\infty} e^{-x}(1 + x/n)^{n-1} dx$$

Analogy with birthday problem



But second failure may hit already struck replica ☹

Analogy with birthday problem



$n = n_{rg}$ bins, red and blue balls

$MNFTI^{\text{ah}}$ = expected number of balls to throw until one bin gets one ball of each color

Exponential failures, $g = 2$

Theorem $MNFTI^{\text{ah}} = \mathbb{E}(NFTI^{\text{ah}}|0)$ where

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

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

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

Exponential failures, $g = 2$ (cont'd)

Proof

$$\mathbb{E}\left(NFTI^{\mathrm{ah}}\,|n_{rg}\right) = \frac{1}{2}\times 1 + \frac{1}{2}\times\left(1+\mathbb{E}\left(NFTI^{\mathrm{ah}}\,|n_{rg}\right)\right).$$

$$\mathbb{E}\left(NFTI^{\mathrm{ah}}|n_f\right) = \frac{2n_{rg}-2n_f}{2n_{rg}}\times\left(1+\mathbb{E}\left(NFTI^{\mathrm{ah}}|n_f+1\right)\right) + \frac{2n_f}{2n_{rg}}\times\left(\frac{1}{2}\times 1+\frac{1}{2}\left(1+\mathbb{E}\left(NFTI^{\mathrm{ah}}|n_f\right)\right)\right).$$

$$MTTI = systemMTBF(2n_{rg})\times MNFTI^{\mathrm{ah}}$$

Exponential failures, $g = 2$ (cont'd)

Proof

$$\mathbb{E}\left(NFTI^{\text{ah}}\,|n_{rg}\right) = \frac{1}{2} \times 1 + \frac{1}{2} \times \left(1 + \mathbb{E}\left(NFTI^{\text{ah}}\,|n_{rg}\right)\right).$$

$$\mathbb{E}\left(NFTI^{\text{ah}}|n_f\right) = \frac{2n_{rg} - 2n_f}{2n_{rg}} \times \left(1 + \mathbb{E}\left(NFTI^{\text{ah}}|n_f + 1\right)\right) + \frac{2n_f}{2n_{rg}} \times \left(\frac{1}{2} \times 1 + \frac{1}{2}\left(1 + \mathbb{E}\left(NFTI^{\text{ah}}|n_f\right)\right)\right).$$

$$MTTI = systemMTBF(2n_{rg}) \times MNFTI^{\text{ah}}$$

Exponential failures, $g = 2$ (cont'd)

Proof

$$\mathbb{E}\left(NFTI^{\text{ah}}\,|n_{rg}\right) = \frac{1}{2} \times 1 + \frac{1}{2} \times \left(1 + \mathbb{E}\left(NFTI^{\text{ah}}\,|n_{rg}\right)\right).$$

$$\mathbb{E}\left(NFTI^{\text{ah}}|n_f\right) = \frac{2n_{rg} - 2n_f}{2n_{rg}} \times \left(1 + \mathbb{E}\left(NFTI^{\text{ah}}|n_f + 1\right)\right) + \frac{2n_f}{2n_{rg}} \times \left(\frac{1}{2} \times 1 + \frac{1}{2}\left(1 + \mathbb{E}\left(NFTI^{\text{ah}}|n_f\right)\right)\right).$$

$$MTTI = systemMTBF(2n_{rg}) \times MNFTI^{\text{ah}}$$

Failure distribution

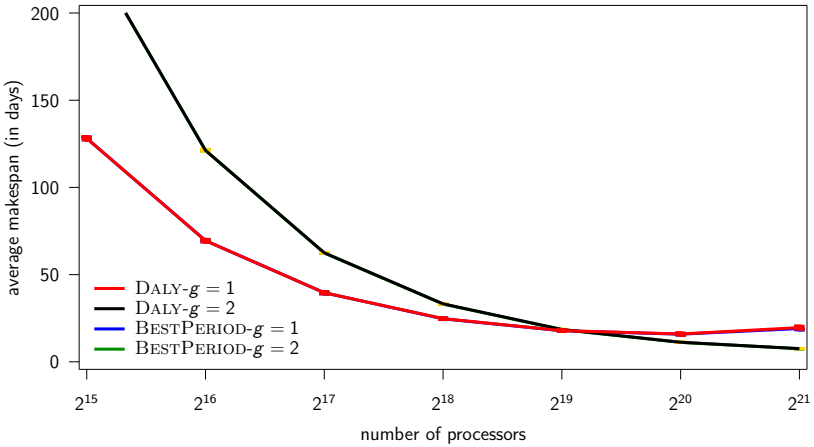
$R(t)$ probability that application still running at time t

- All replica-groups have at least one replica running
- Exponential: $R(t) = \left(1 - \left(1 - e^{-\lambda t}\right)^{g}\right)^{n_{rg}}$
- Weibull: $R(t) = \left(1 - \left(1 - e^{-\left(\frac{t}{\lambda}\right)^{k}}\right)^{g}\right)^{n_{rg}}$
- Can use dynamic programming algorithms for sequential/parallel jobs

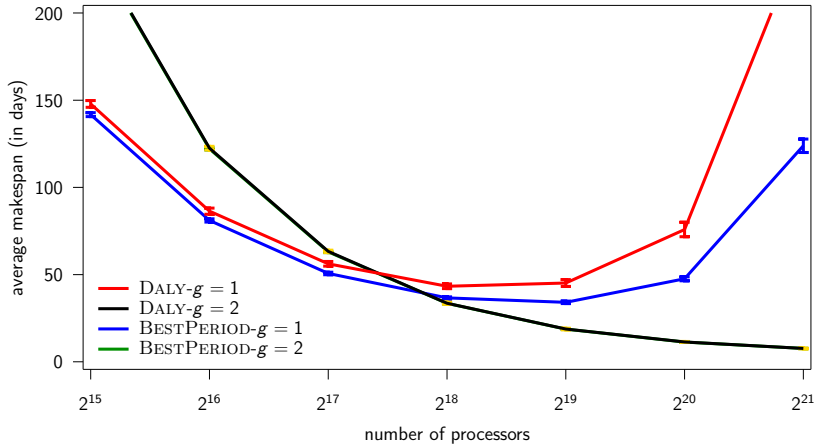
MTTI

- $MTTI = \int_0^{+\infty} R(t)\mathrm{d}t \rightarrow$ closed-form formulas
- Assess the impact of replication for various scenarios ☺

Failure distribution



(a) Exponential



(b) Weibull, $k = 0.7$

Crossover point for replication when $\mu = 125$ years

Outline

Conclusion

- Multiple approaches to Fault Tolerance
- Application-specific FT will always provide more benefits
- General-purpose FT will always be needed
  - Not every computer scientist needs to learn how to write fault-tolerant applications
  - Not all parallel applications can be ported to a fault-tolerant version
- Faults are a feature of the platform. Why should it be the role of the programmers to handle them?

Conclusion

- Software/hardware techniques to reduce checkpoint, recovery, migration times and to improve failure prediction
- Multi-criteria scheduling problem
  execution time/energy/reliability
  add replication
  best resource usage (performance trade-offs)
- Need combine all these approaches!

Several challenging algorithmic/scheduling problems ☺

Extended version of this talk: see SC'12 tutorial