UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Mitigation of Failures in High Performance Computing via Runtime Techniques

Xiang Ni
Advisor: Laxmikant Kale

illinois.edu

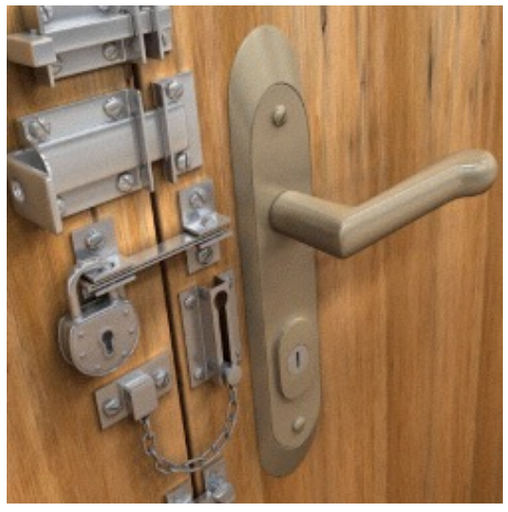
Table of Contents

Fault Tolerance is Everywhere

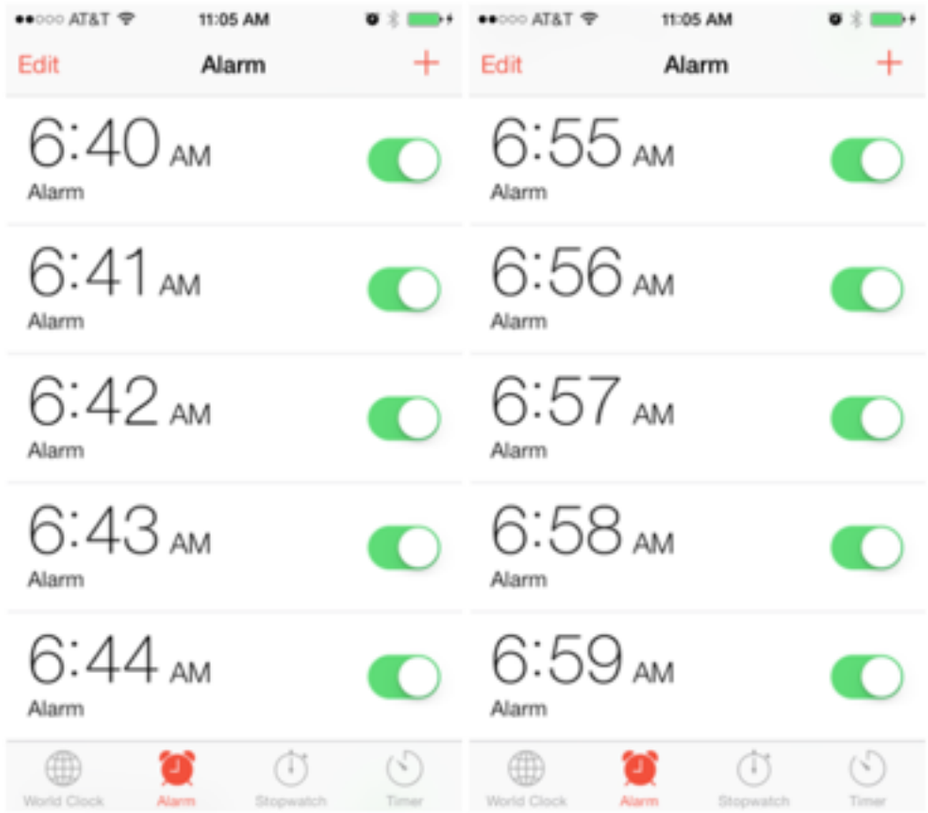
Fault Tolerance is Everywhere

We use **multiple** locks to **protect** our house

Fault Tolerance is Everywhere

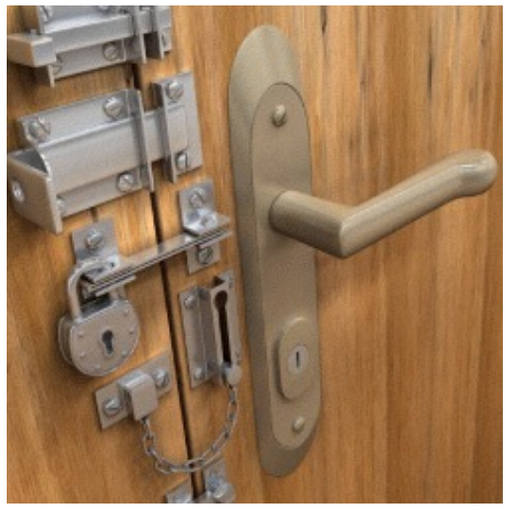


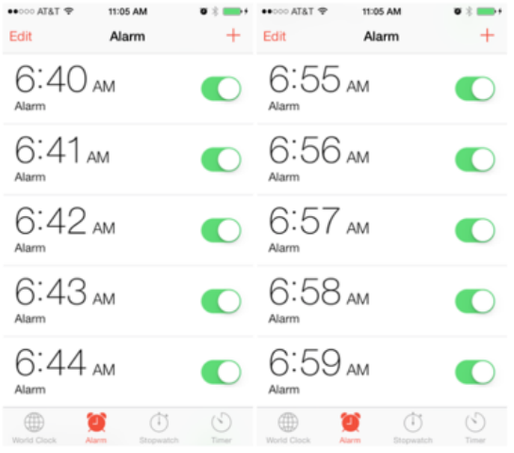
Fault Tolerance is Everywhere



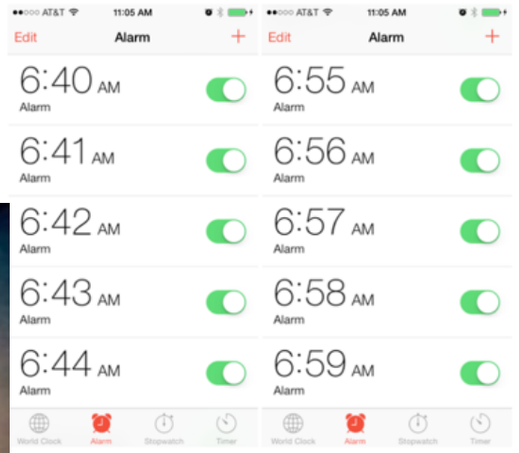
We set as many alarms as possible

Fault Tolerance is Everywhere



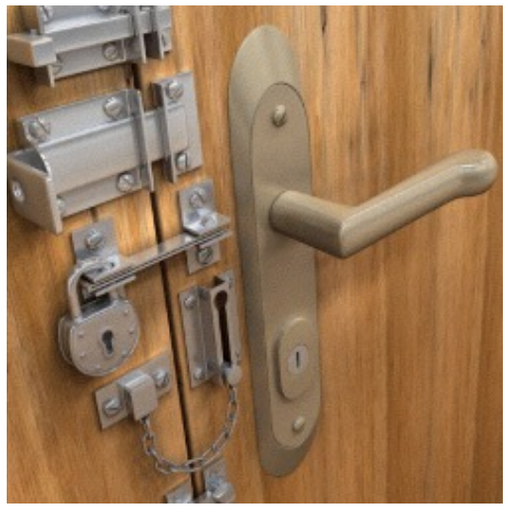


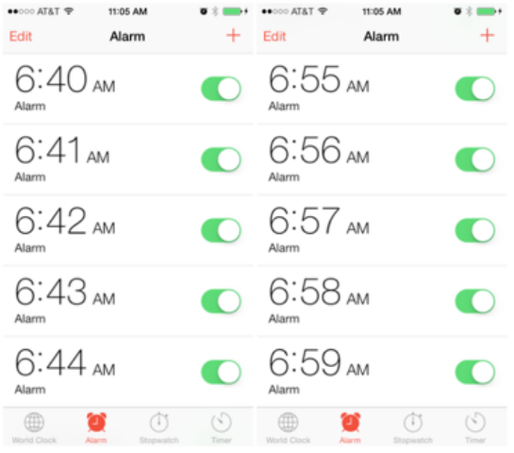
Fault Tolerance is Everywhere



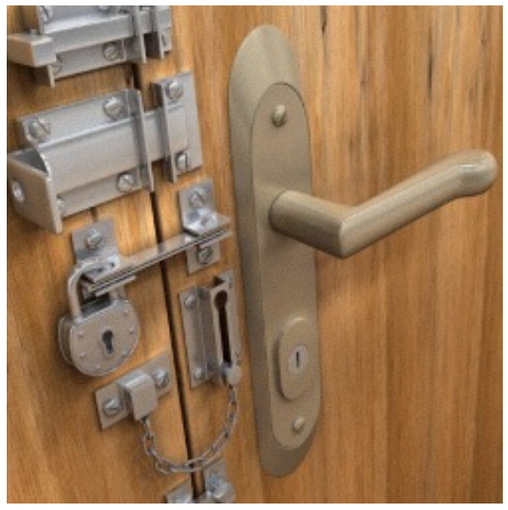
In games or movies, we can **restart** to make progress

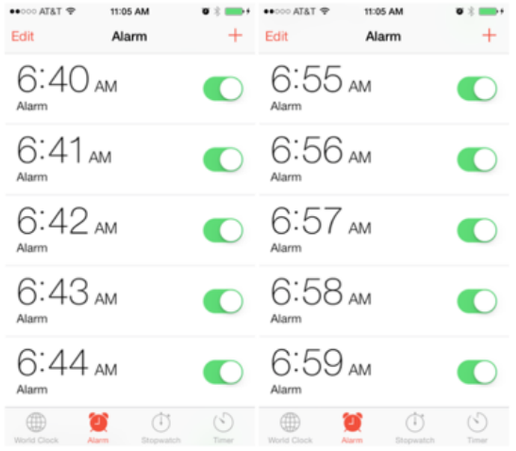
Fault Tolerance is Everywhere





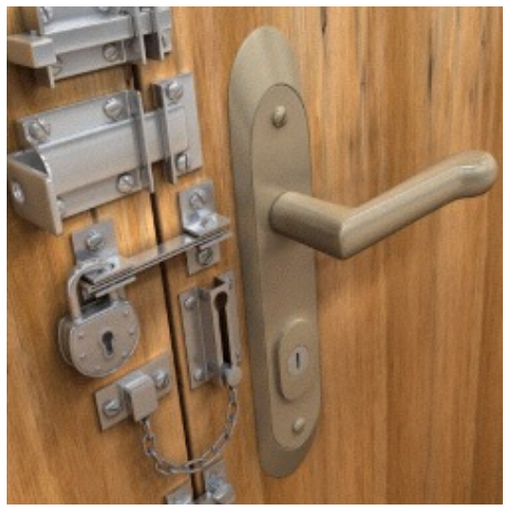
Fault Tolerance is Everywhere

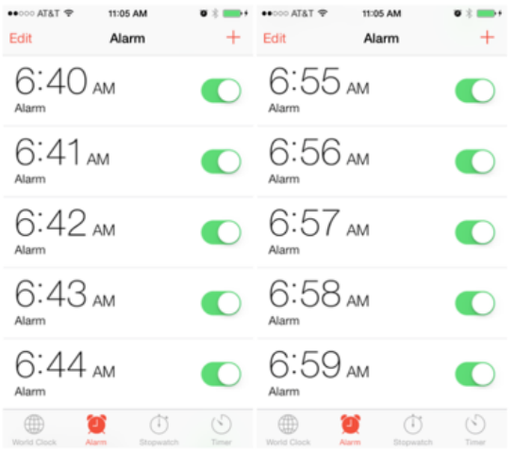




But not with your traditional HPC applications

Fault Tolerance is Everywhere





But not with your traditional HPC applications

Fault Tolerance

Failures are Rising in HPC

1.61 Failures/day
(2014)

Memory Errors
Machine Check Exception
Voltage Fault

- Titan: 2nd largest in the world at Oak Ridge National Lab
- More than 17 PFlops with 560640 cores

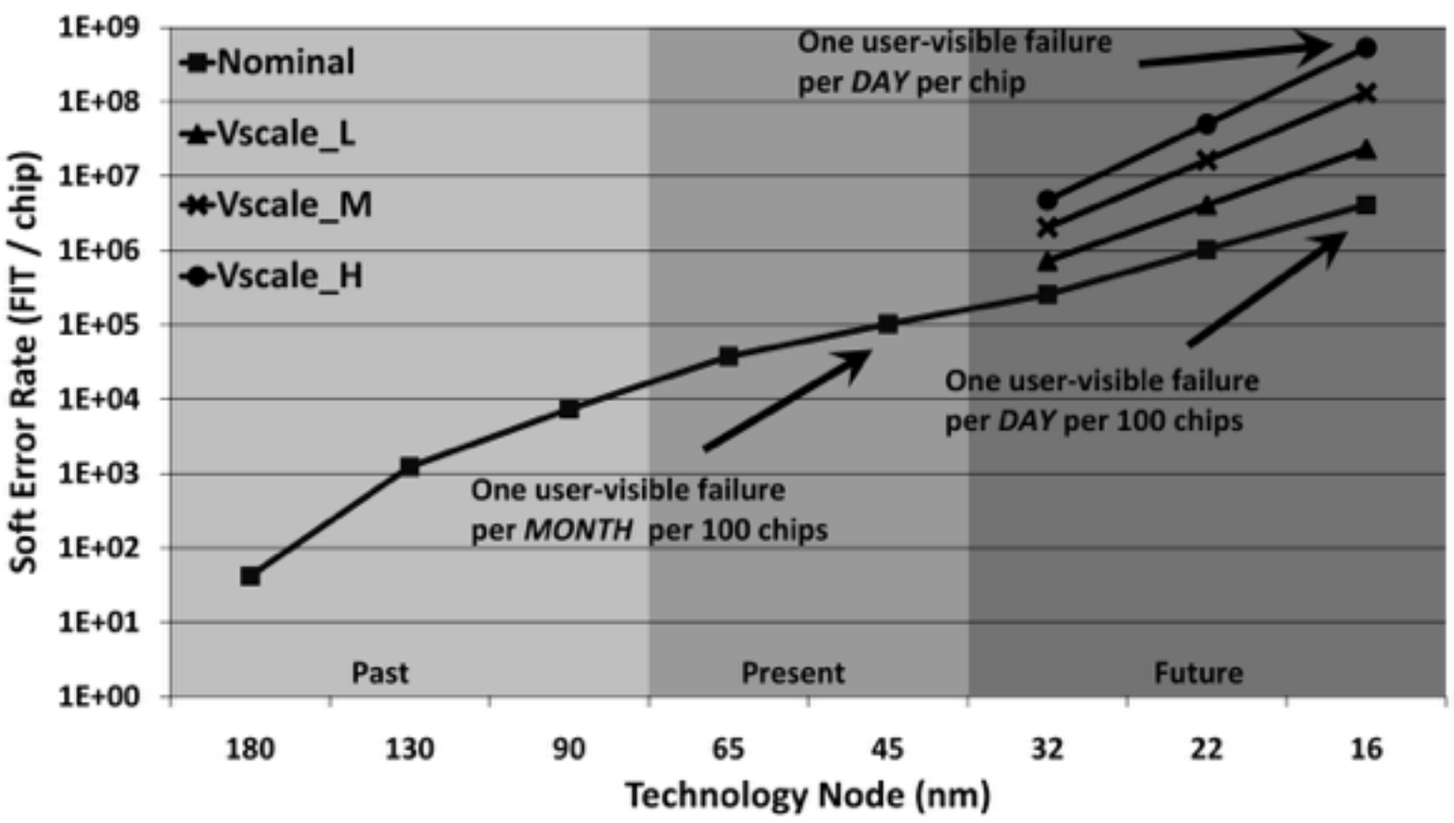
Exascale machines that are **100** times powerful will have more obstacles.

Commercial vendors unlikely to address resilience issues for HPC market.

CUG15: Analyzing the Interplay of Failures and Workload on Titan Supercomputer

Failures maybe Silent

- Common source of soft errors
  - Particle induced single event upset
  - Manufacturing fault
- Data corruption: you may or may not know



Shrinking chip size

- More energy efficient
- Higher soft error rate

Less energy required to corrupt data in memory

Failures maybe Silent

Platform Testing at LANL

Platform 1: 70 incorrect Linpack calculations;
all involve 1 node.

Platform 2: 2 incorrect Linpack calculations

Platform 3: 2 incorrect Linpack calculations

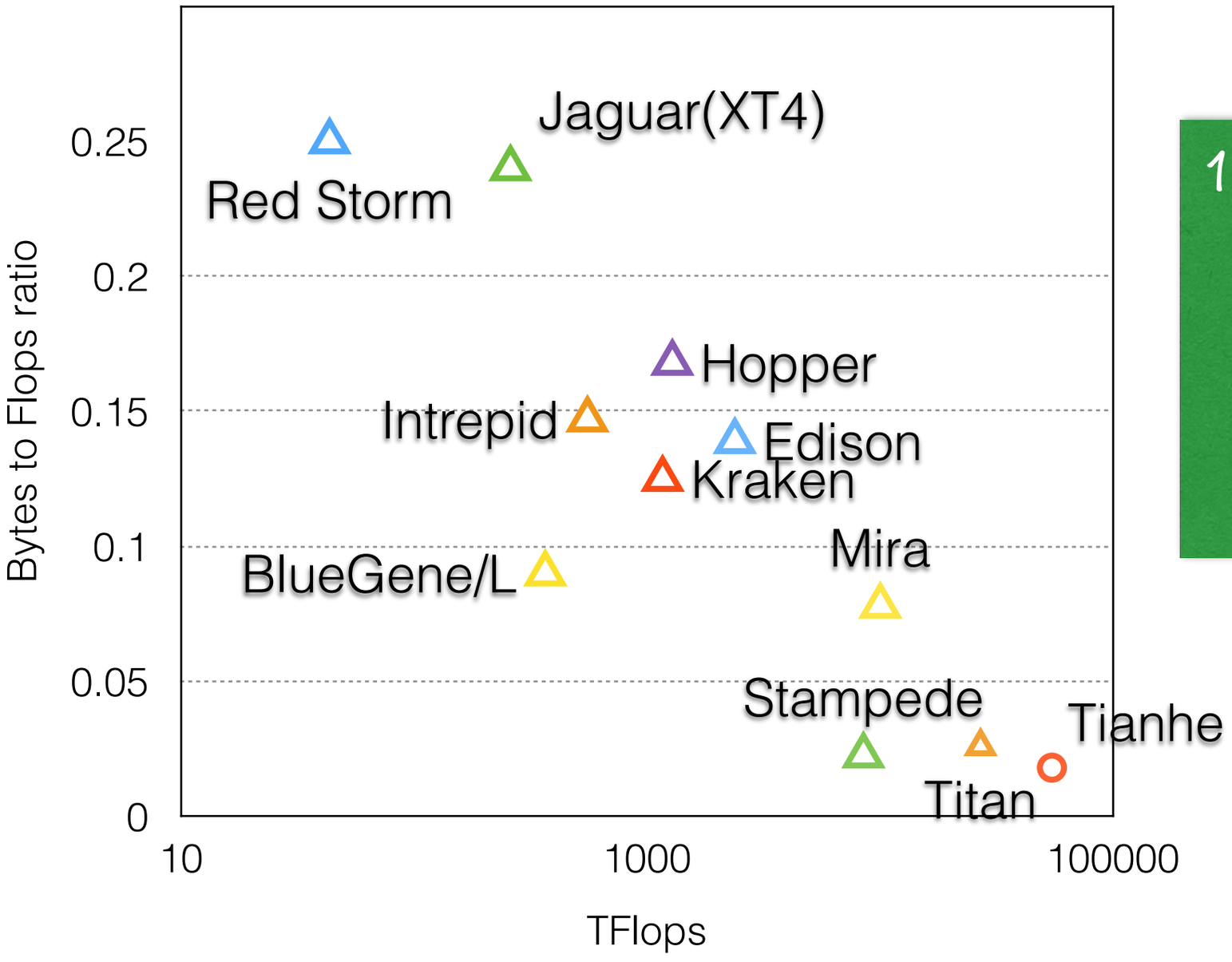
Platform 4&5: no SDC

Neutron Beam Testing at LANL

Crashes and SDCs observed.

Memory is Limited

Decreasing memory-to-flops ratio



1 node of Summit (ORNL)

Memory: 0.5 TB

Flops: 44 TFlops

Bytes/Flops: 0.011

Memory is Limited

	Read Latency	Write Latency
DDR3	.01μs	.01μs
PCM	.05μs	1μs
NAND	10μs	100μs
DISK	1000us	1000us

It is promising to use new types of memory in HPC.

Thesis Focus

- Develop runtime techniques to protect HPC applications from failures

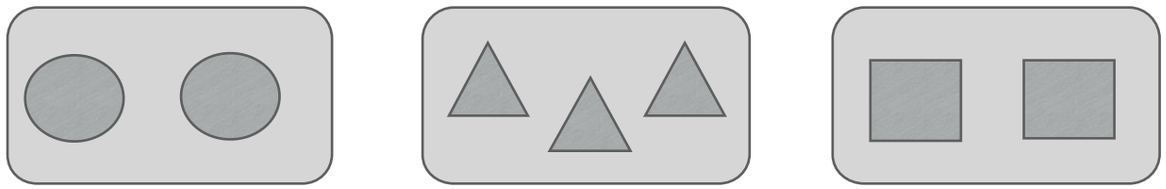
  - **Hard errors**: reducing checkpoint and restart overhead
  - **Soft errors**: detecting and correcting silent data corruptions
  - **Lack of memory**: how to utilize NVRAM for checkpointing and application execution?

Part 1

Protection for Hard Errors

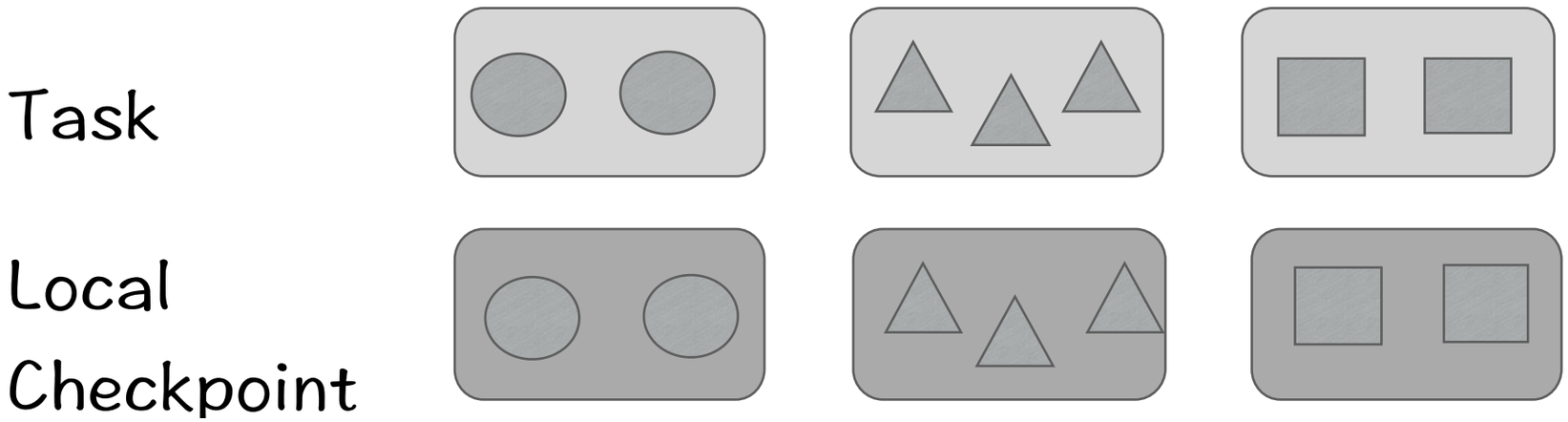
Double In-memory Checkpointing

Task

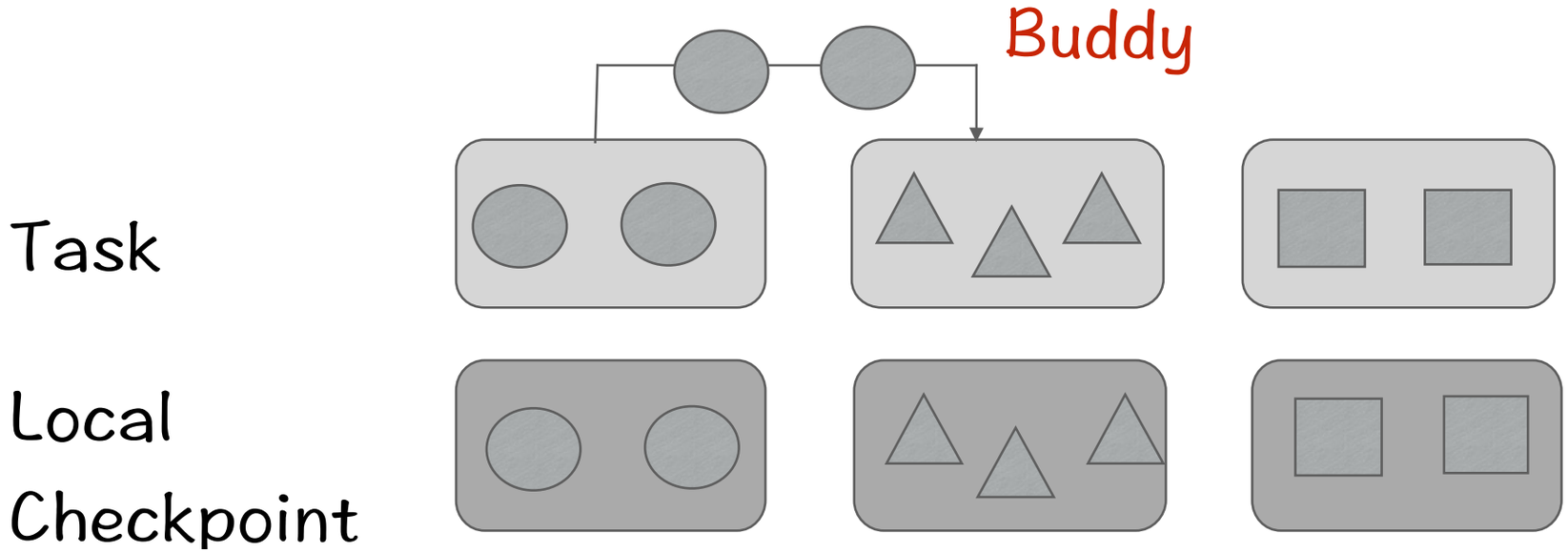


Node A Node B Node C

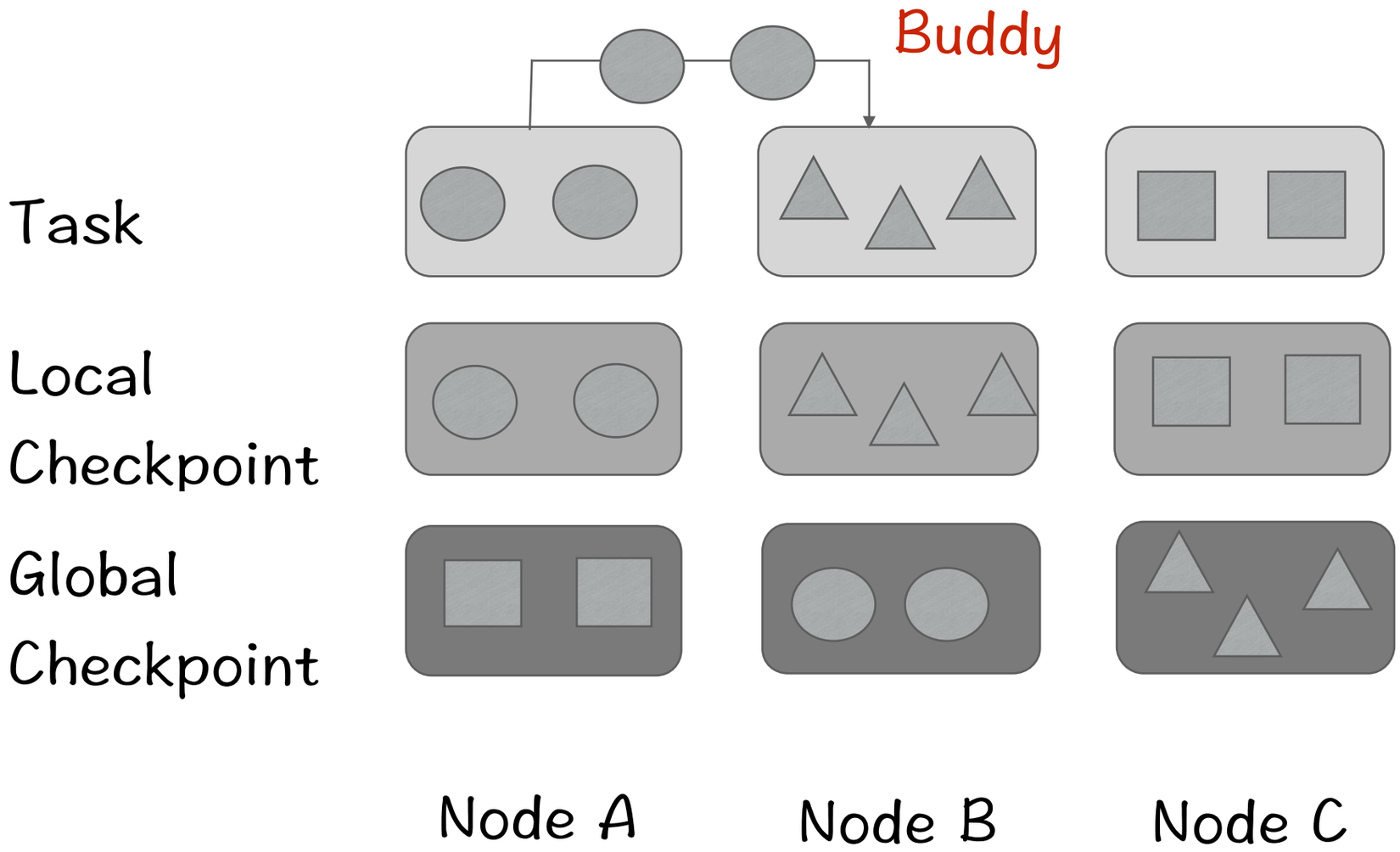
Double In-memory Checkpointing



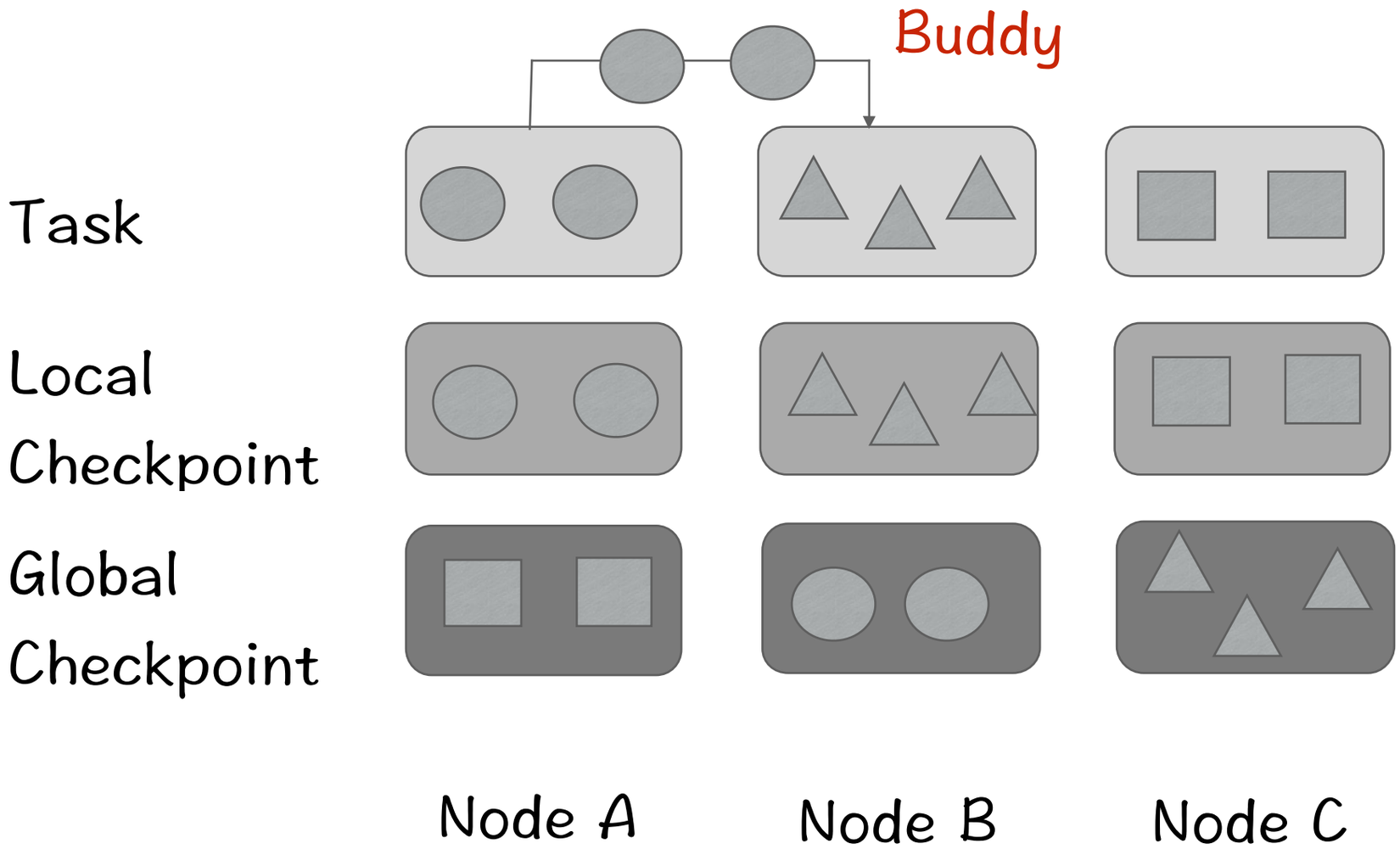
Double In-memory Checkpointing



Double In-memory Checkpointing

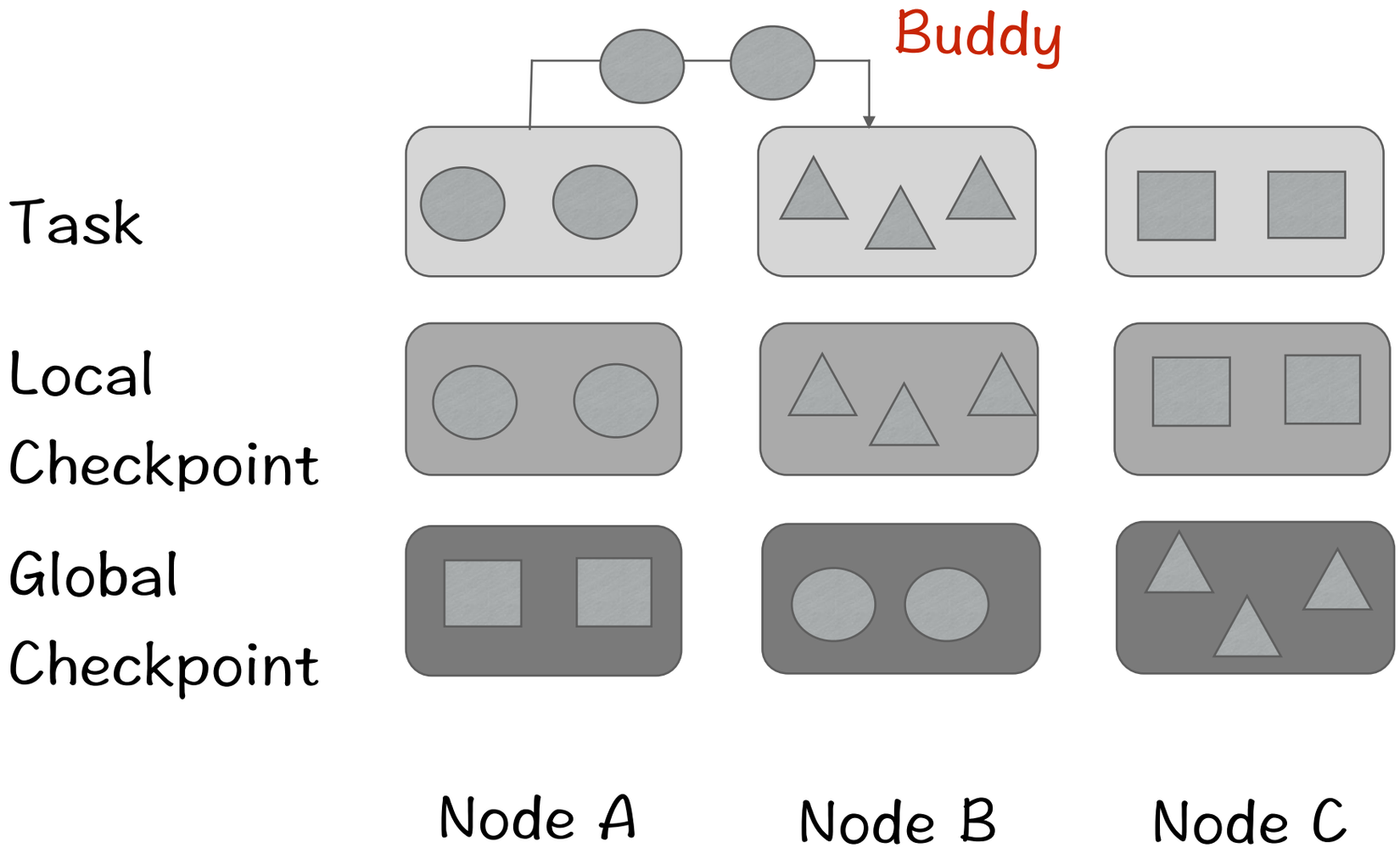


Double In-memory Checkpointing



- Charm++
- SCR
- FTI

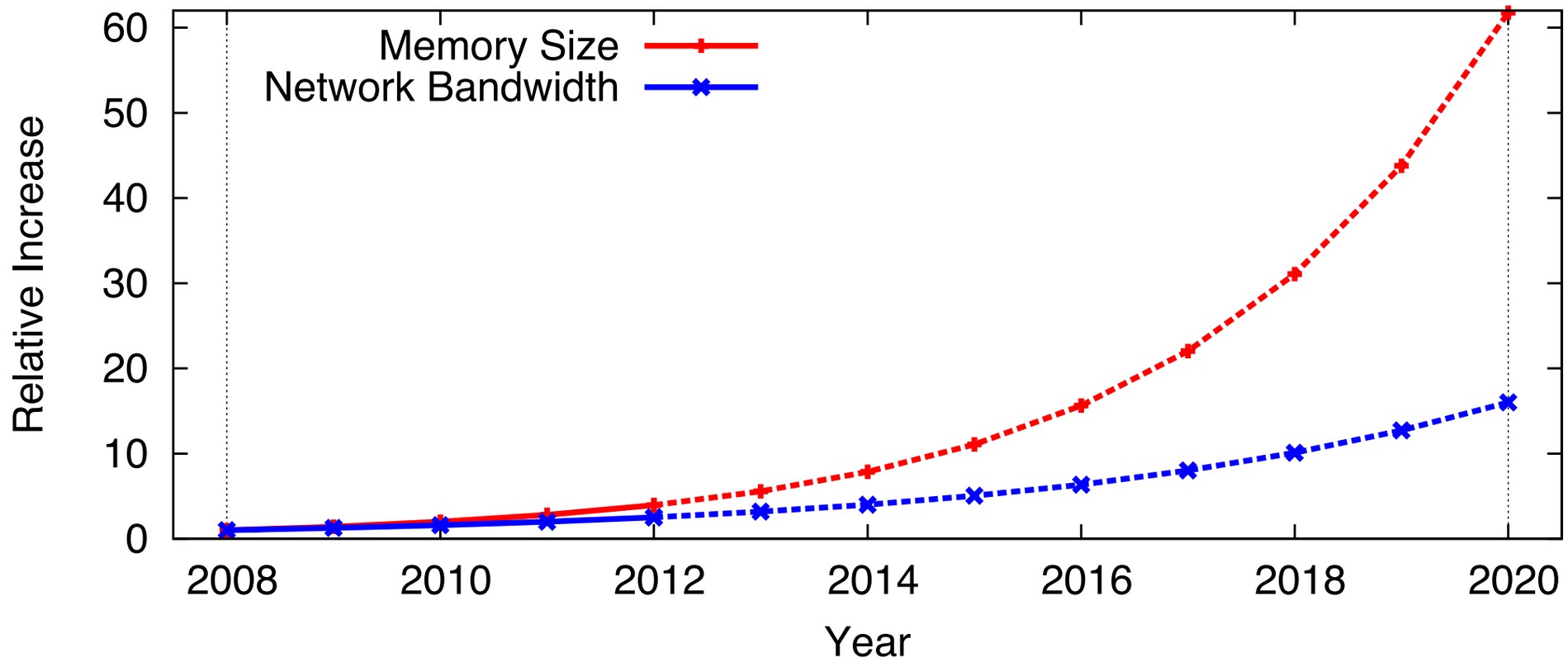
Double In-memory Checkpointing



- Charm++
- SCR
- FTI

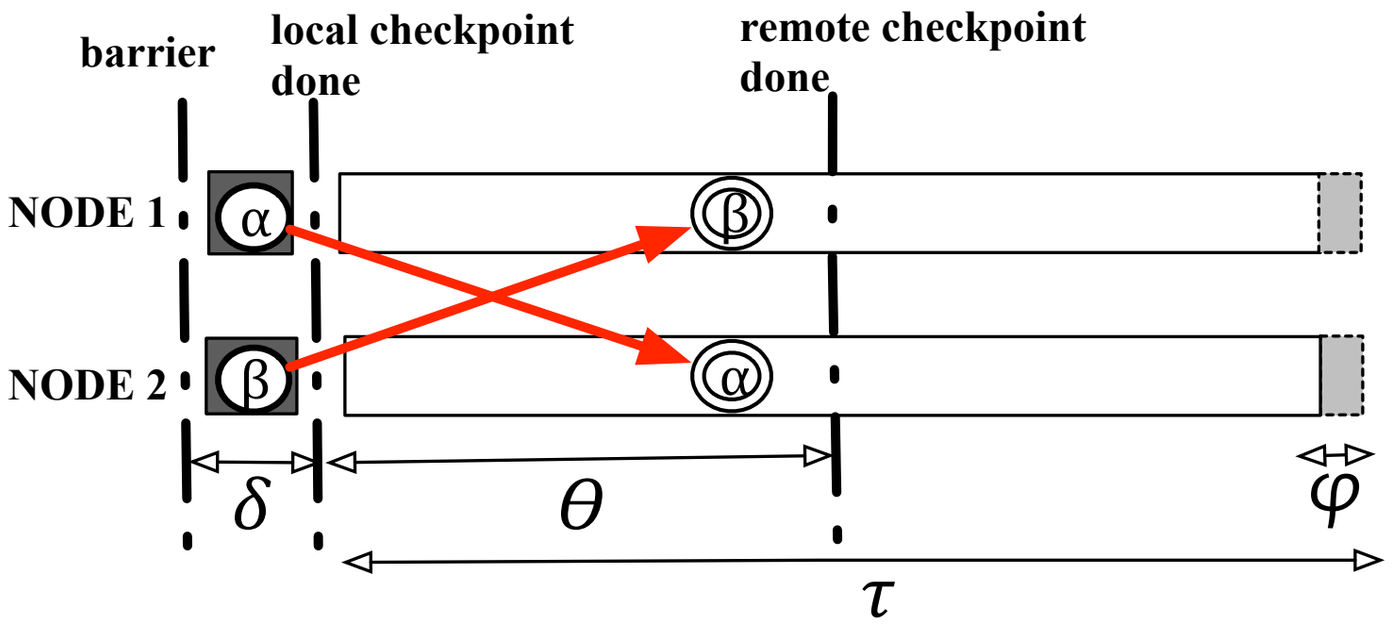
Application resumes computation after all the nodes have successfully saved the checkpoints in their buddy nodes.

Limitation of Checkpoint/Restart



- Increase in memory size per year: 41%
- Increase in network bandwidth per year: 26%

Semi-Blocking Checkpoint

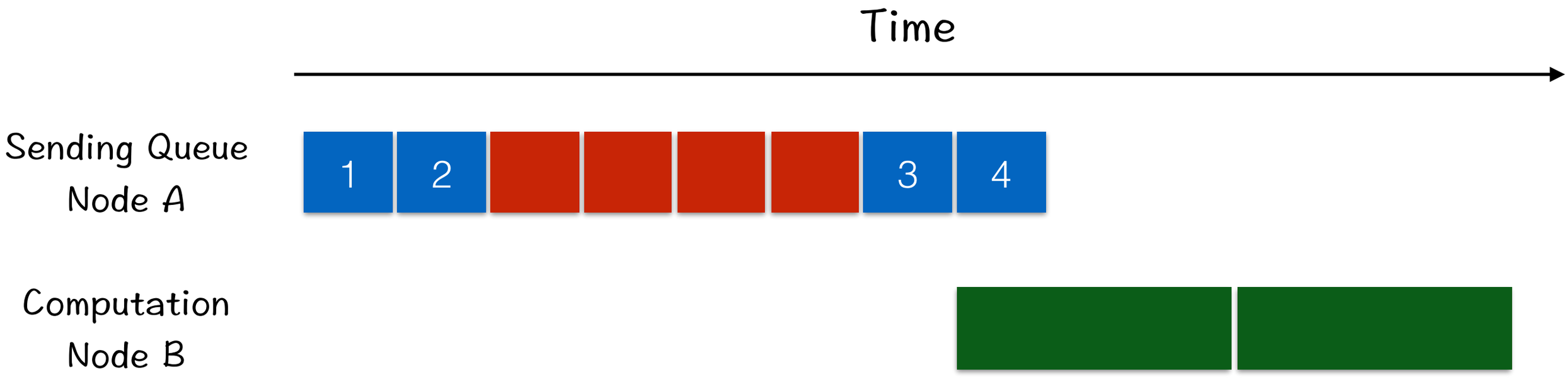


τ	checkpoint interval
δ	local checkpoint overhead
θ	overlap period
φ	remote checkpoint interference

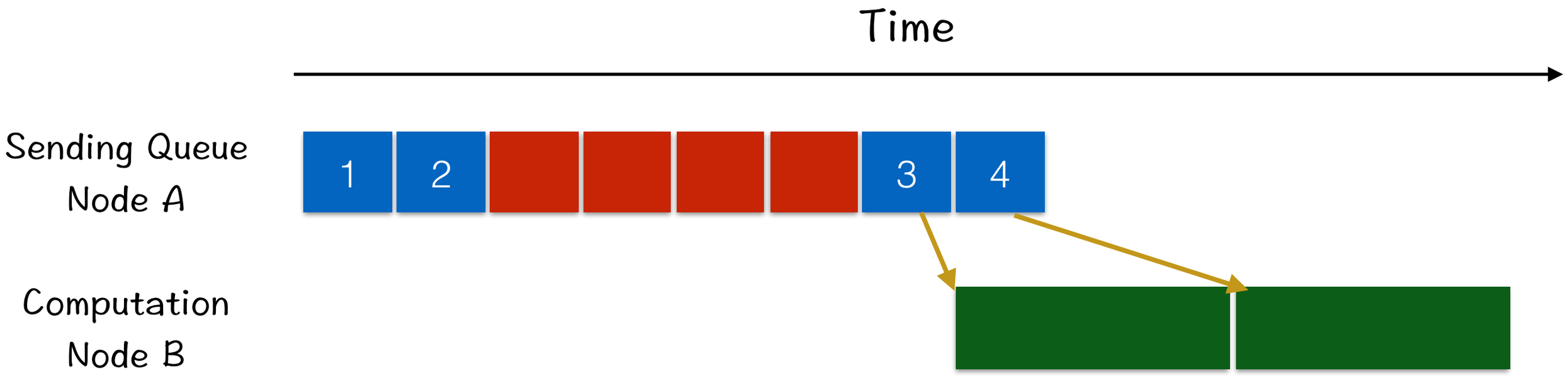
- Resume computation as soon as each node stores its own checkpoint (local checkpoint).
- Interleave the transmission of the checkpoint to buddy with application execution (remote checkpoint).

CLUSTER12: Hiding Checkpoint Overhead in HPC Applications with a Semi-Blocking Algorithm

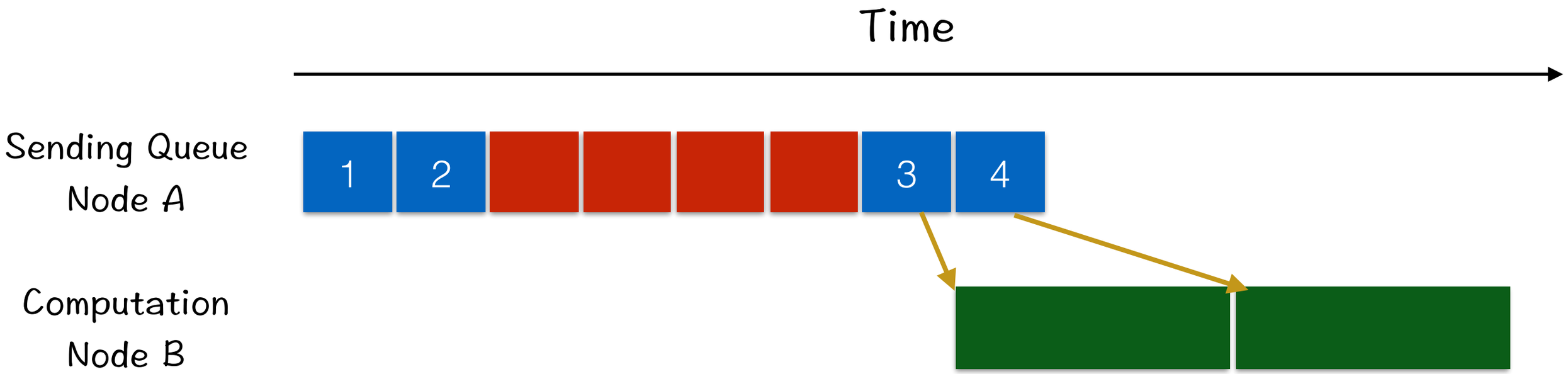
Minimize Checkpoint Interference



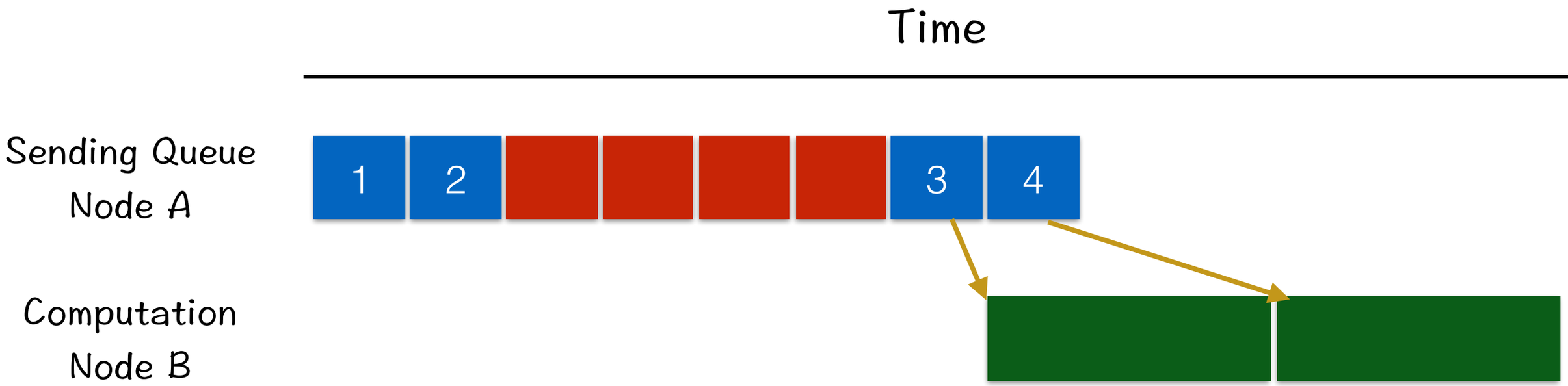
Minimize Checkpoint Interference

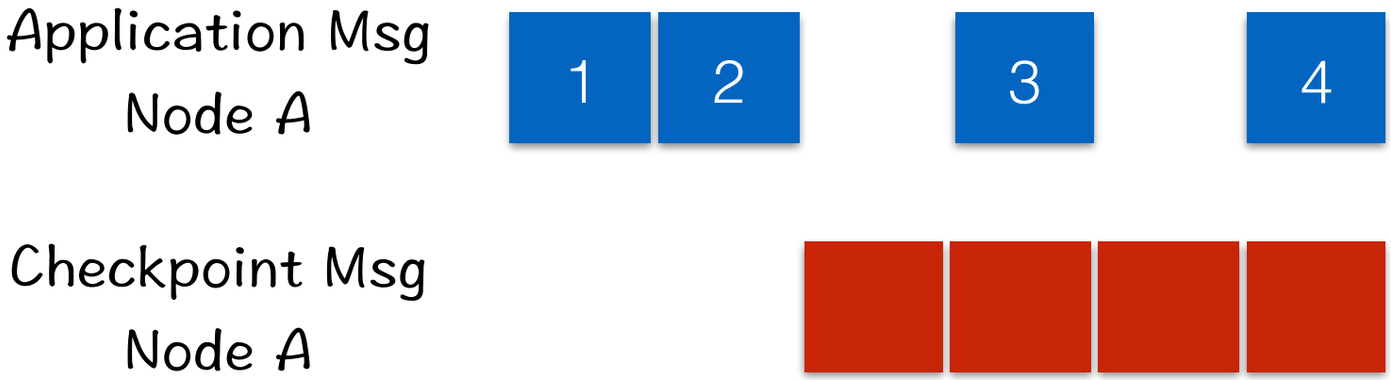


Minimize Checkpoint Interference

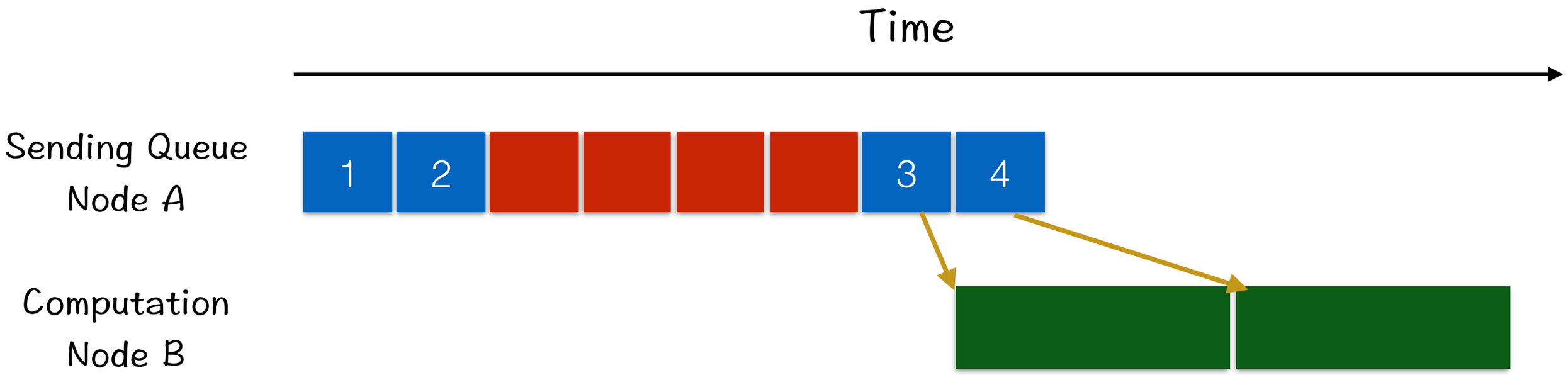


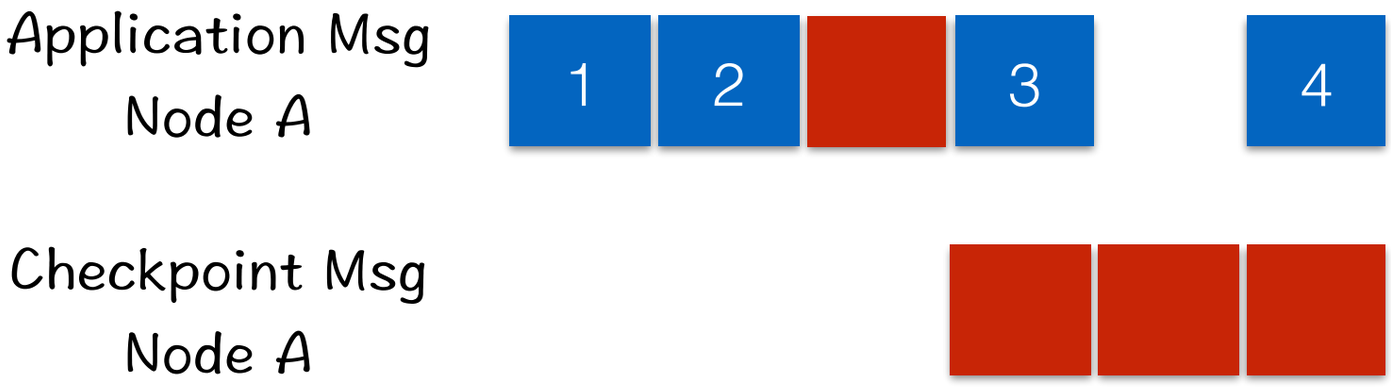
Minimize Checkpoint Interference



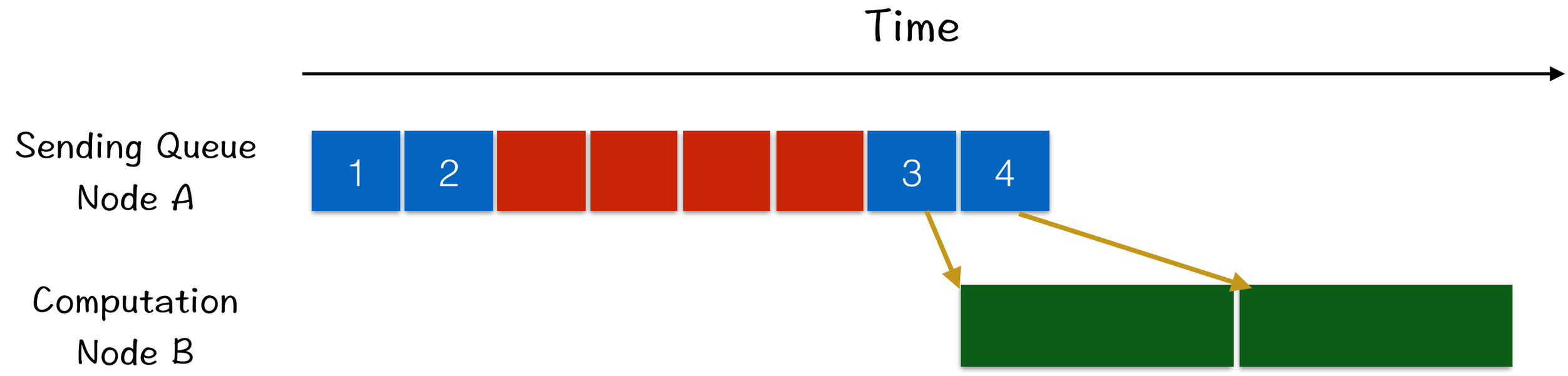


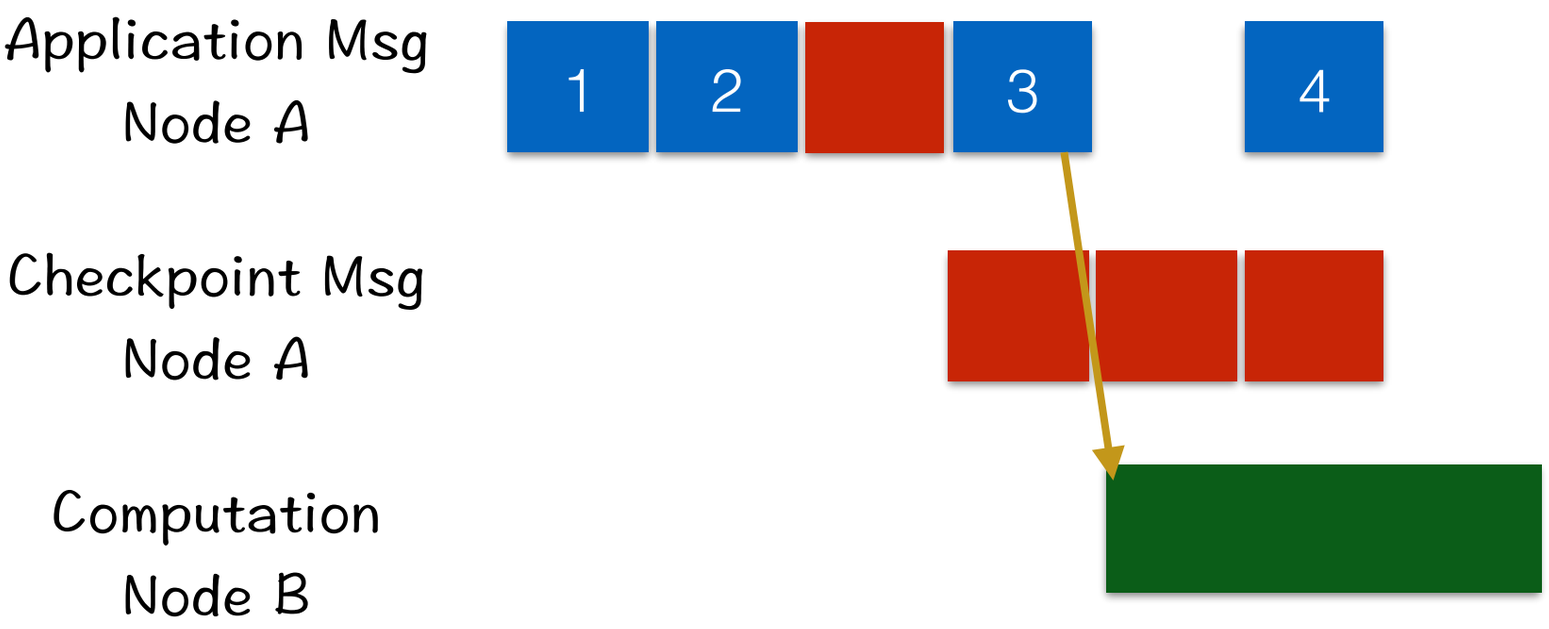
Minimize Checkpoint Interference



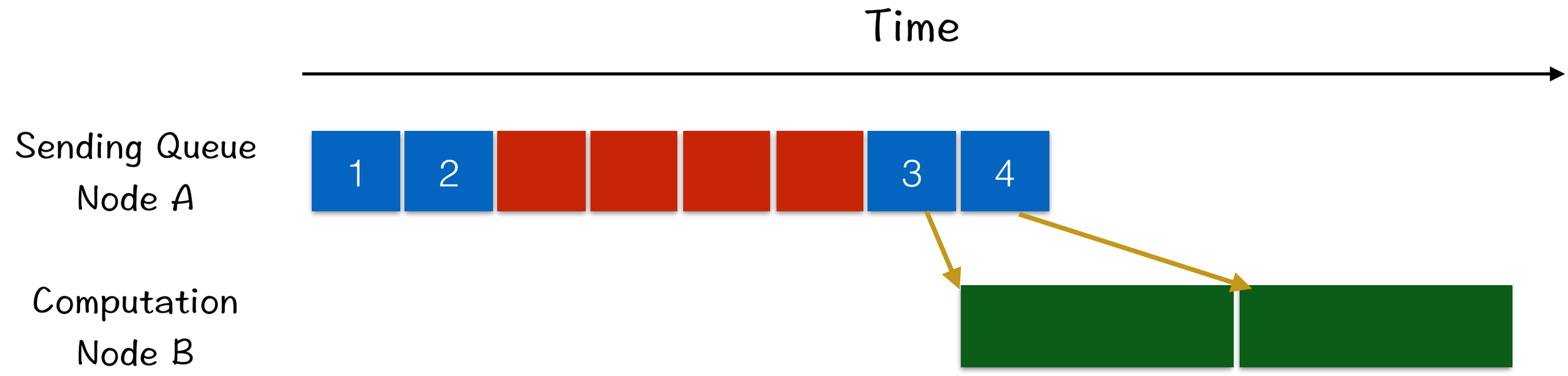


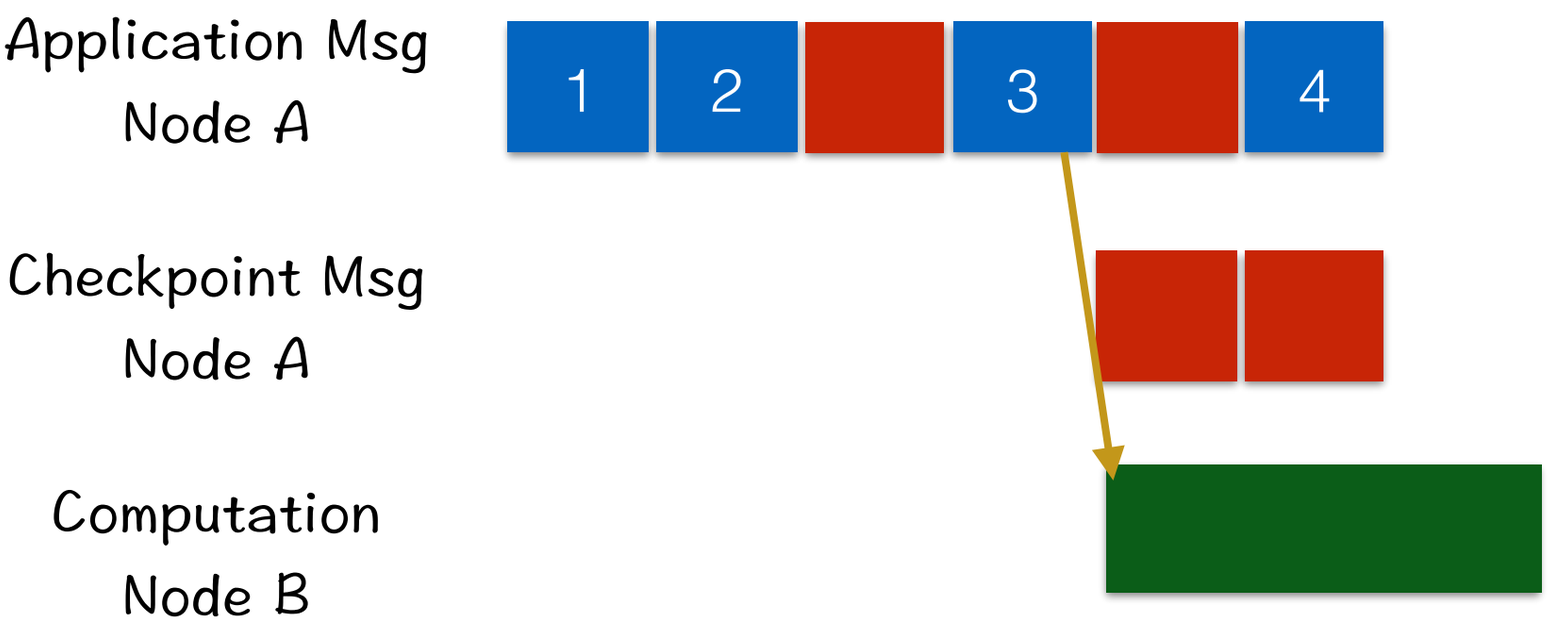
Minimize Checkpoint Interference



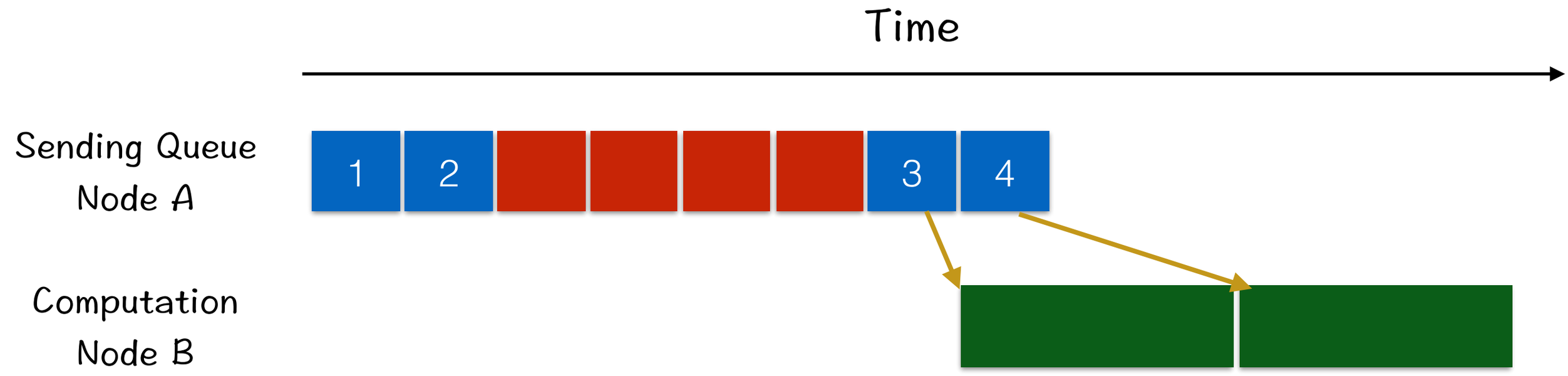


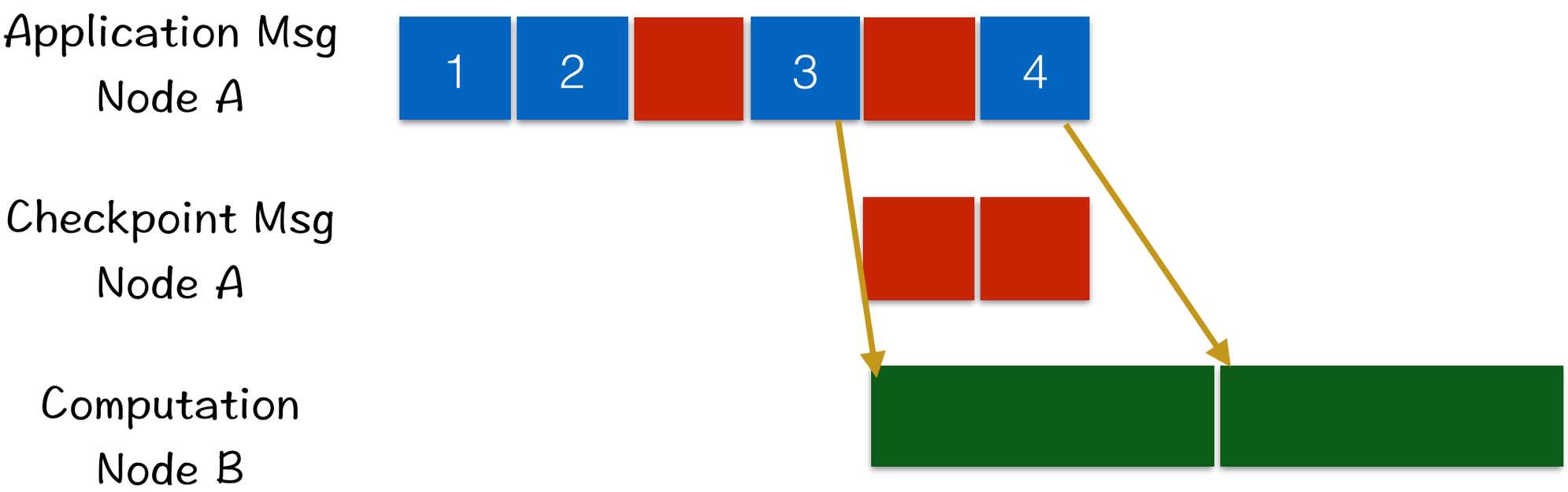
Minimize Checkpoint Interference



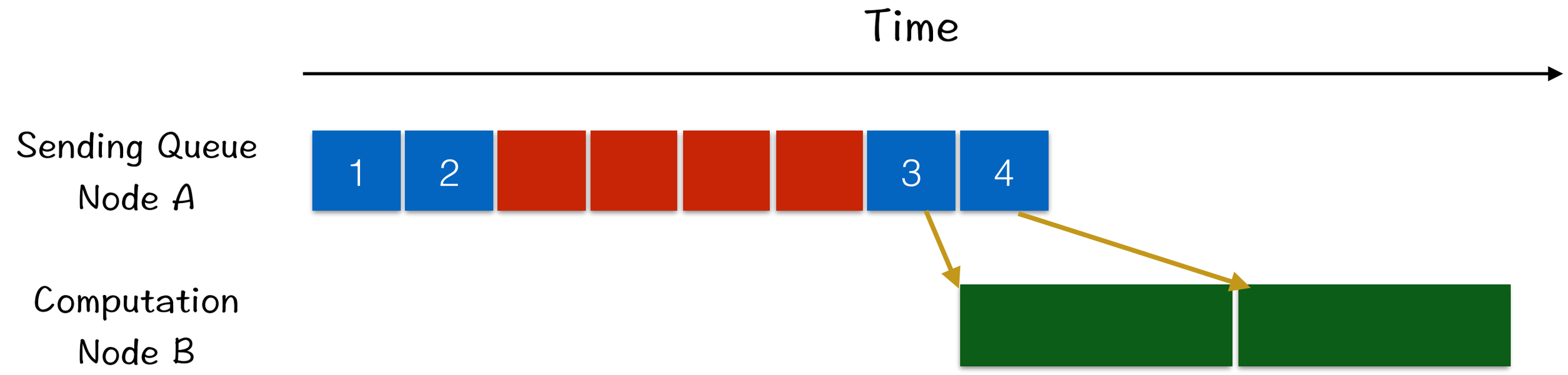


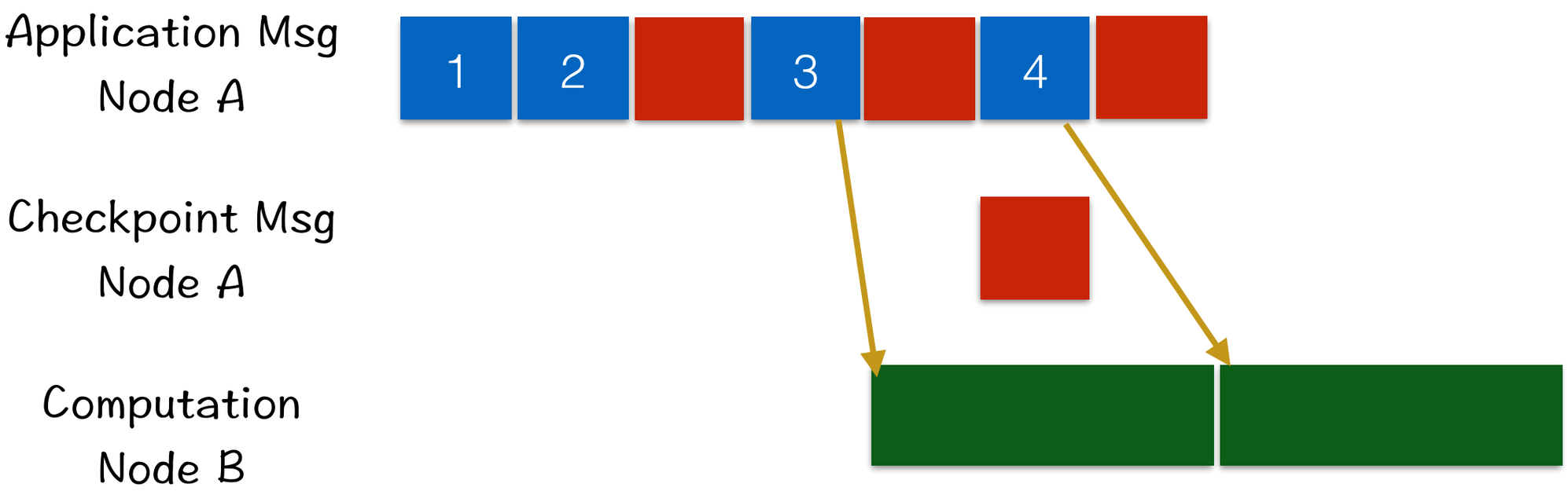
Minimize Checkpoint Interference



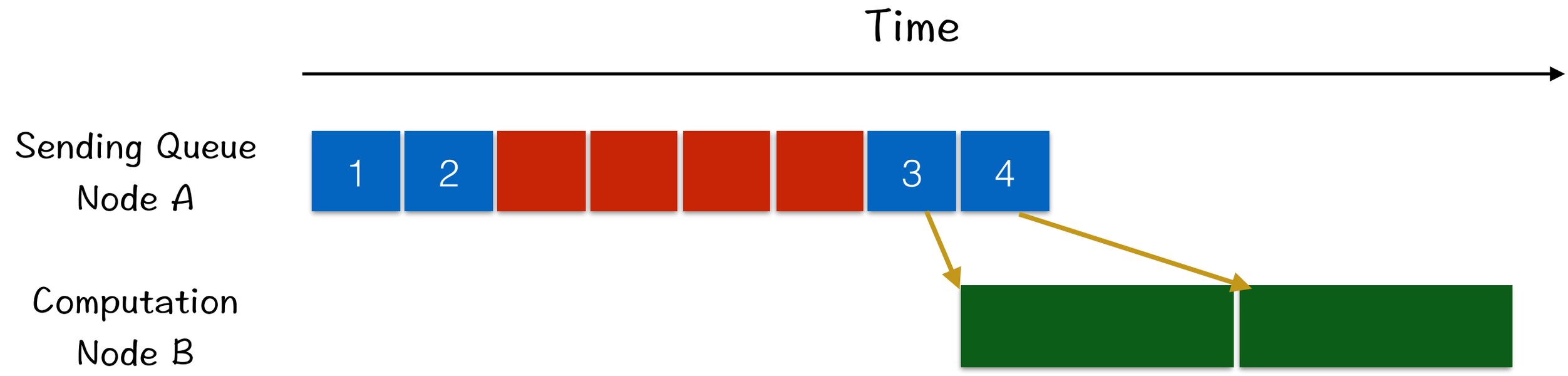


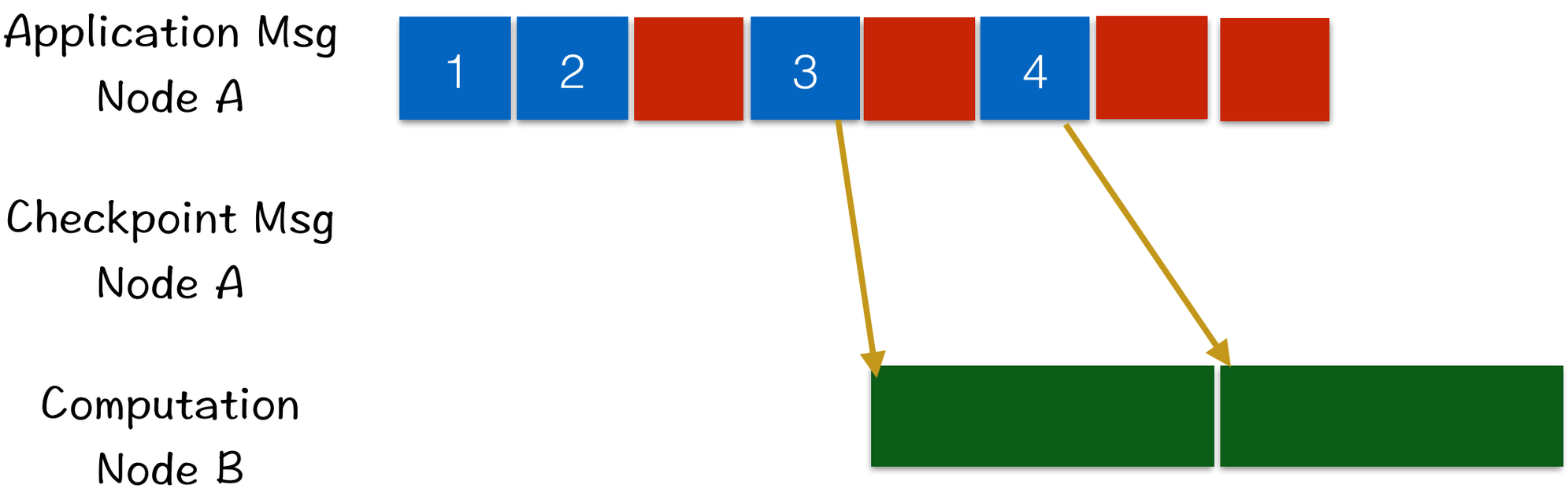
Minimize Checkpoint Interference





Minimize Checkpoint Interference



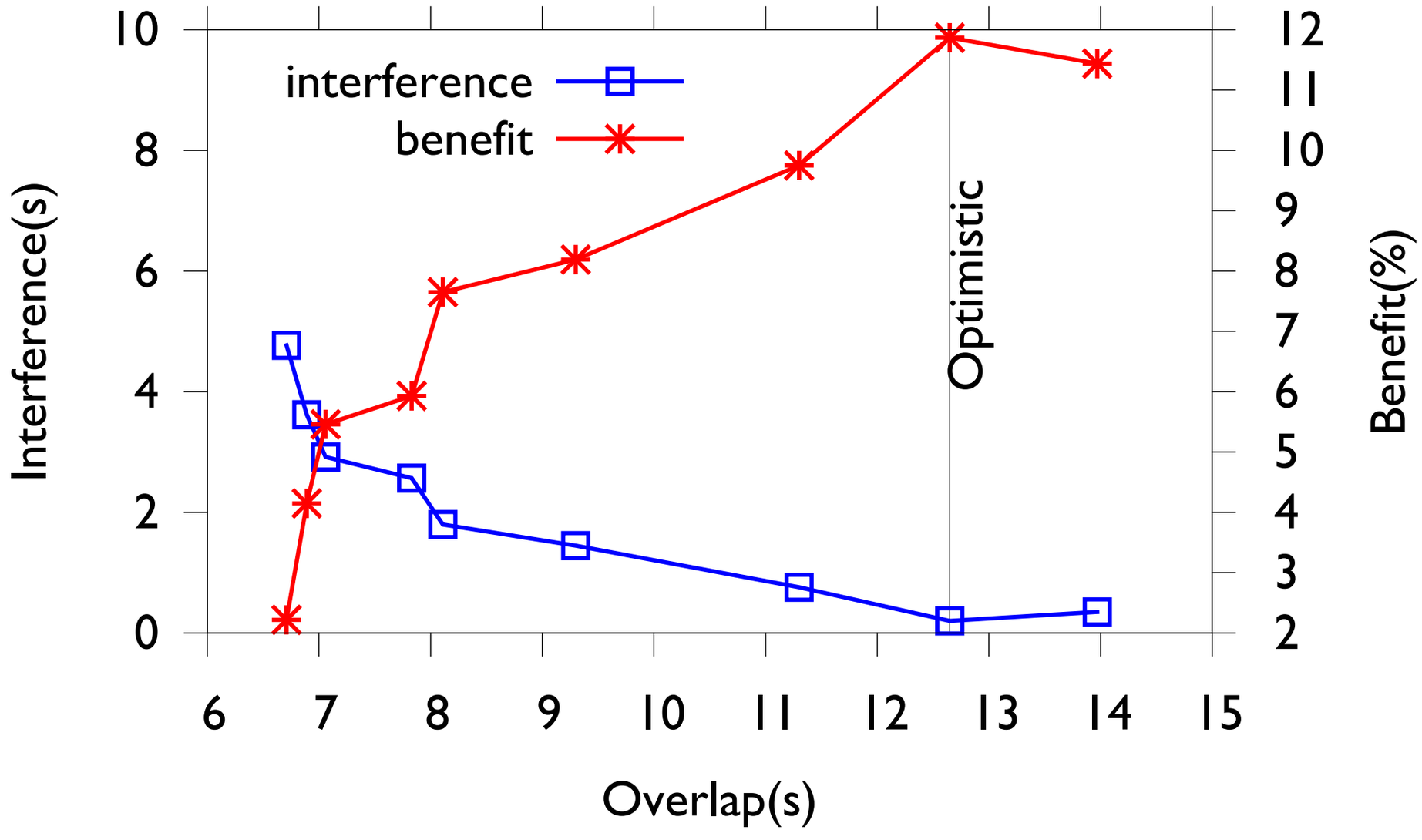


Optimistic Scheduling

Overlap Period ↑ more rework

Overlap Period ↓ more interference to applications

Random Scheduling: based on pre-defined overlap period, probabilistic deciding whether application or checkpoint queue can send message.

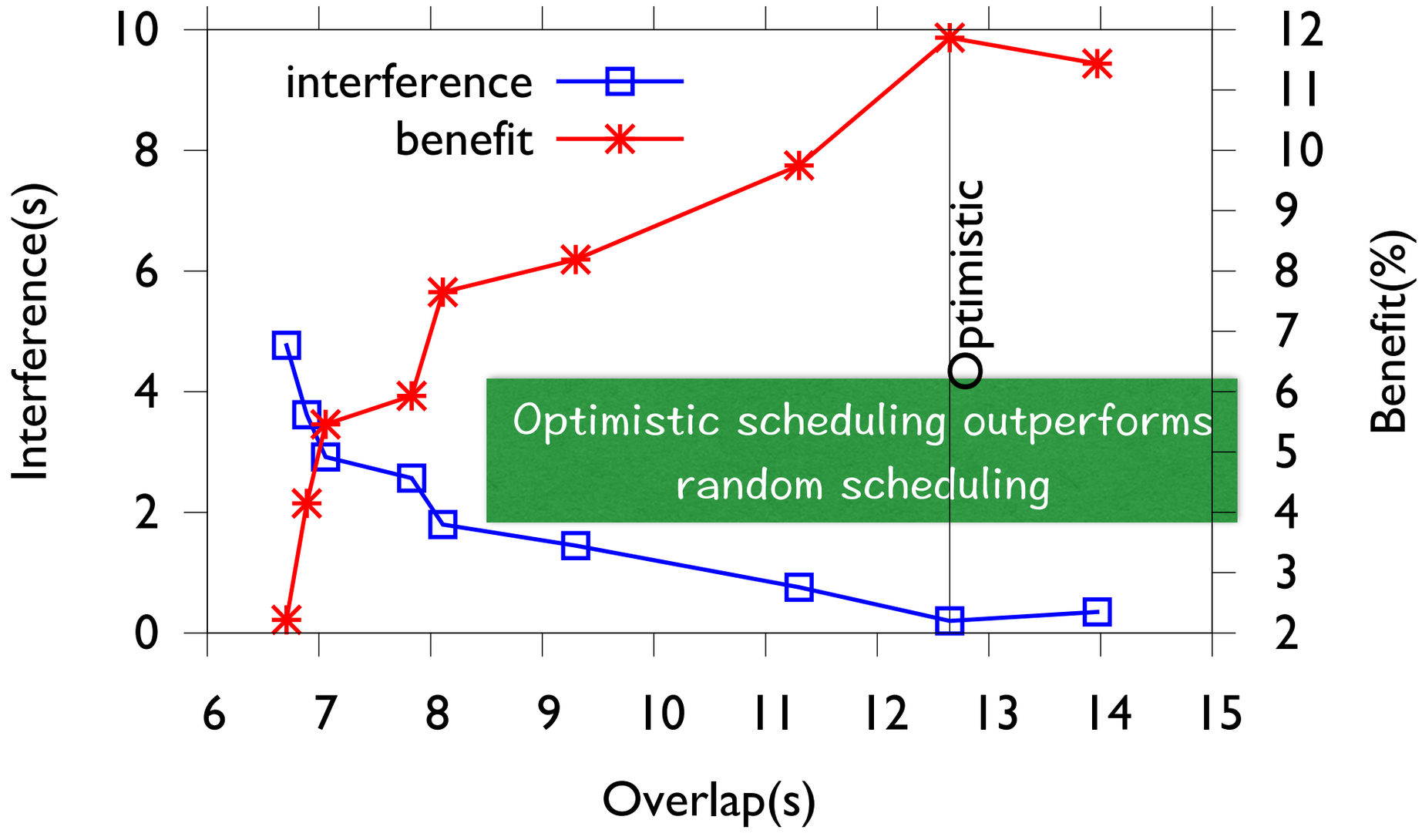


Optimistic Scheduling

Overlap Period ↑ more rework

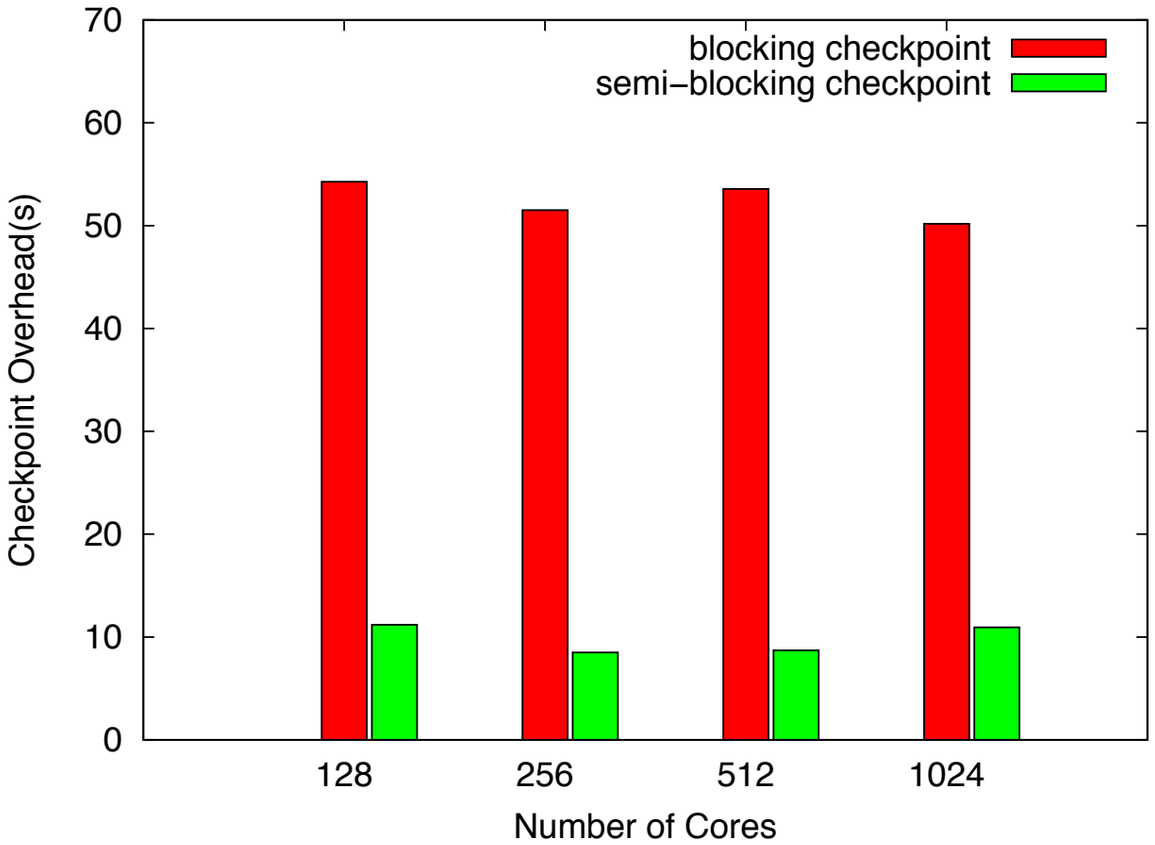
Overlap Period ↓ more interference to applications

Random Scheduling: based on pre-defined overlap period, probabilistic deciding whether application or checkpoint queue can send message.

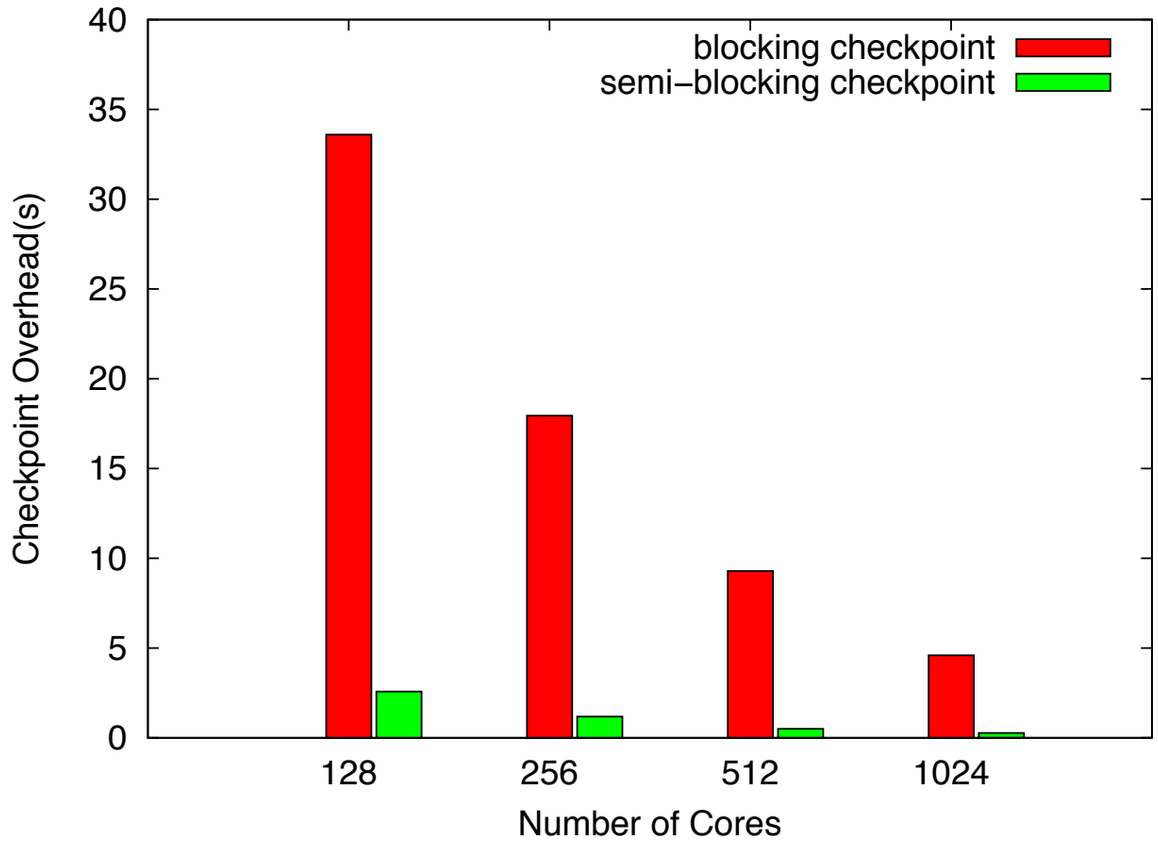


Single Checkpoint Overhead

Trestles@SDSC



Wave2D Weak Scaling



ChaNGa Strong Scaling

Semi-Blocking checkpoint reduces checkpoint overhead by 5 times.

Automatic Checkpoint

Automatic Checkpoint

- Optimal Checkpoint Interval

Automatic Checkpoint

- Optimal Checkpoint Interval
  - ✓ Traditionally, application checkpoints at fixed interval.

Automatic Checkpoint

- Optimal Checkpoint Interval
  - ✓ Traditionally, application checkpoints at fixed interval.
  - ✓ If MTBF follows Poisson process, Daly's model

Automatic Checkpoint

- Optimal Checkpoint Interval
  - ✓ Traditionally, application checkpoints at fixed interval.
  - ✓ If MTBF follows Poisson process, Daly's model

$$\tau_{\text{opt}} = \sqrt{2\delta(M+R)} \quad \text{for } \tau + \delta \ll M.$$

Automatic Checkpoint

- Optimal Checkpoint Interval
  - ✓ Traditionally, application checkpoints at fixed interval.
  - ✓ If MTBF follows Poisson process, Daly's model

$$\tau_{opt} = \sqrt{2\delta(M+R)} \quad \text{for } \tau + \delta \ll M.$$

  - ✓ What if MTBF follows Weibull process, increasing or decreasing failure rate?

Automatic Checkpoint

- Optimal Checkpoint Interval
  - ✓ Traditionally, application checkpoints at fixed interval.
  - ✓ If MTBF follows Poisson process, Daly's model

$$\tau_{opt} = \sqrt{2\delta(M+R)} \quad \text{for } \tau + \delta \ll M.$$

  - ✓ What if MTBF follows Weibull process, increasing or decreasing failure rate?
  - ✓ How can applications automatically take preventative checkpoint based on failure prediction?

Automatic Checkpoint

- Optimal Checkpoint Interval
  - ✓ Traditionally, application checkpoints at fixed interval.
  - ✓ If MTBF follows Poisson process, Daly's model

$$\tau_{opt} = \sqrt{2\delta(M+R)} \quad \text{for } \tau + \delta \ll M.$$

  - ✓ What if MTBF follows Weibull process, increasing or decreasing failure rate?
  - ✓ How can applications automatically take preventative checkpoint based on failure prediction?
  - ✖ Too expensive for HPC applications to synchronize very often for consistent checkpoint.

Automatic Checkpoint

- Optimal Checkpoint Interval
  - ✓ Traditionally, application checkpoints at fixed interval.
  - ✓ If MTBF follows Poisson process, Daly's model

$$\tau_{opt} = \sqrt{2\delta(M+R)} \quad \text{for } \tau + \delta \ll M.$$

  - ✓ What if MTBF follows Weibull process, increasing or decreasing failure rate?
  - ✓ How can applications automatically take preventative checkpoint based on failure prediction?
  - Too expensive for HPC applications to synchronize very often for consistent checkpoint.
  - Otherwise applications may hang due to inconsistent checkpoint.

Automatic Checkpoint

- Optimal Checkpoint Interval
  - ✓ Traditionally, application checkpoints at fixed interval.
  - ✓ If MTBF follows Poisson process, Daly's model

$$\tau_{opt} = \sqrt{2\delta(M+R)} \quad \text{for } \tau + \delta \ll M.$$

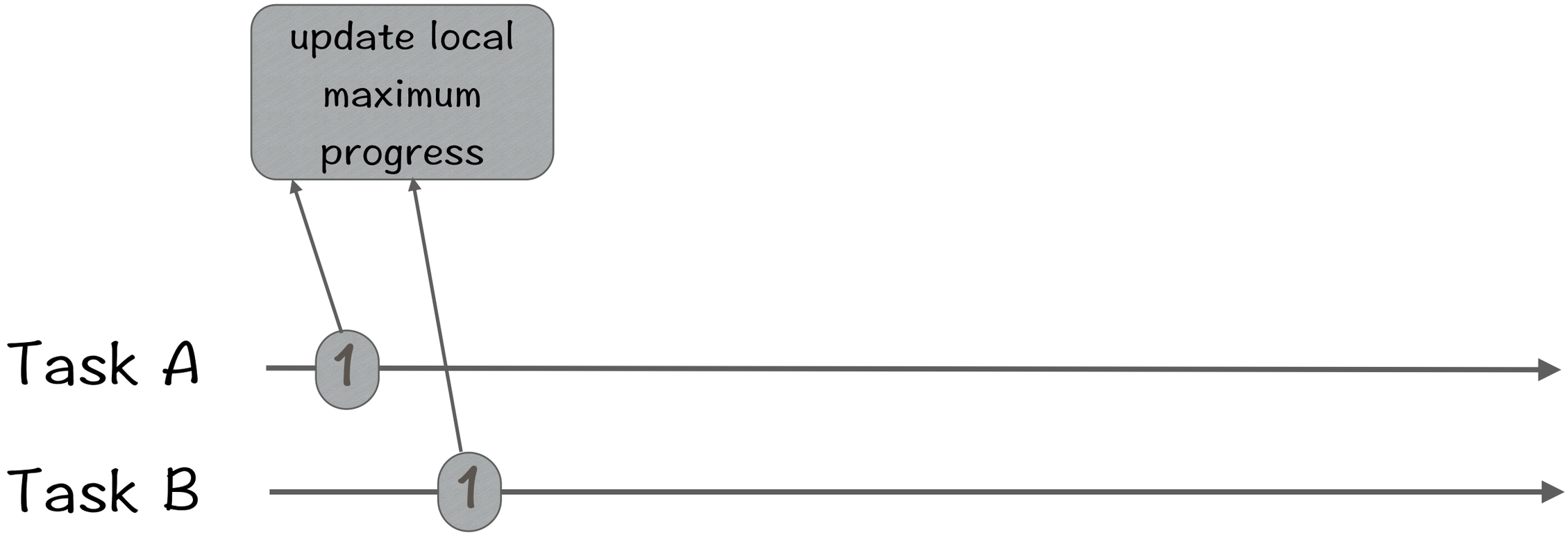
  - ✓ What if MTBF follows Weibull process, increasing or decreasing failure rate?
  - ✓ How can applications automatically take preventative checkpoint based on failure prediction?
  - Too expensive for HPC applications to synchronize very often for consistent checkpoint.
  - Otherwise applications may hang due to inconsistent checkpoint.

Runtime Support Automatic Checkpoint Decision

Automatic Checkpoint Decision

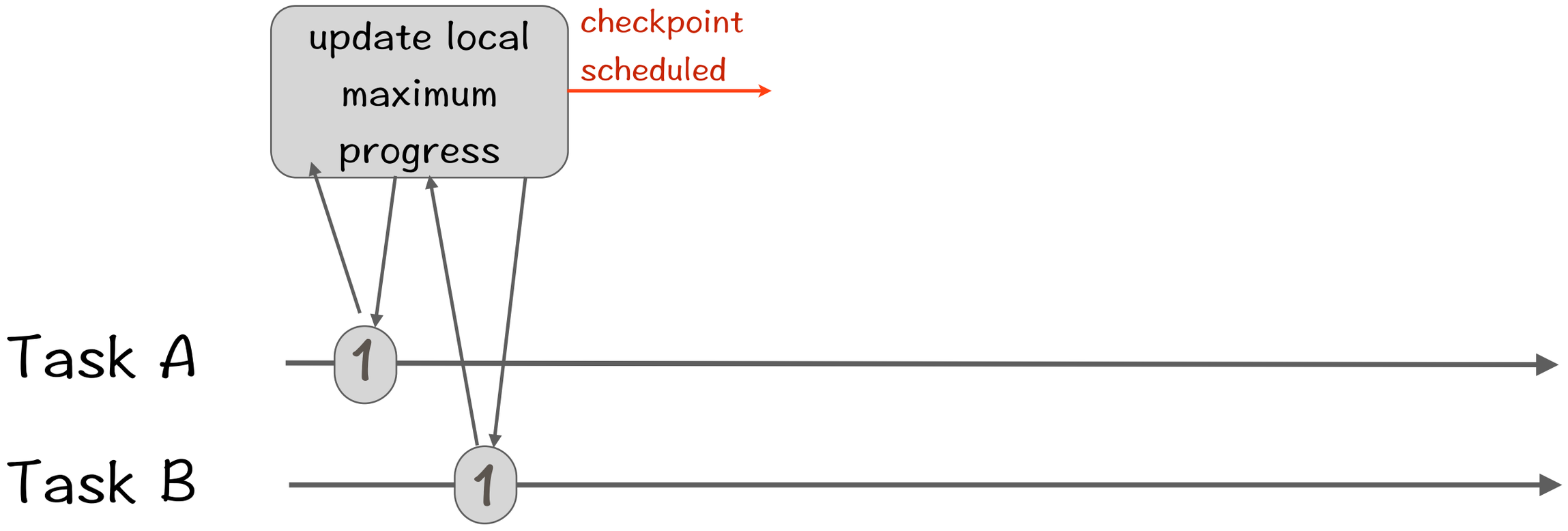
update local maximum progress

Automatic Checkpoint Decision

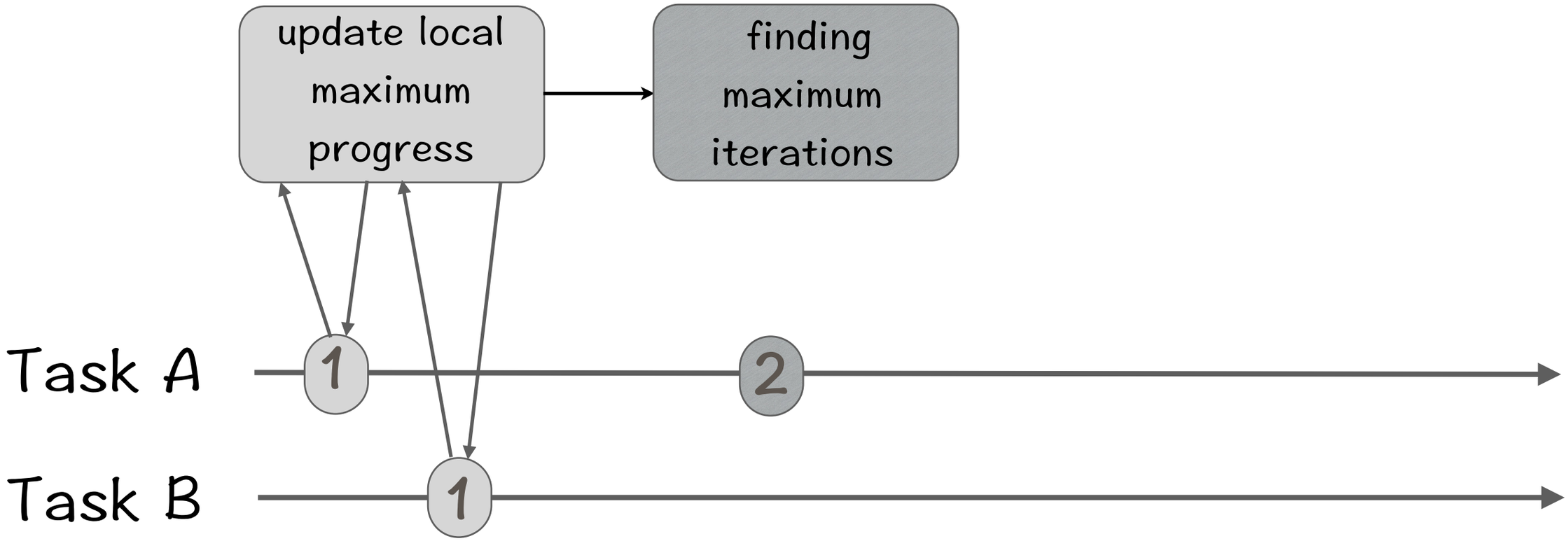


Automatic Checkpoint Decision

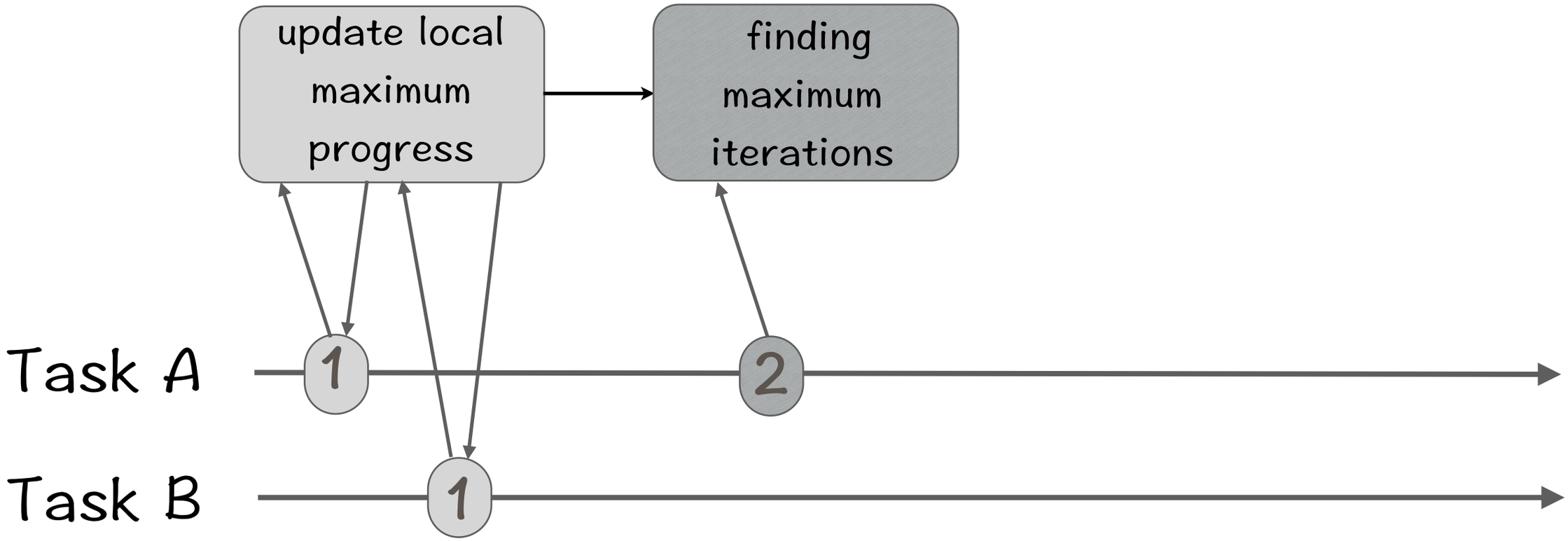
Automatic Checkpoint Decision



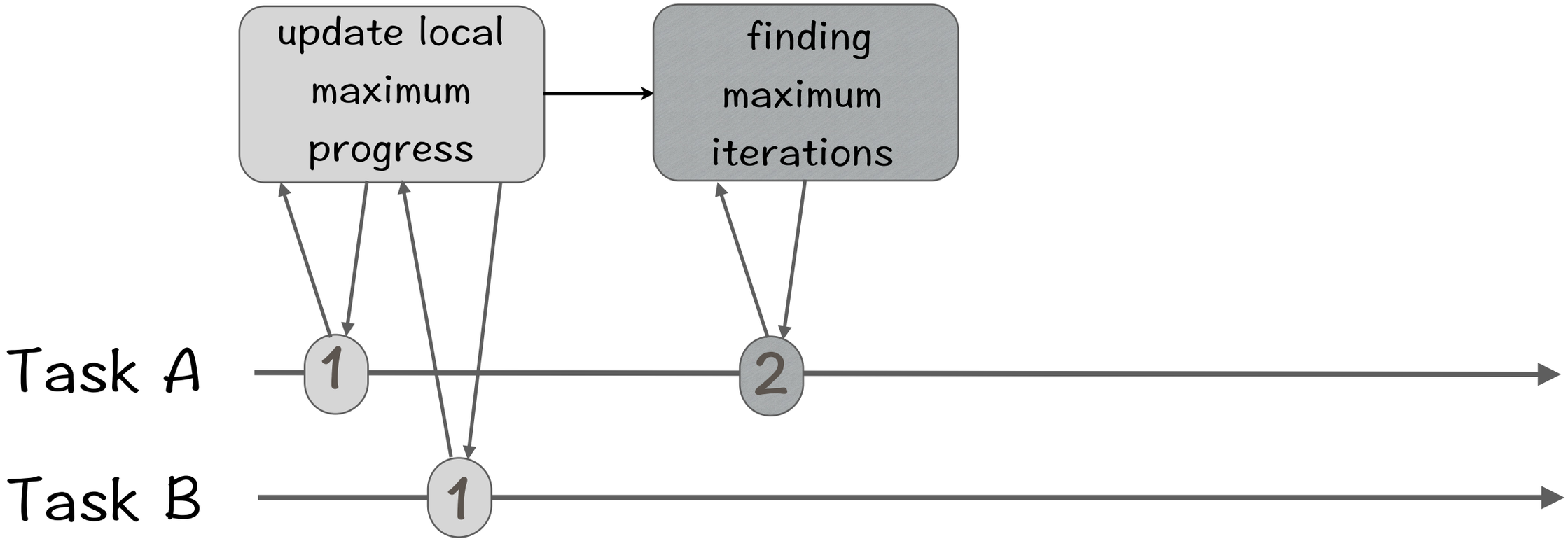
Automatic Checkpoint Decision



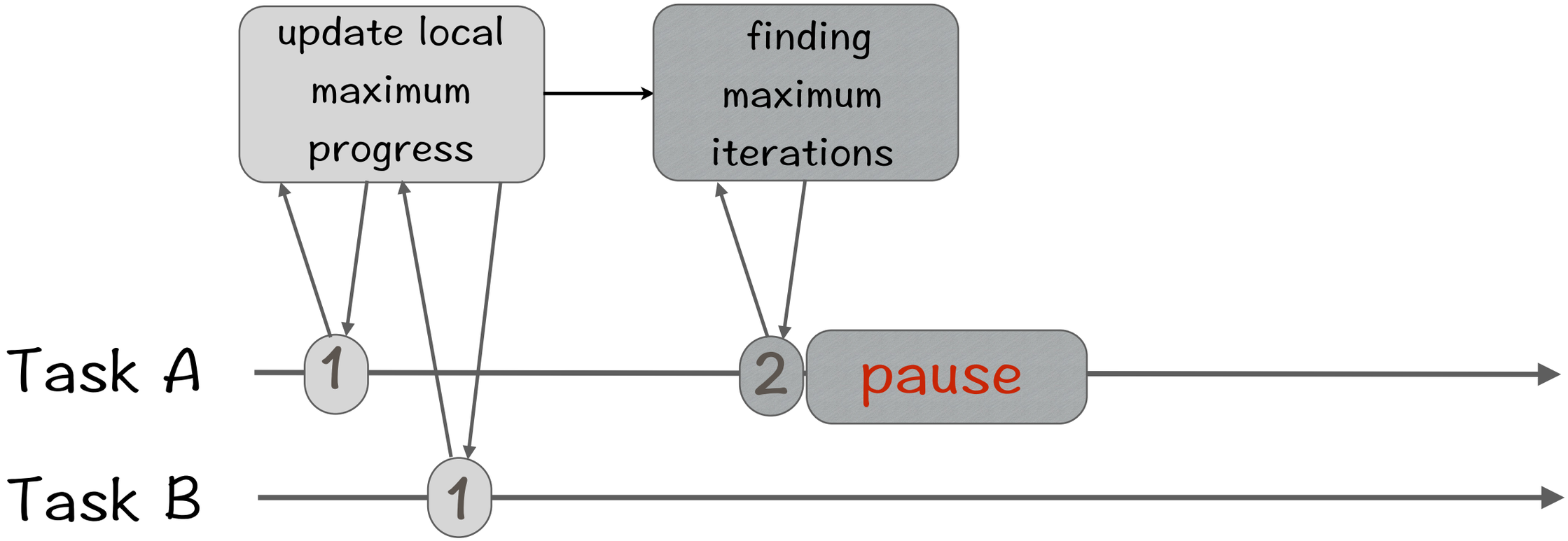
Automatic Checkpoint Decision



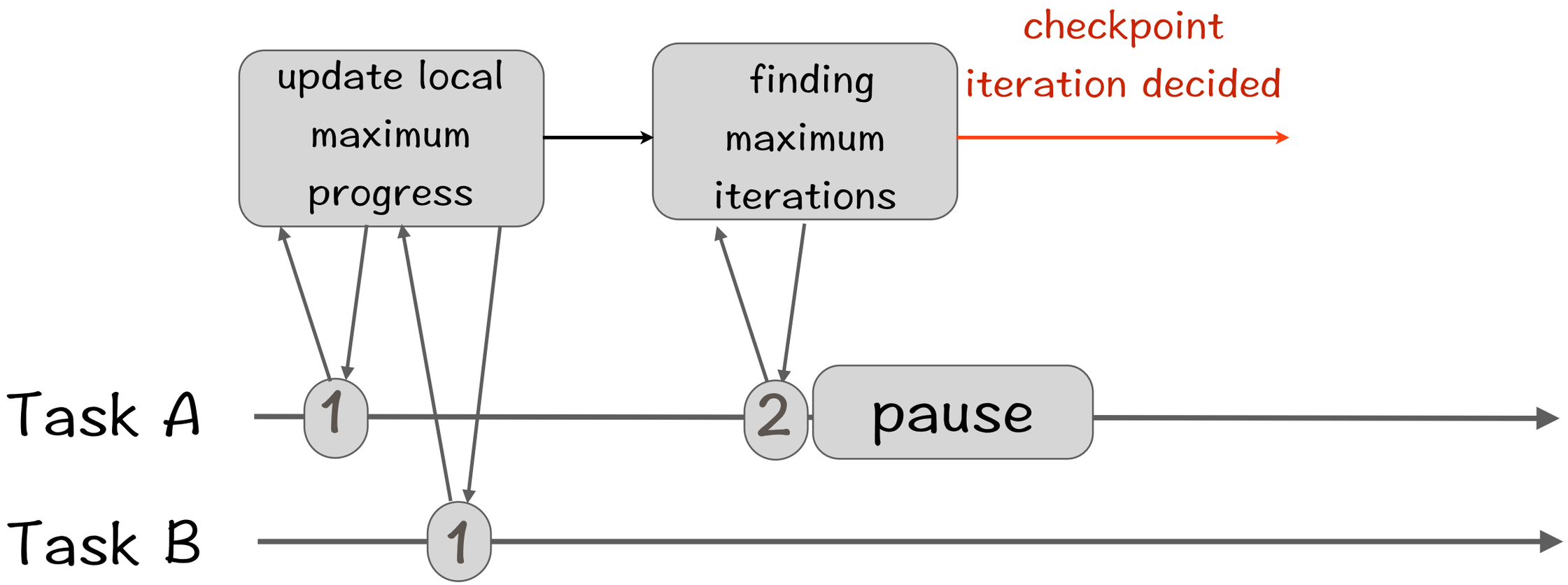
Automatic Checkpoint Decision



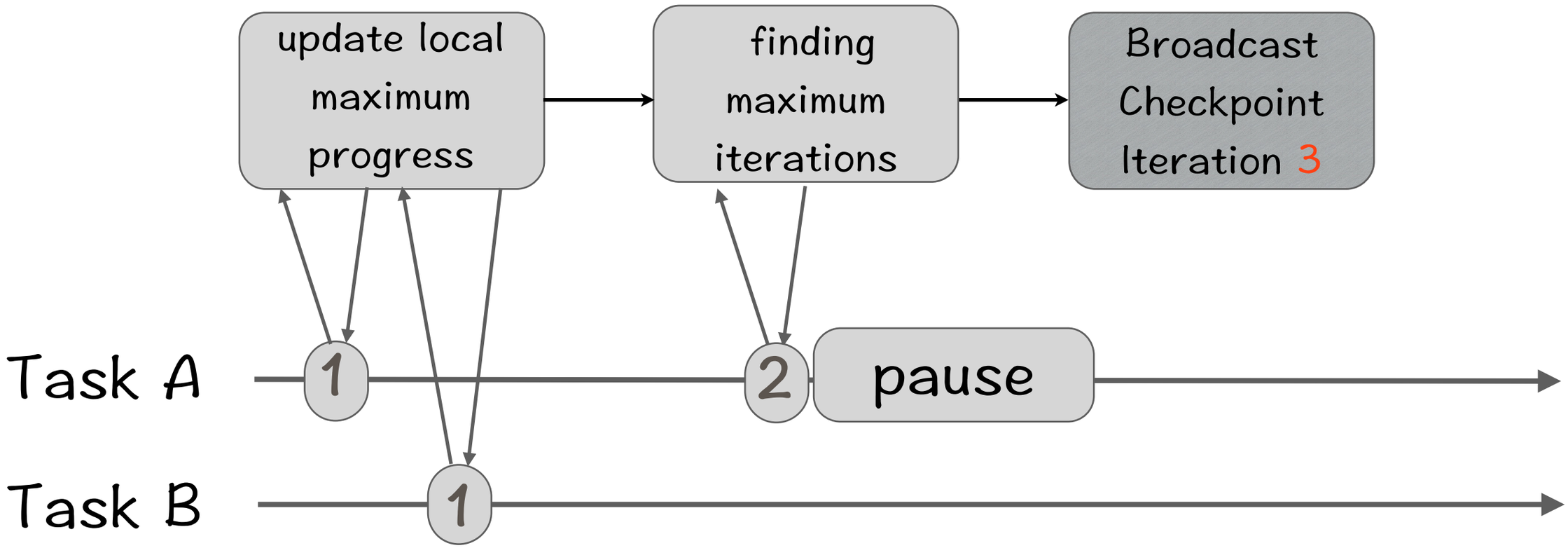
Automatic Checkpoint Decision



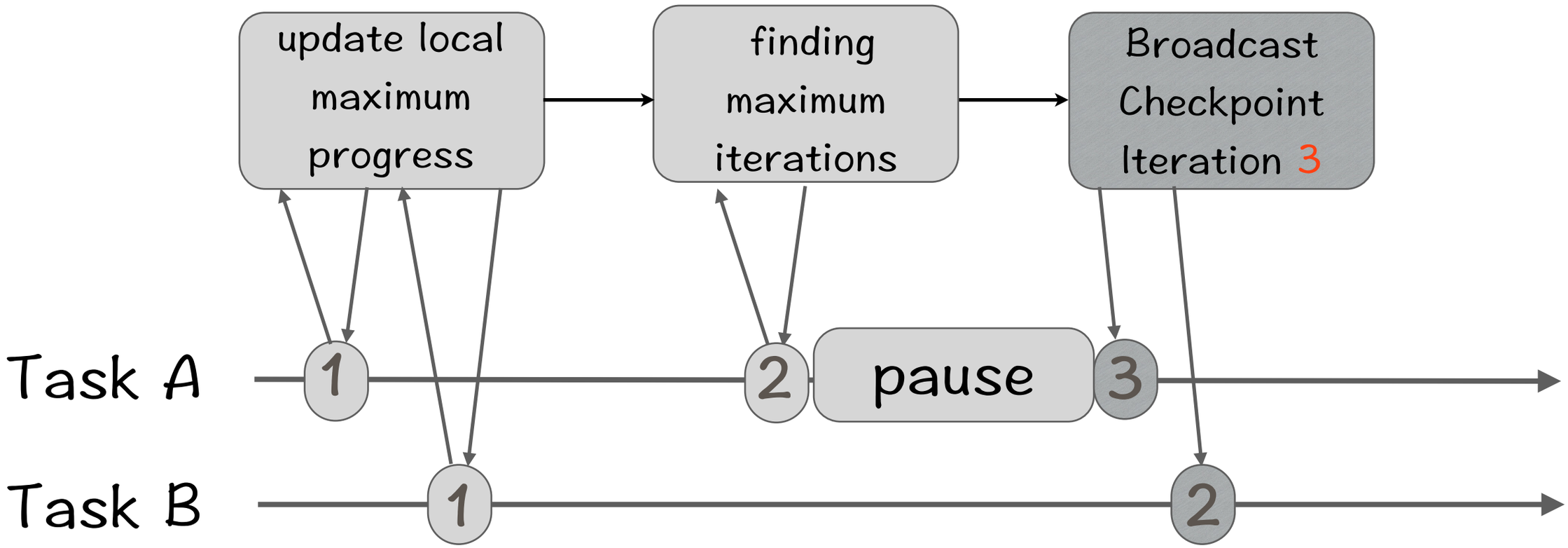
Automatic Checkpoint Decision



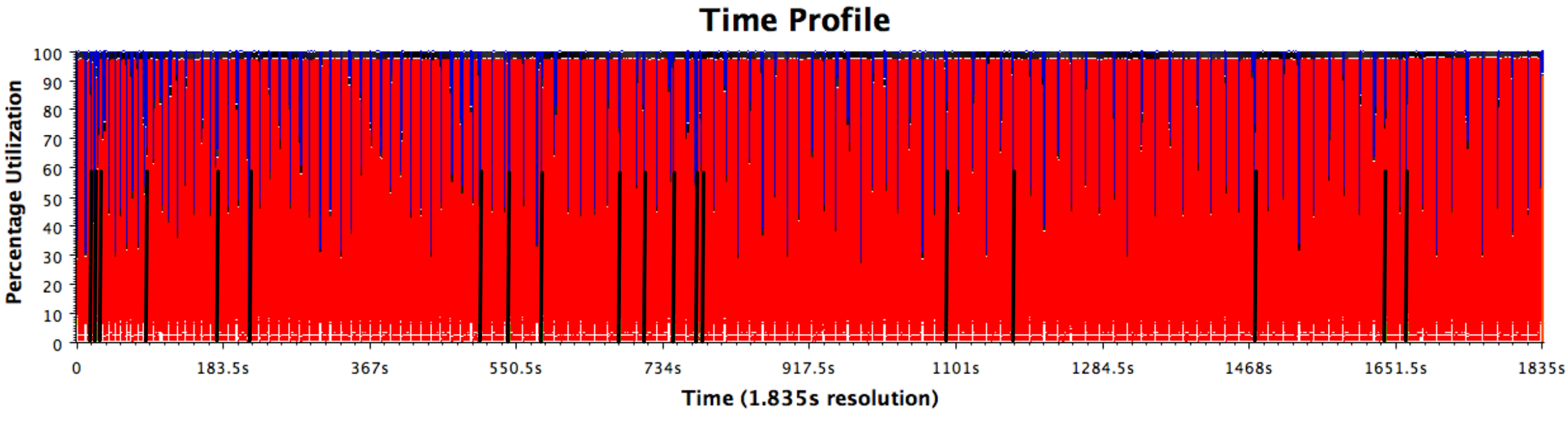
Automatic Checkpoint Decision



Automatic Checkpoint Decision

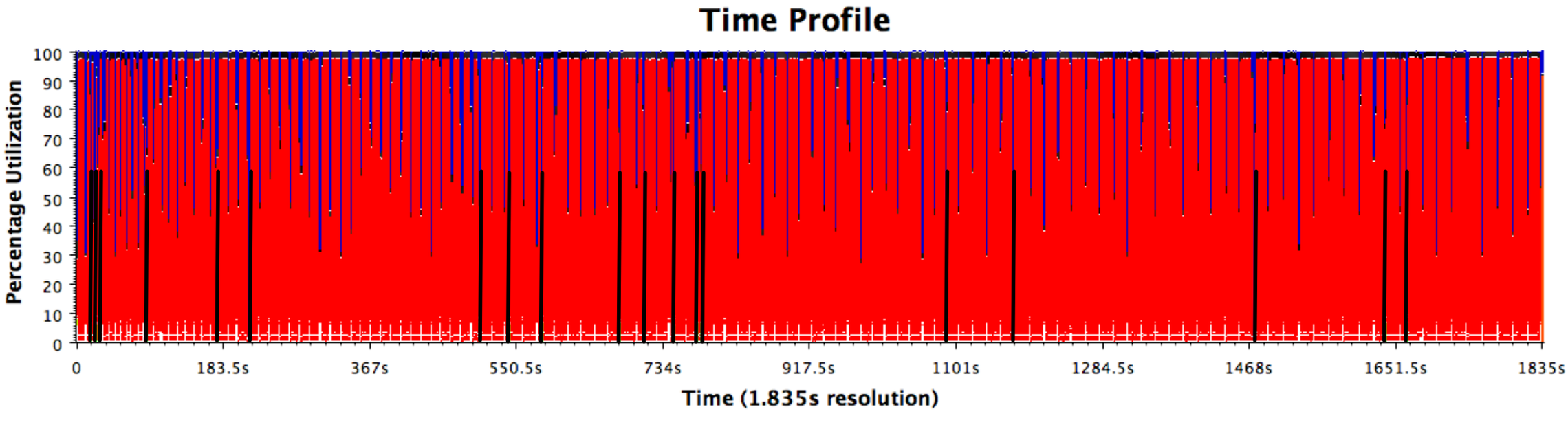


Adapting to Failures



- Failures are injected according to Weibull process: shape parameter 0.6
- Changing Checkpoint period

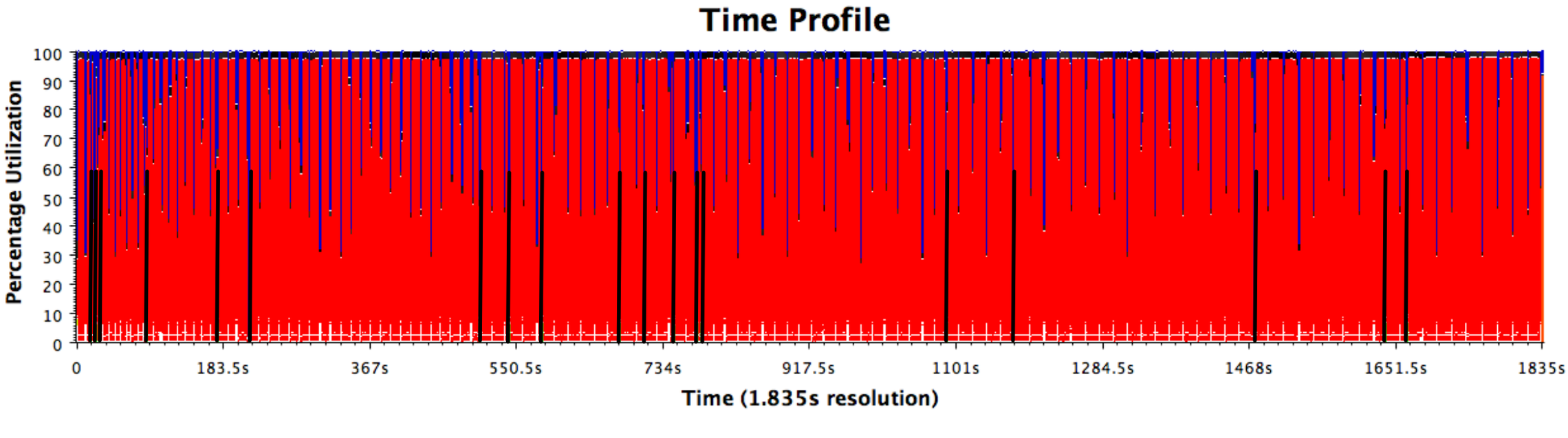
Adapting to Failures



6s

- Failures are injected according to Weibull process: shape parameter 0.6
- Changing Checkpoint period

Adapting to Failures

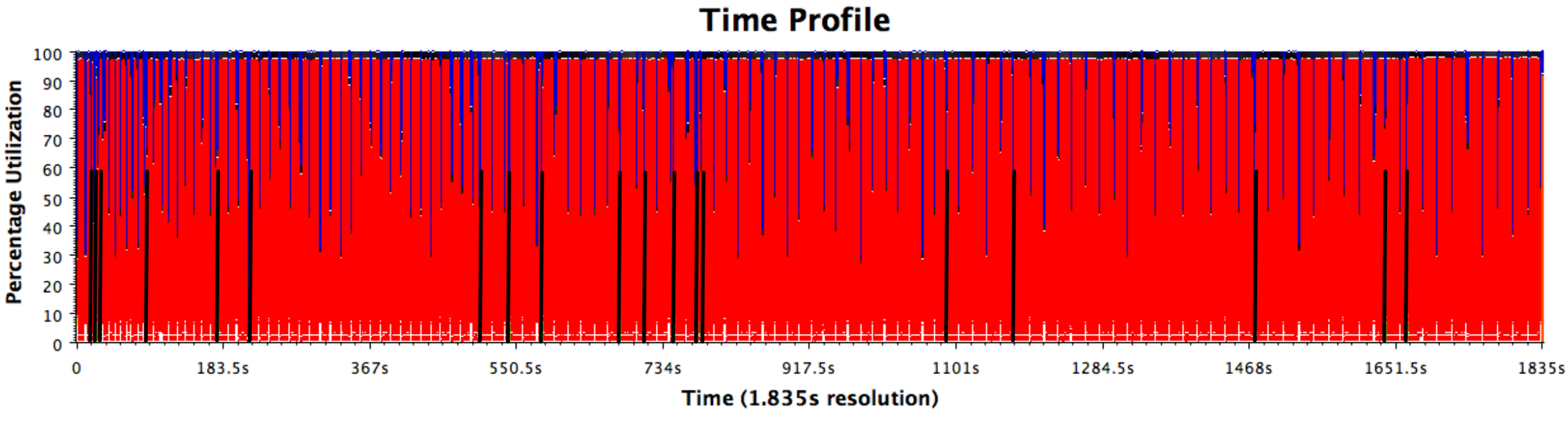


6s

- Failures are injected according to Weibull process: shape parameter 0.6
- Changing Checkpoint period

17s

Adapting to Failures



6s

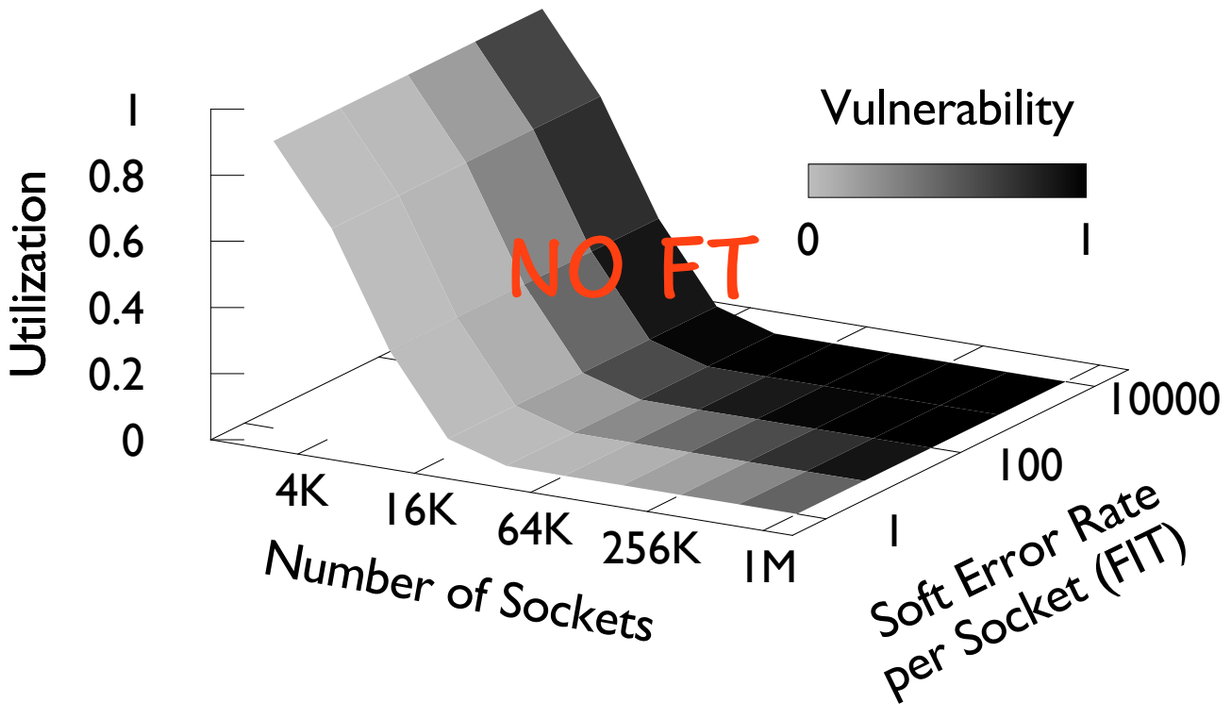
- Failures are injected according to Weibull process: shape parameter 0.6
- Changing Checkpoint period

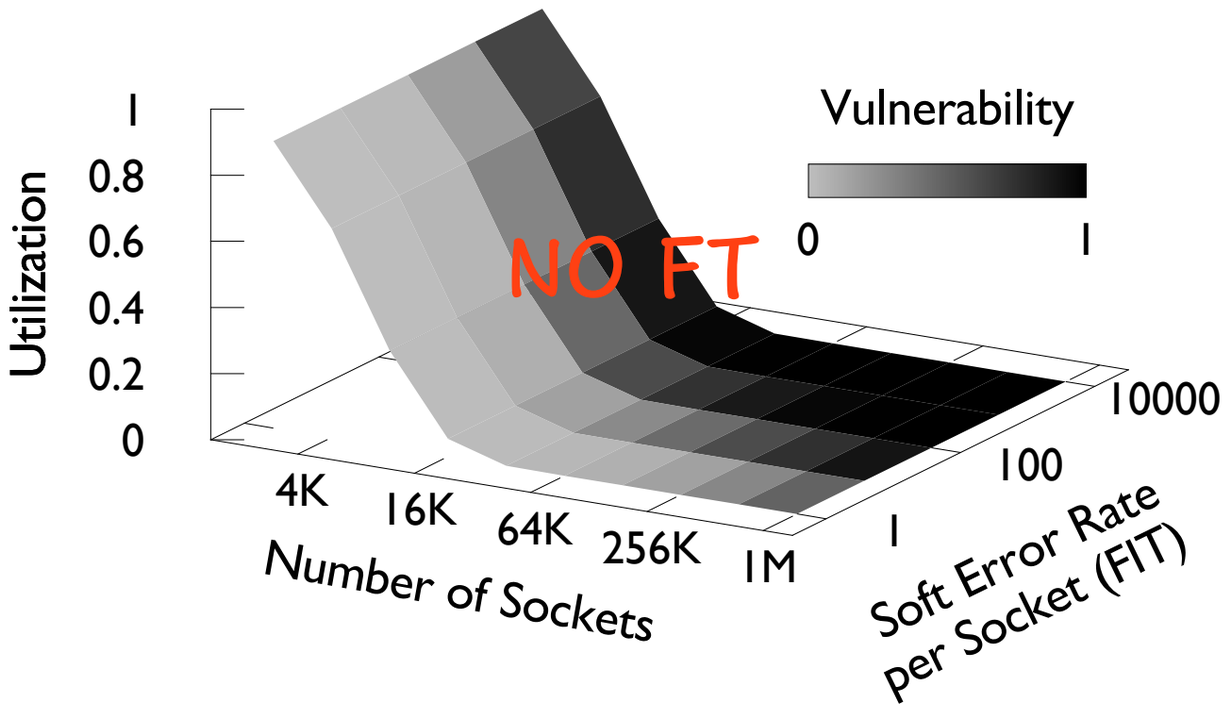
17s

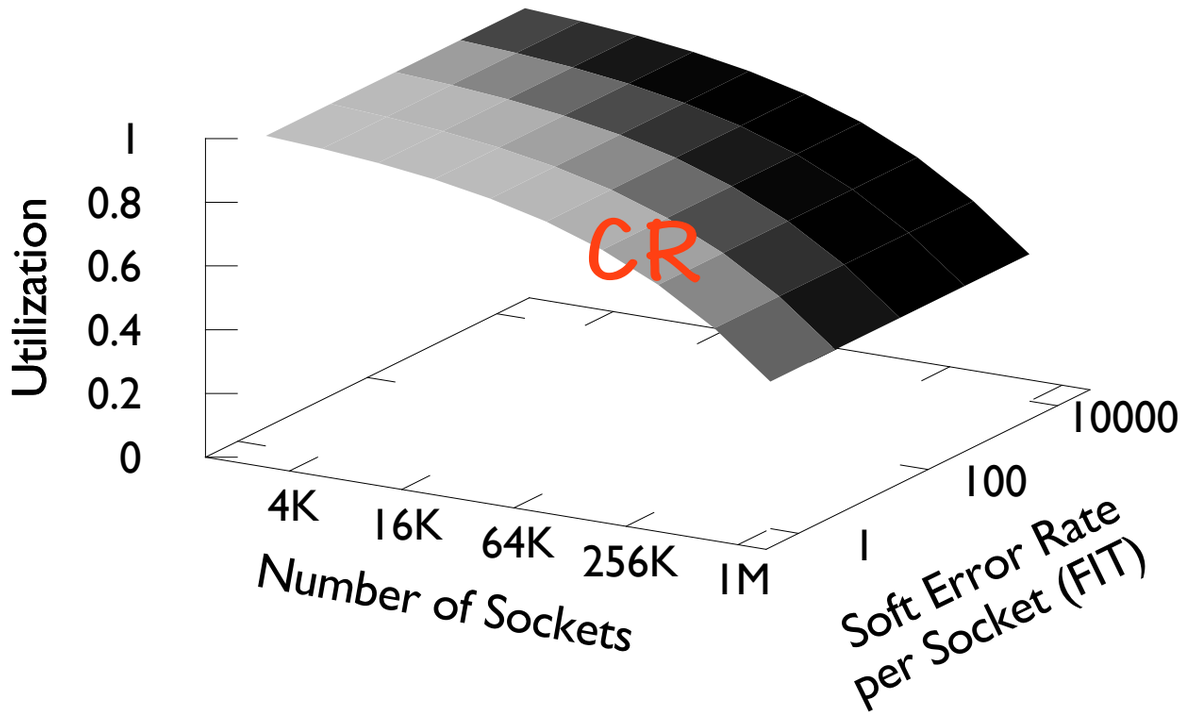
- Real failures injected: node becomes unresponsive
- Automatic restart: with the support of spare nodes and thus no need to submit the job again

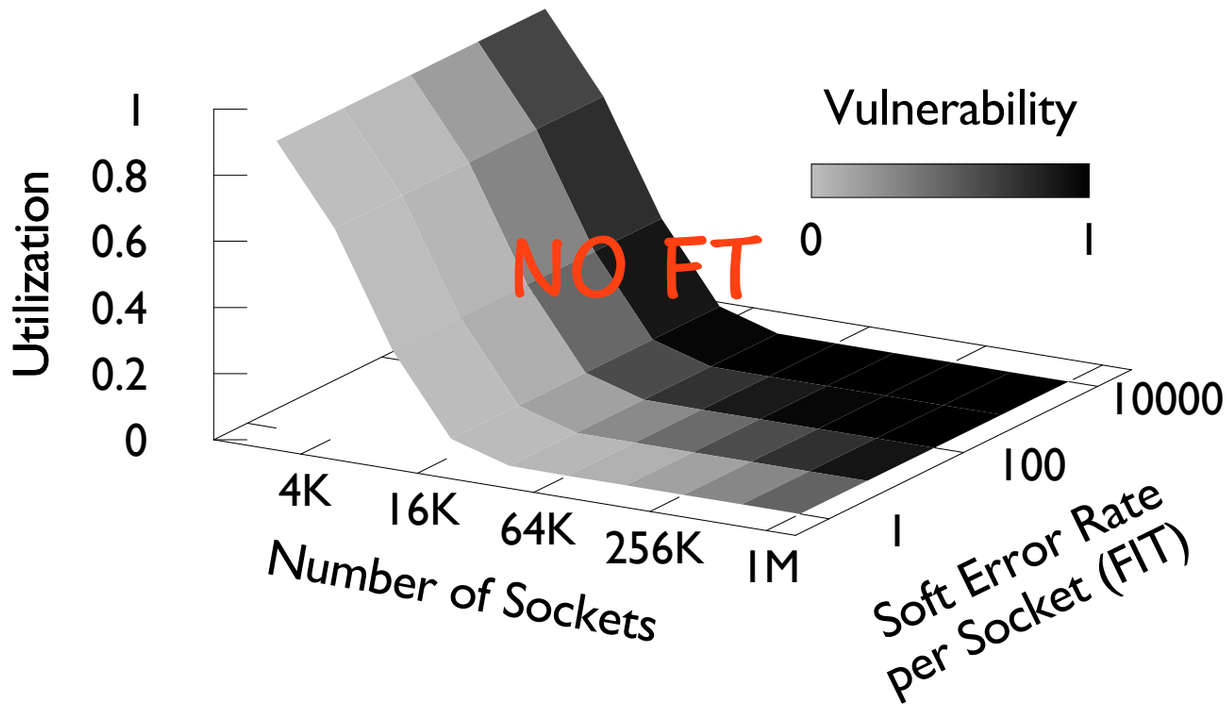
Part 2

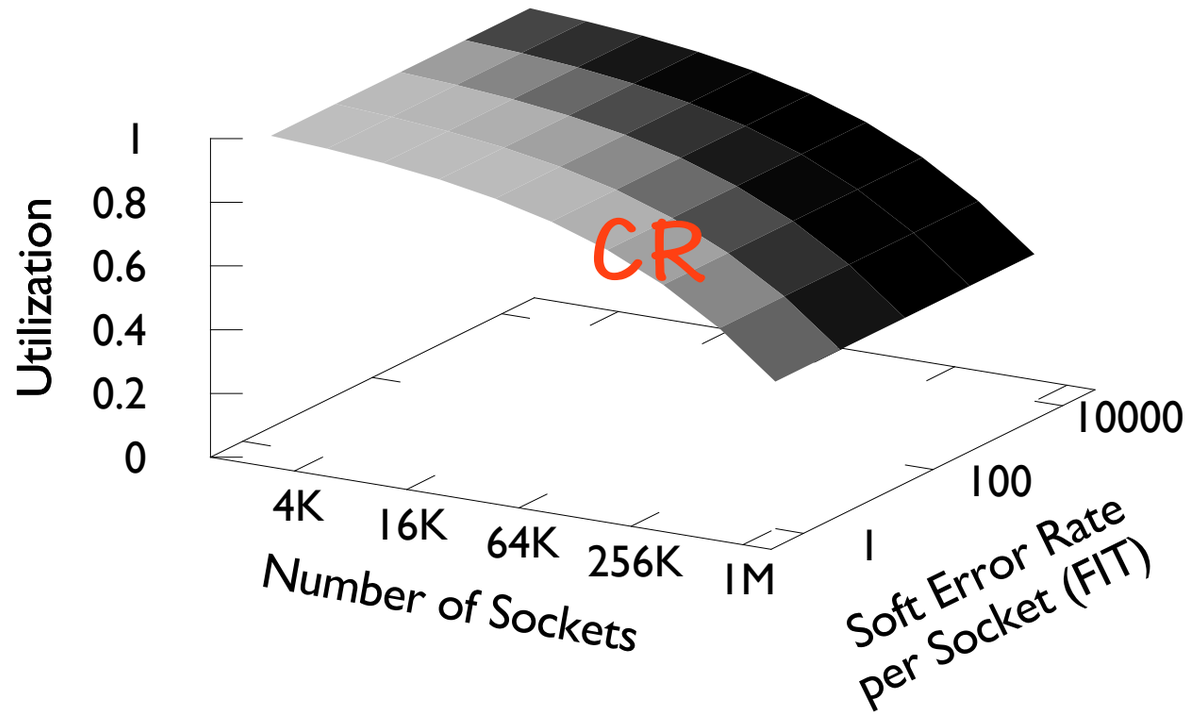
Detection and Correction of Silent Data Corrections

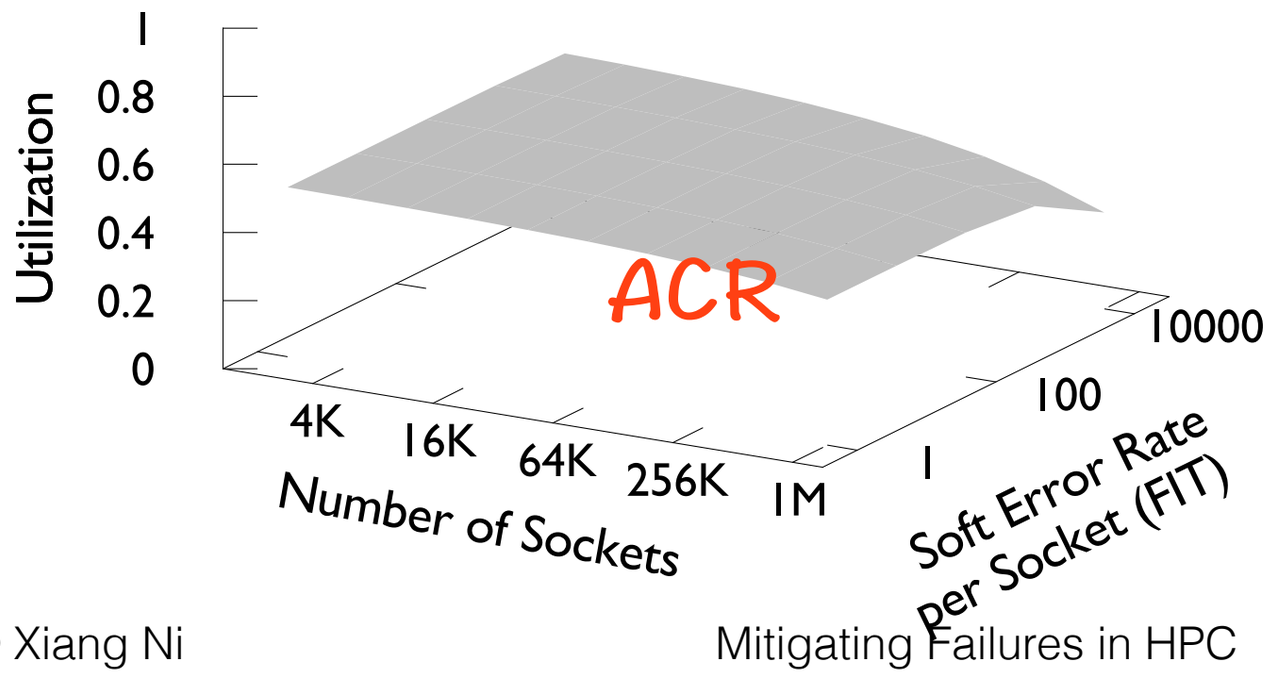

Utilization
1
0.8
0.6
0.4
0.2
0
Vulnerability
0
1
NO FT
4K
16K
64K
256K
1M
Number of Sockets
1
100
10000
Soft Error Rate
per Socket (FIT)


Utilization
1
0.8
0.6
0.4
0.2
0
Vulnerability
0
1
NO FT
4K
16K
64K
256K
1M
Number of Sockets
1
100
10000
Soft Error Rate
per Socket (FIT)

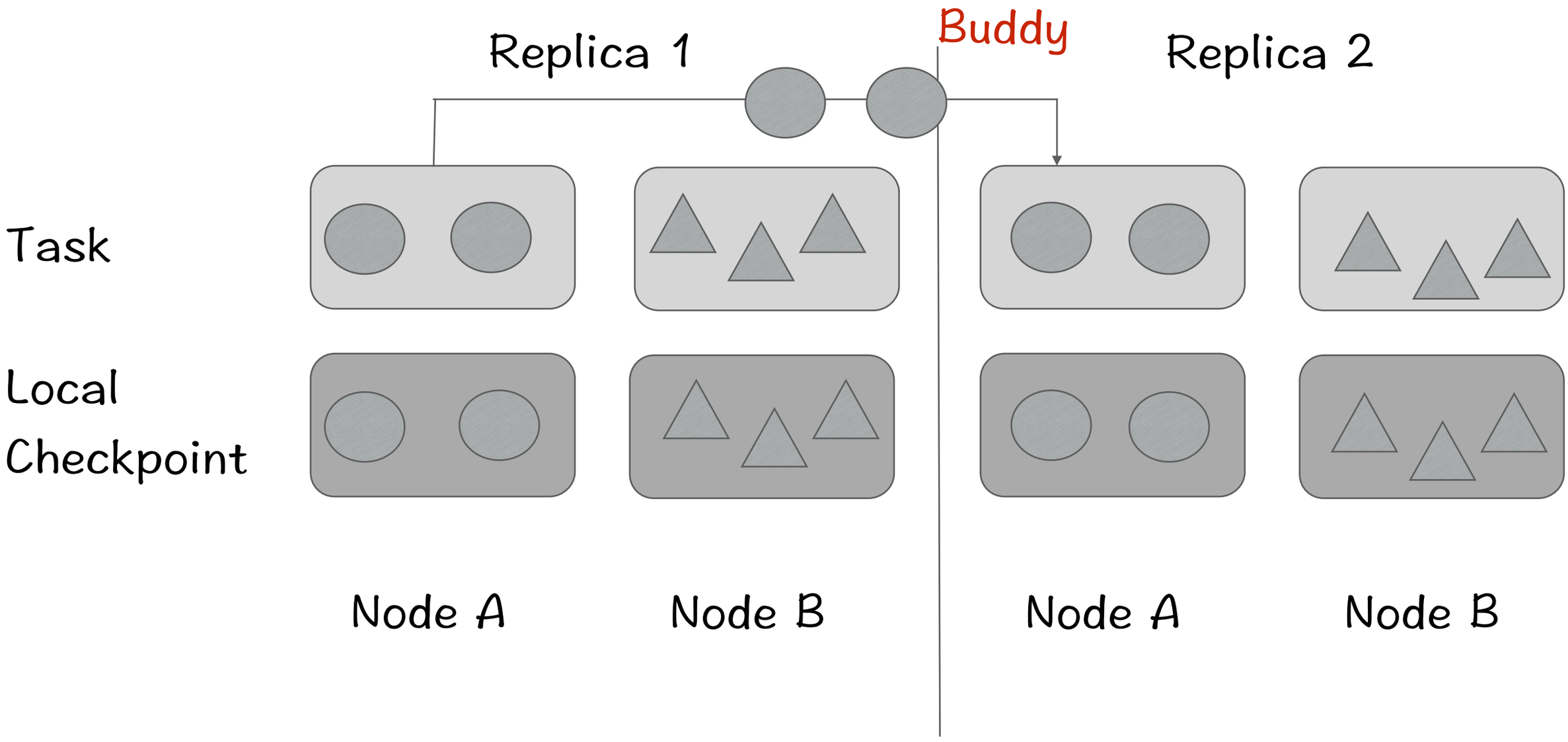

Utilization
1
0.8
0.6
0.4
0.2
0
CR
4K
16K
64K
256K
1M
Number of Sockets
1
100
10000
Soft Error Rate
per Socket (FIT)


Utilization
1
0.8
0.6
0.4
0.2
0
Vulnerability
0
1
NO FT
4K
16K
64K
256K
1M
Number of Sockets
1
100
10000
Soft Error Rate
per Socket (FIT)


Utilization
1
0.8
0.6
0.4
0.2
0
CR
4K
16K
64K
256K
1M
Number of Sockets
1
100
10000
Soft Error Rate
per Socket (FIT)


Utilization
1
0.8
0.6
0.4
0.2
0
ACR
4K
16K
64K
256K
1M
Number of Sockets
1
100
10000
Soft Error Rate
per Socket (FIT)

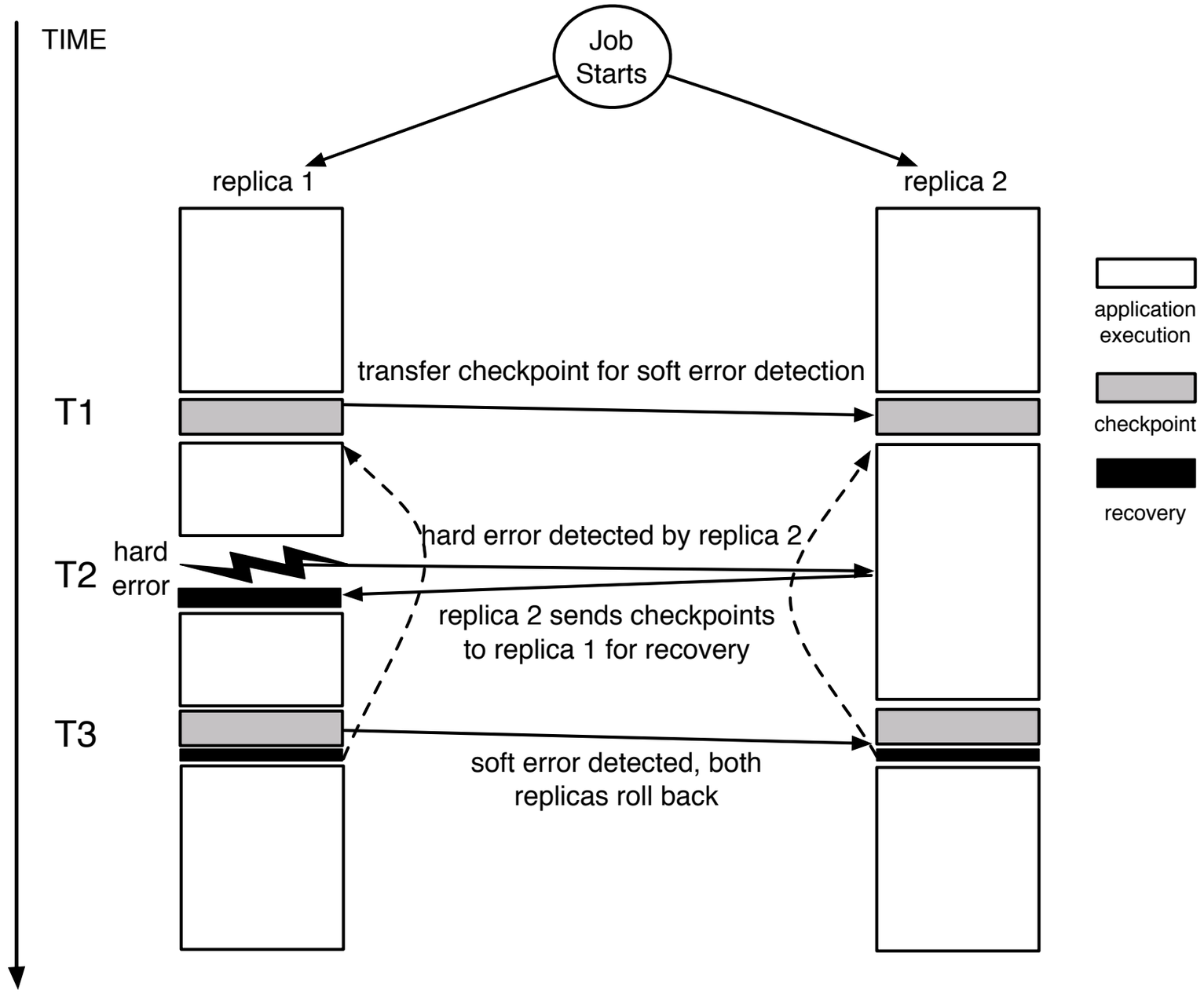
Replication Enhanced Checkpointing



Comparison for silent data corruption

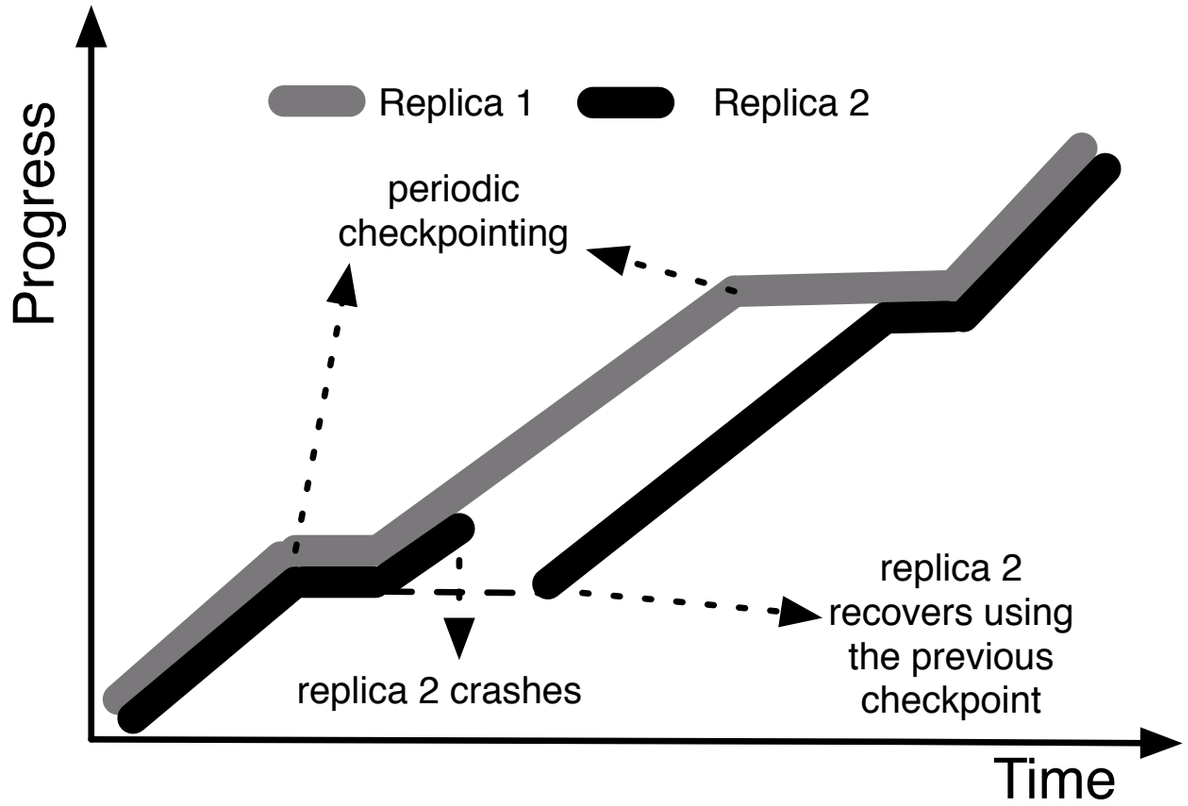
SC13: ACR: Automatic Checkpoint/Restart for Soft and Hard Error protection

Replication Enhanced Checkpointing

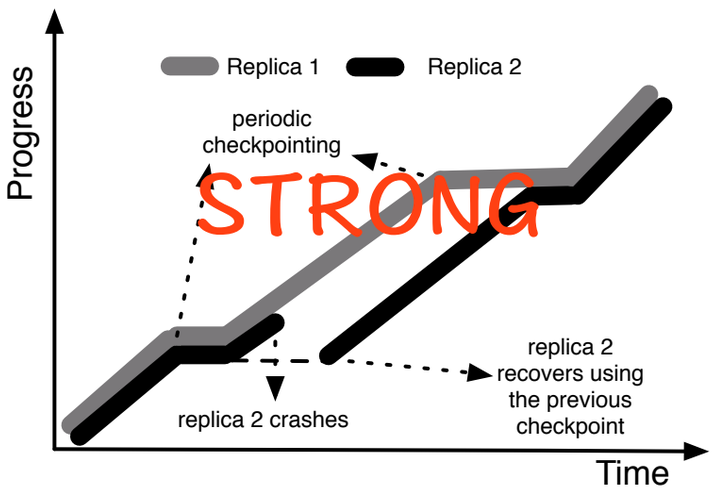


Different Ways to Restart from Hard Errors

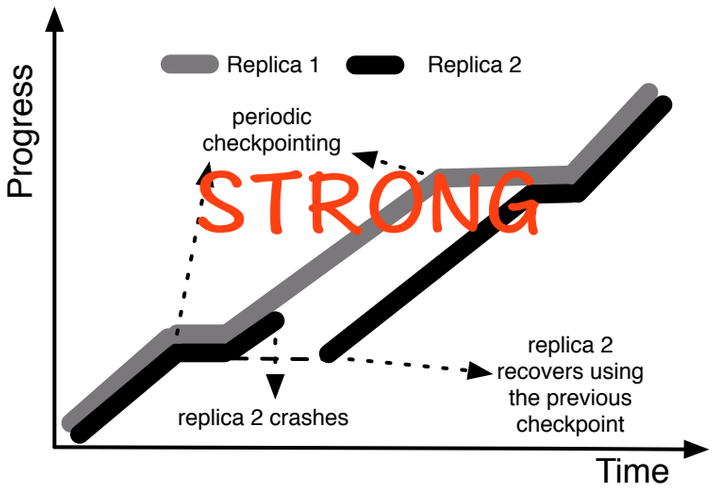
Different Ways to Restart from Hard Errors

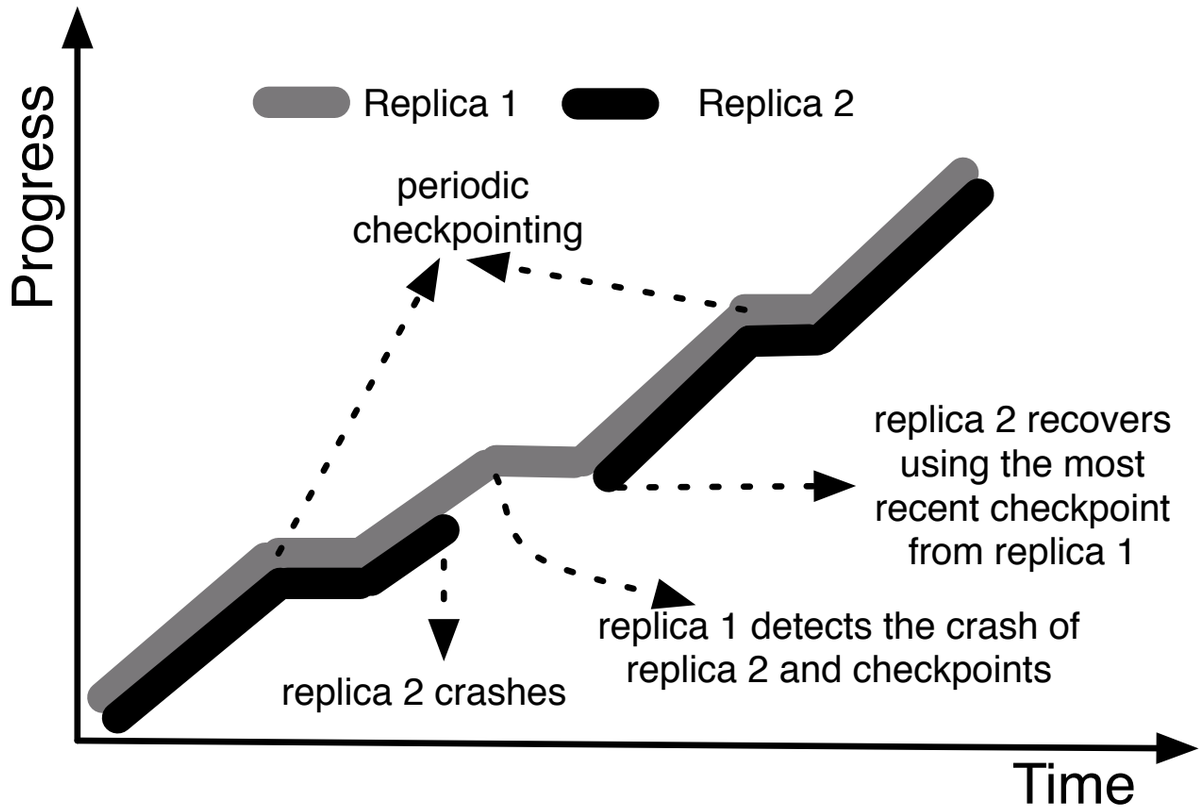


Different Ways to Restart from Hard Errors

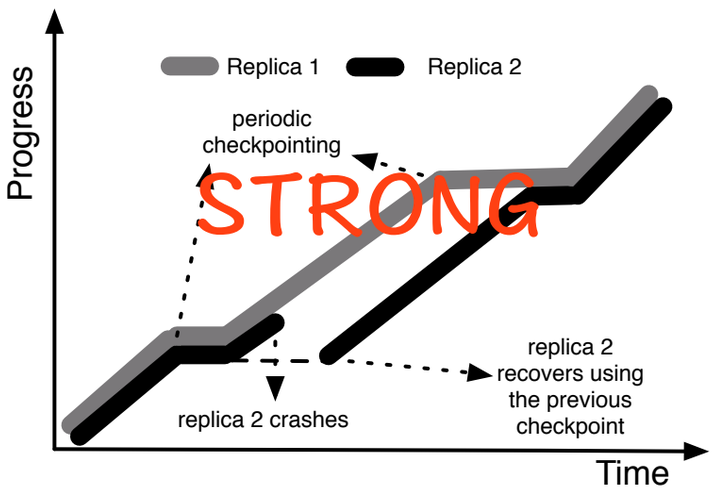


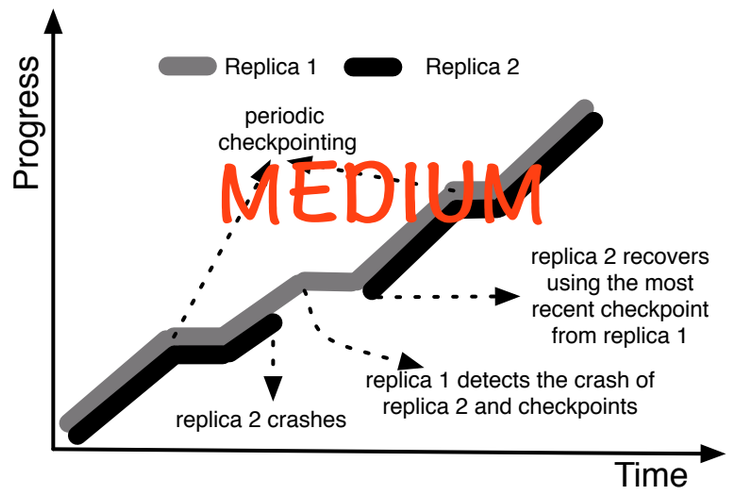
Different Ways to Restart from Hard Errors



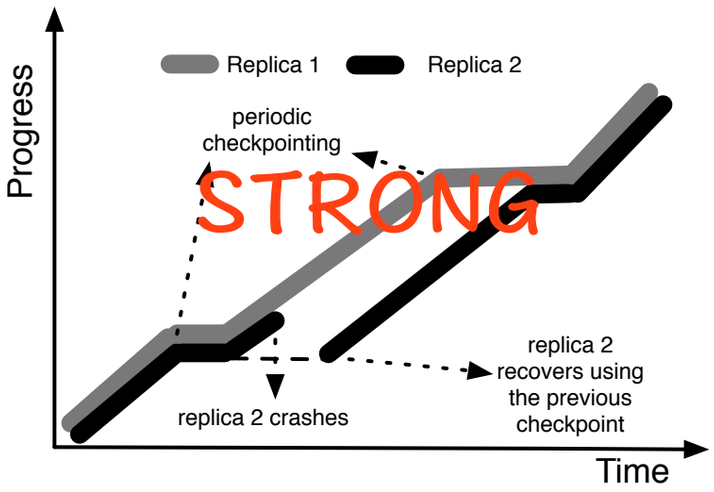


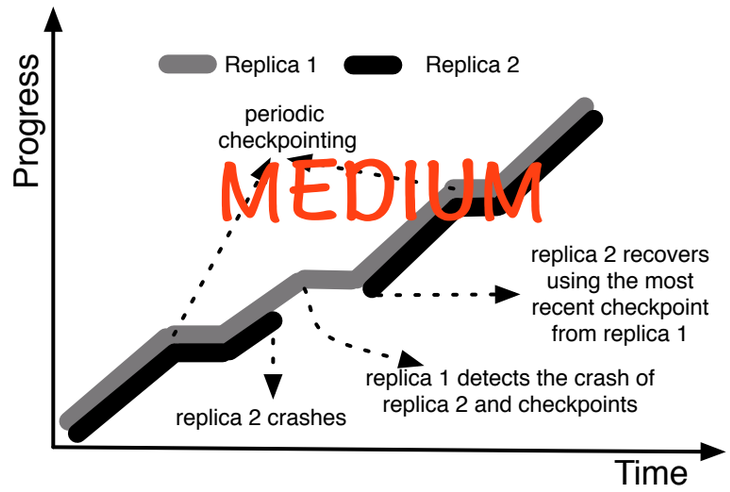
Different Ways to Restart from Hard Errors

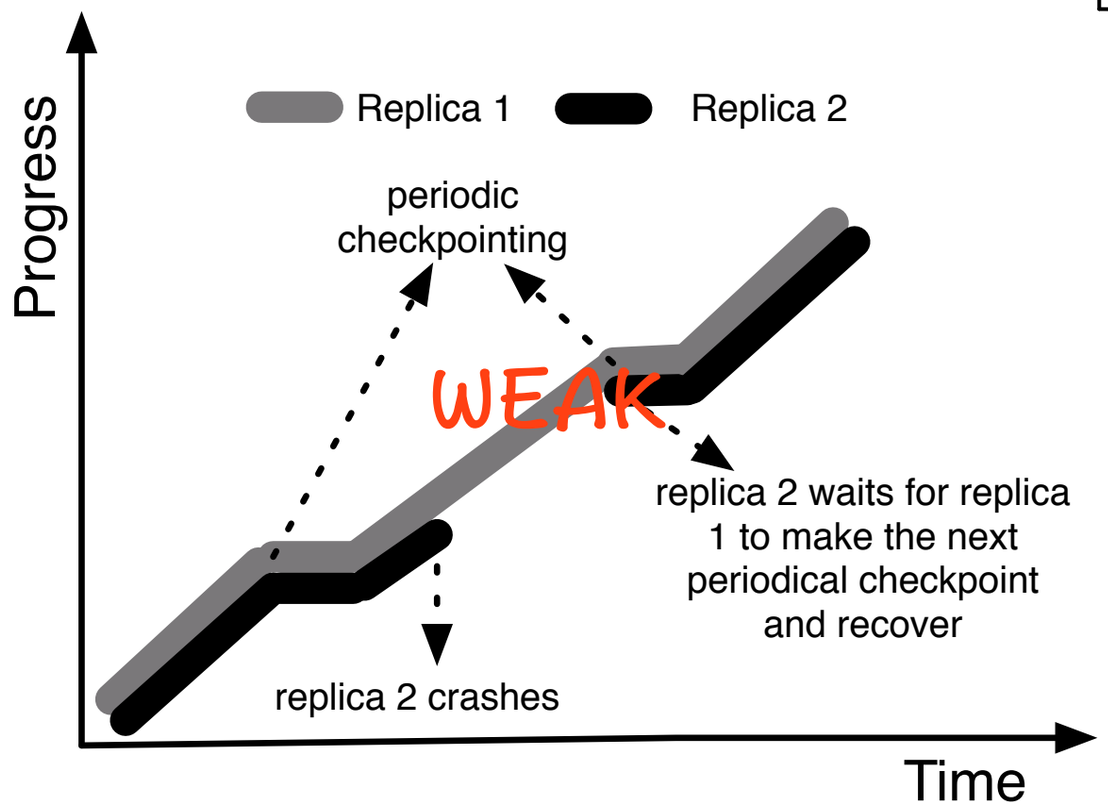




Different Ways to Restart from Hard Errors



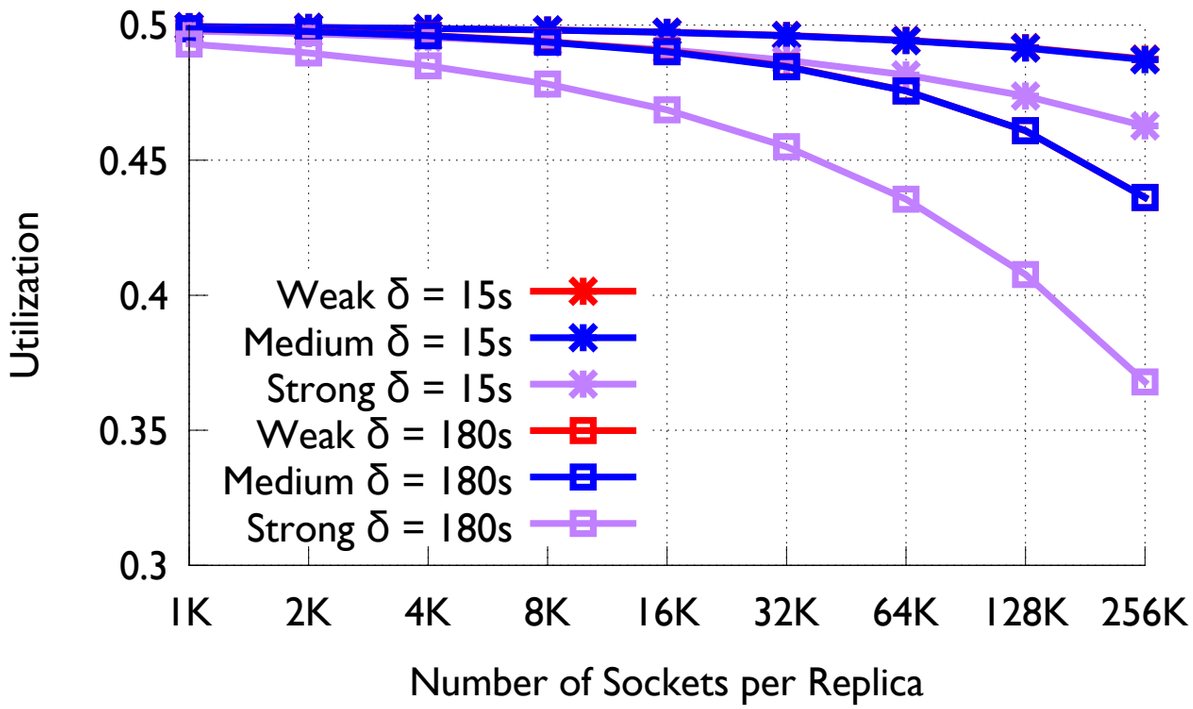


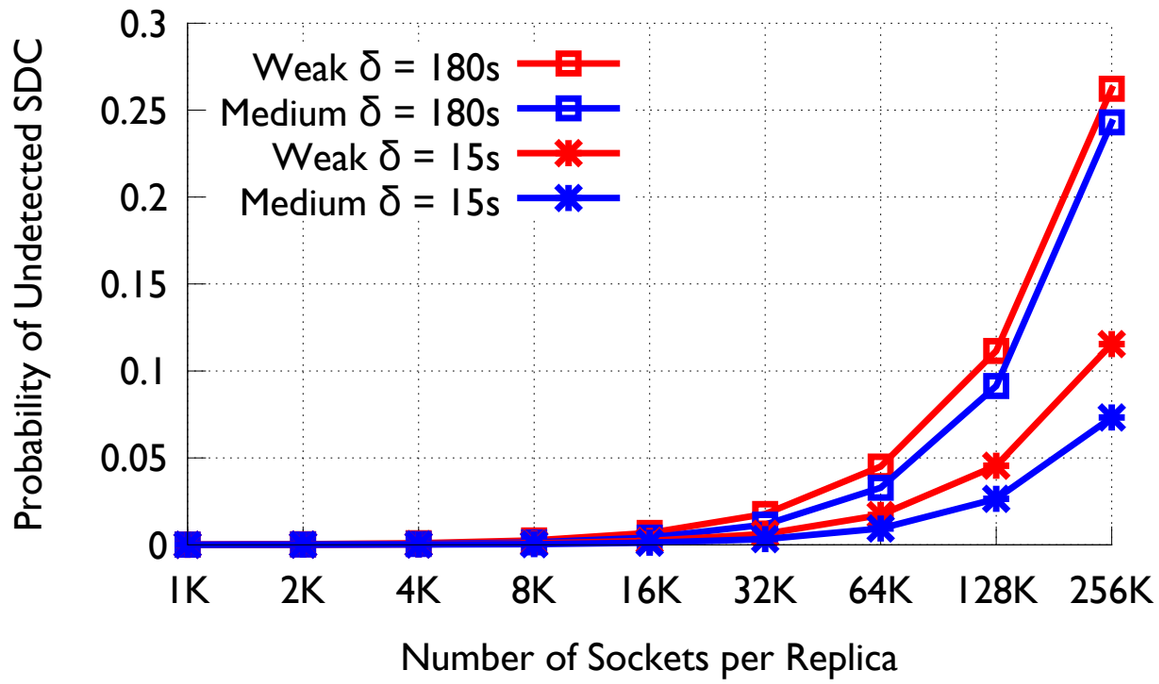


Utilization vs. Vulnerability

Mean time between hard errors per socket: 50 years

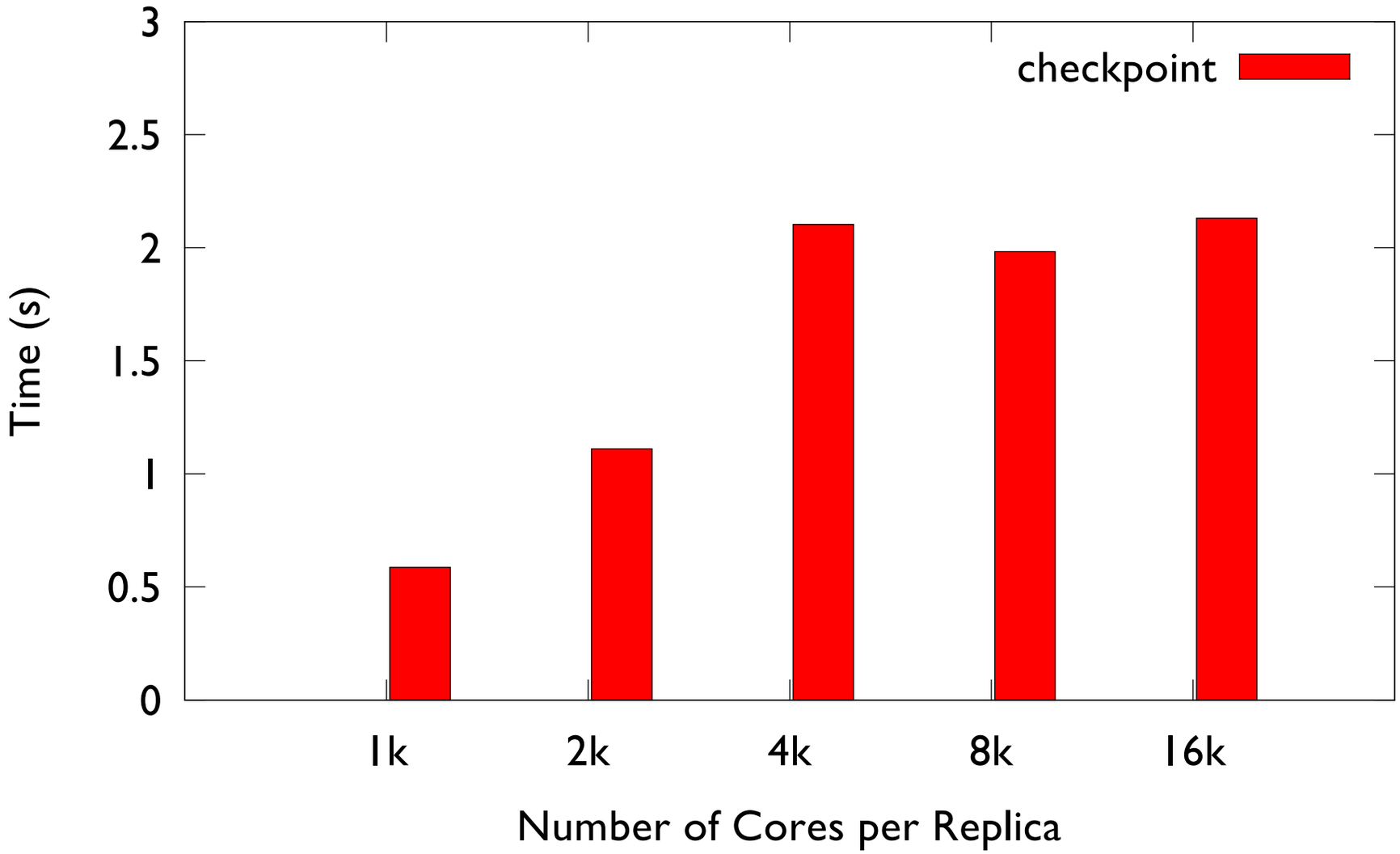
SDC FIT: 100





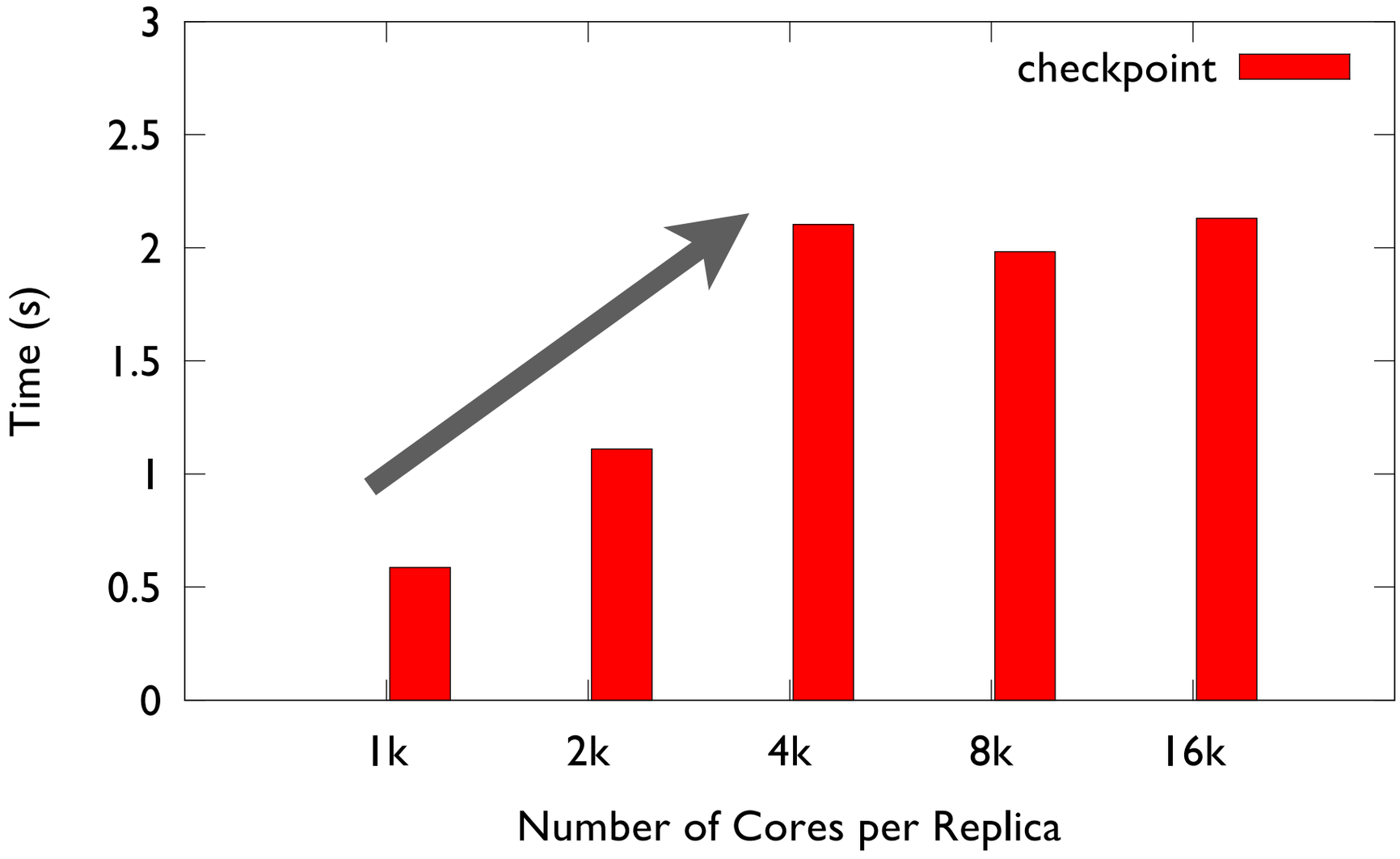
- Weak&medium resilience schemes achieve much higher utilization ratio
- Even on 64K sockets, the probability of undetected SDC is less than 0.1

Base performance



Jacobi3d BGP

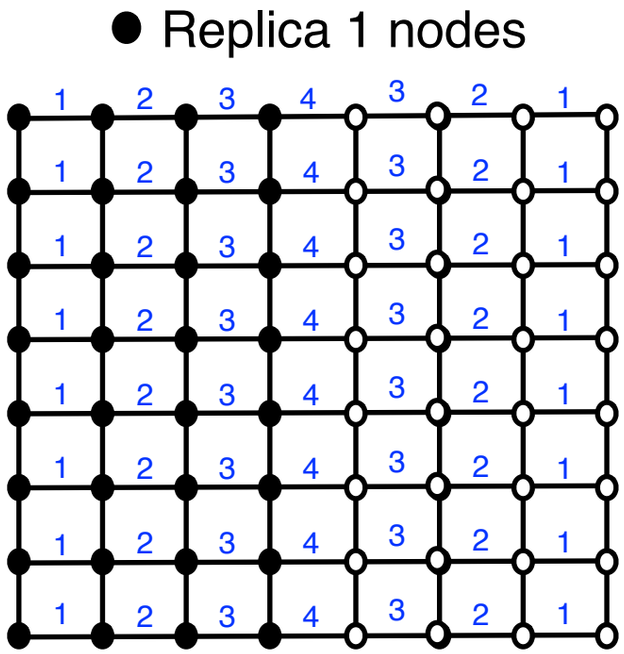
Base performance



Jacobi3d BGP

Optimization: Topology Aware Mapping

Optimization: Topology Aware Mapping



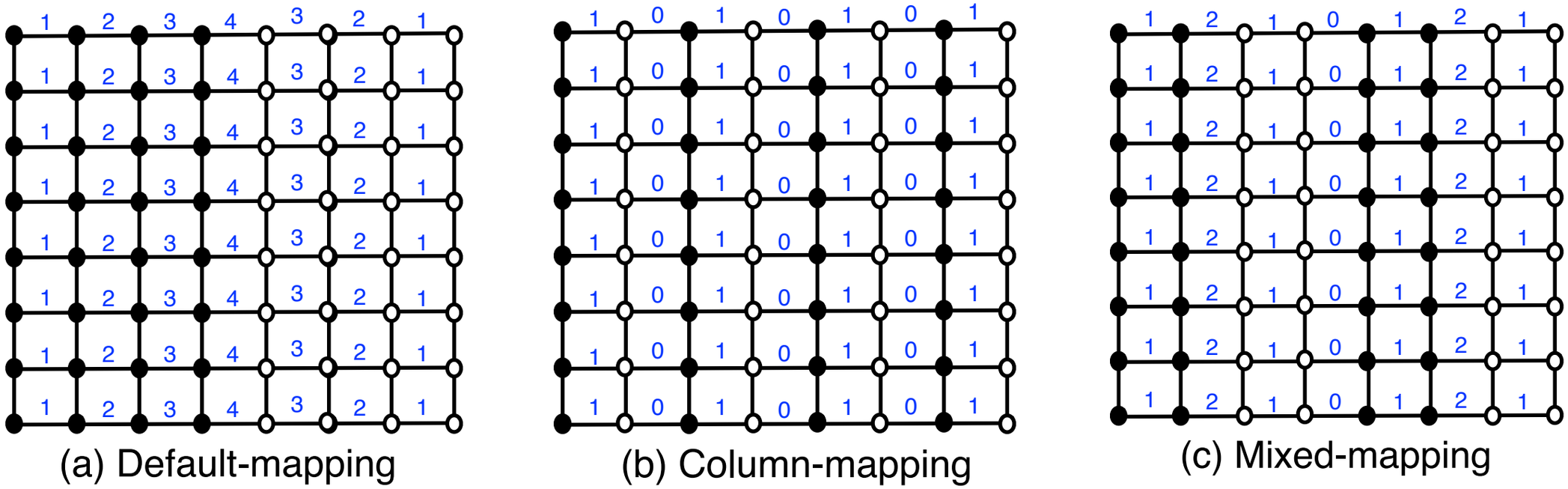
(a) Default-mapping

Optimization: Topology Aware Mapping

● Replica 1 nodes

o Replica 2 nodes

[0-4] # inter-replica messages



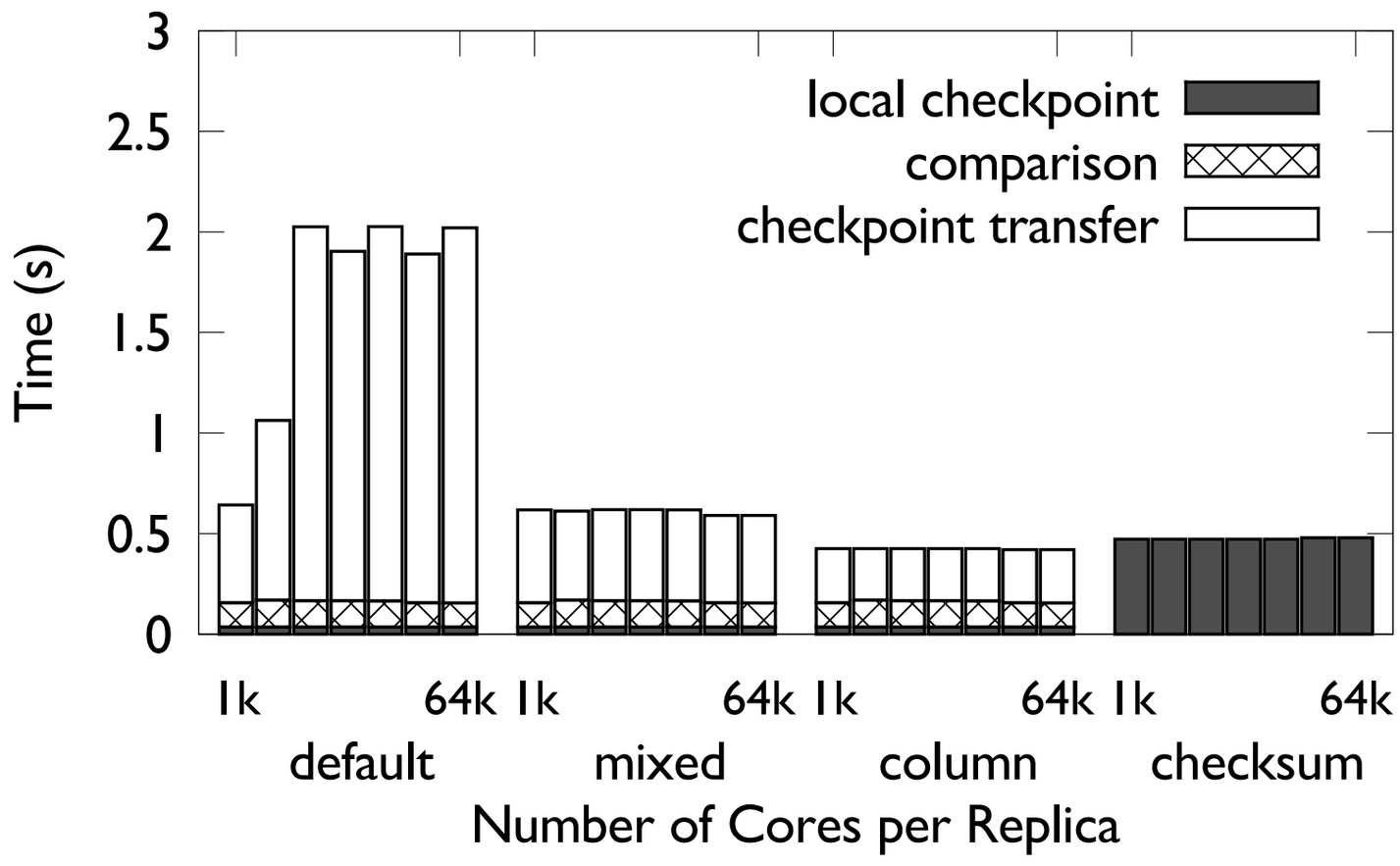
(a) Default-mapping (b) Column-mapping (c) Mixed-mapping

1) Reduce the inter-replica communication distance.

2) Trade-off between inter and intra replica communication.

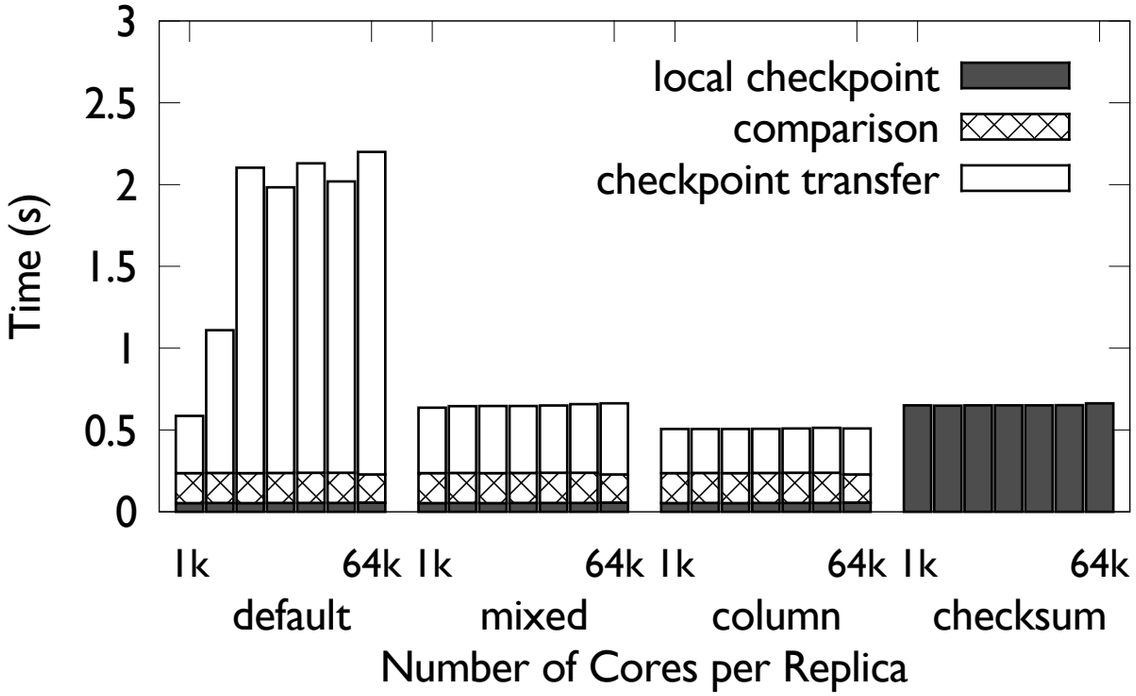
Optimization: Checksum

- Transfer the checksum of 1 integer instead of the whole checkpoints
- Floating point round-off error

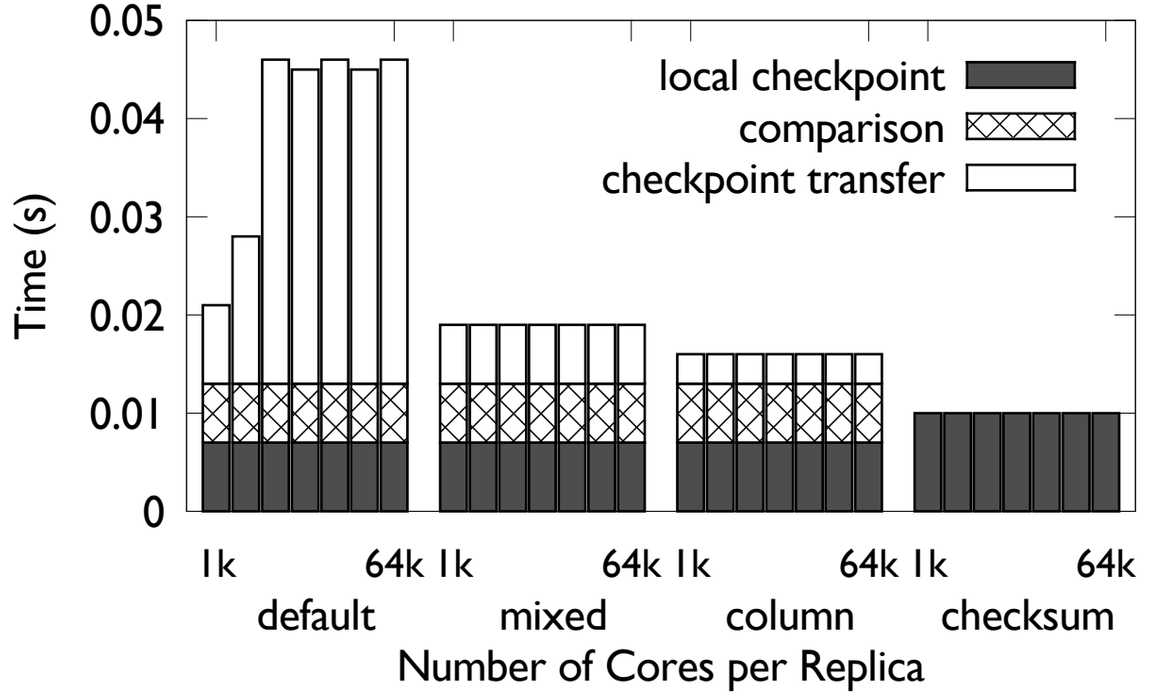


Experimental Results: Checkpoint

Benchmark	Description	Configuration per core	Memory Pressure	Runtime
Jacobi3D	7-point stencil	64*64*128	High	AMPI
LeanMD	Short-range non-bonded force	4000 atoms	Low	Charm++

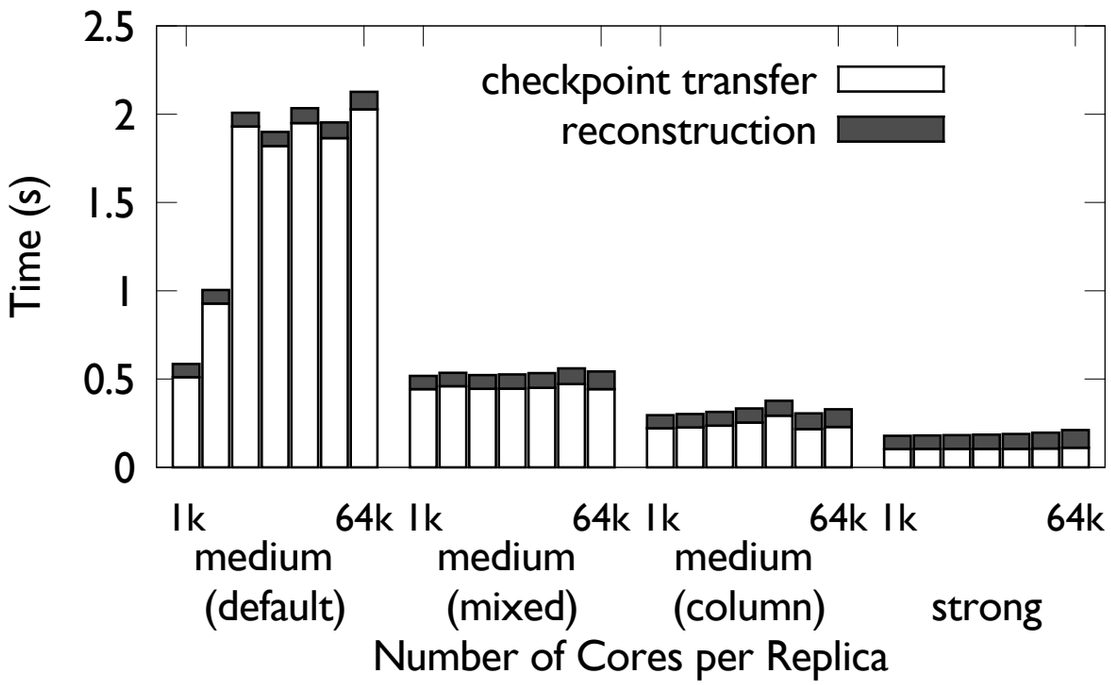


Jacobi3D AMPI

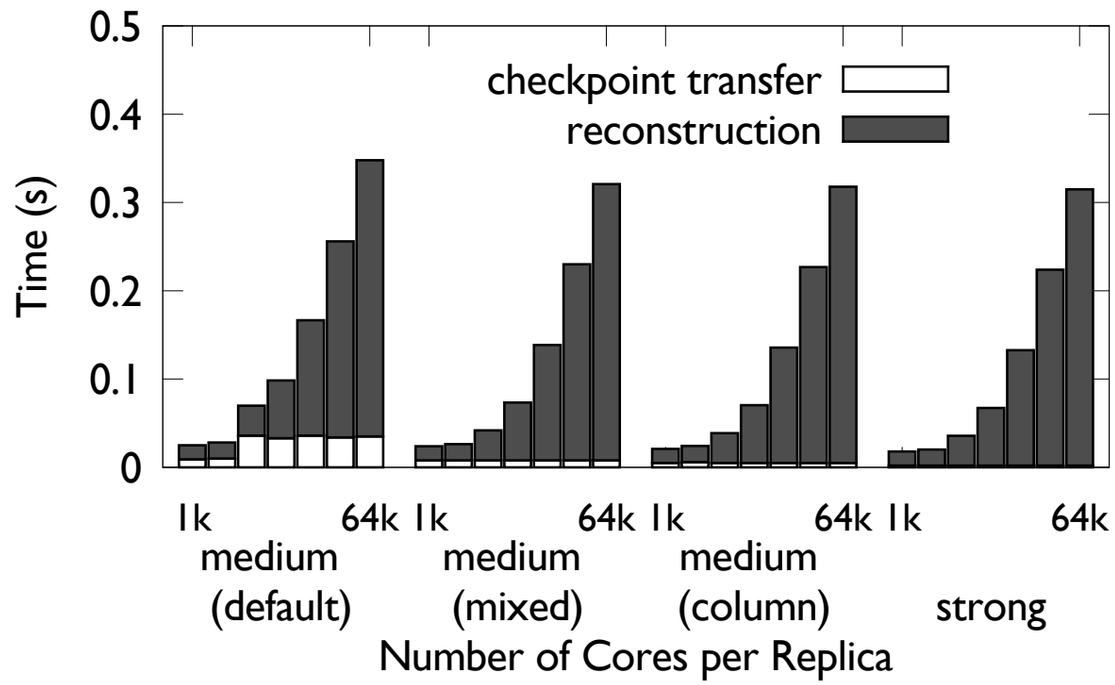


LeanMD

Experimental Results: Restart

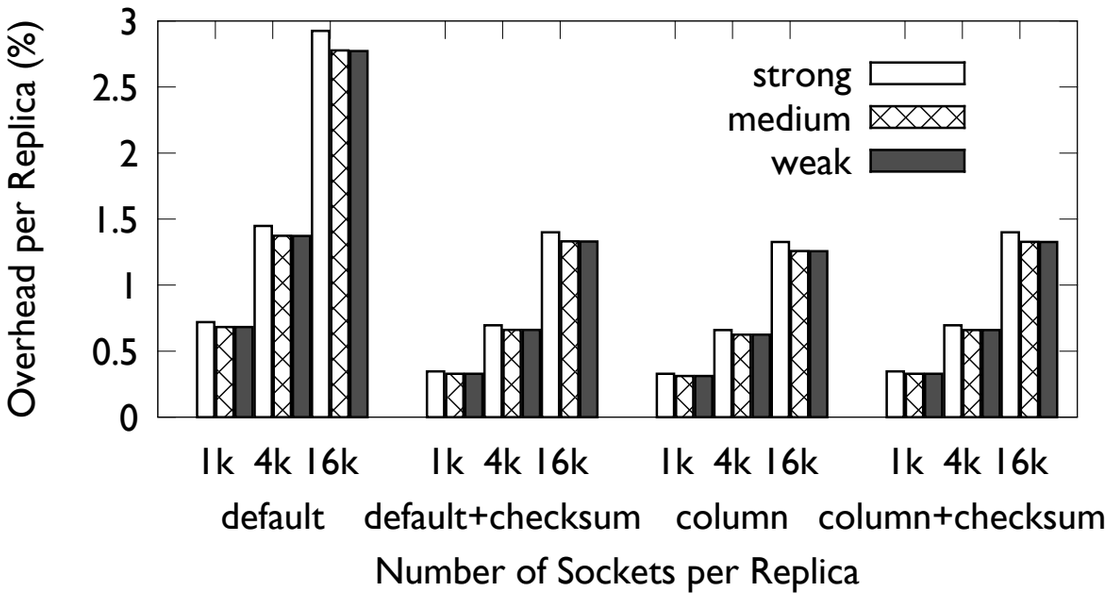


Jacobi3D AMPI

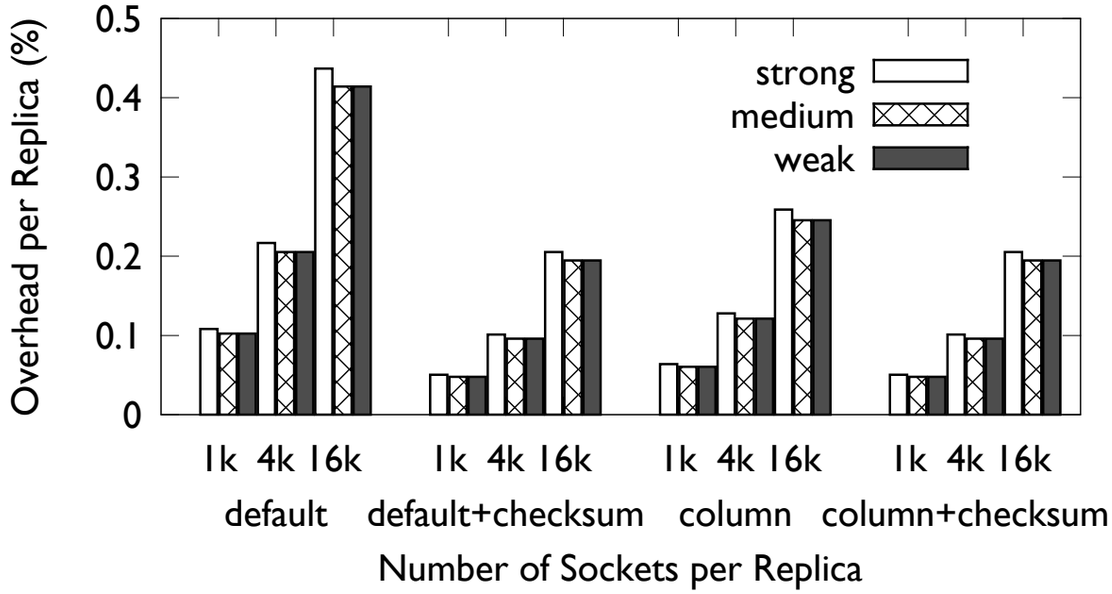


LeanMD

Checkpoint Overhead



Jacobi3D AMPI

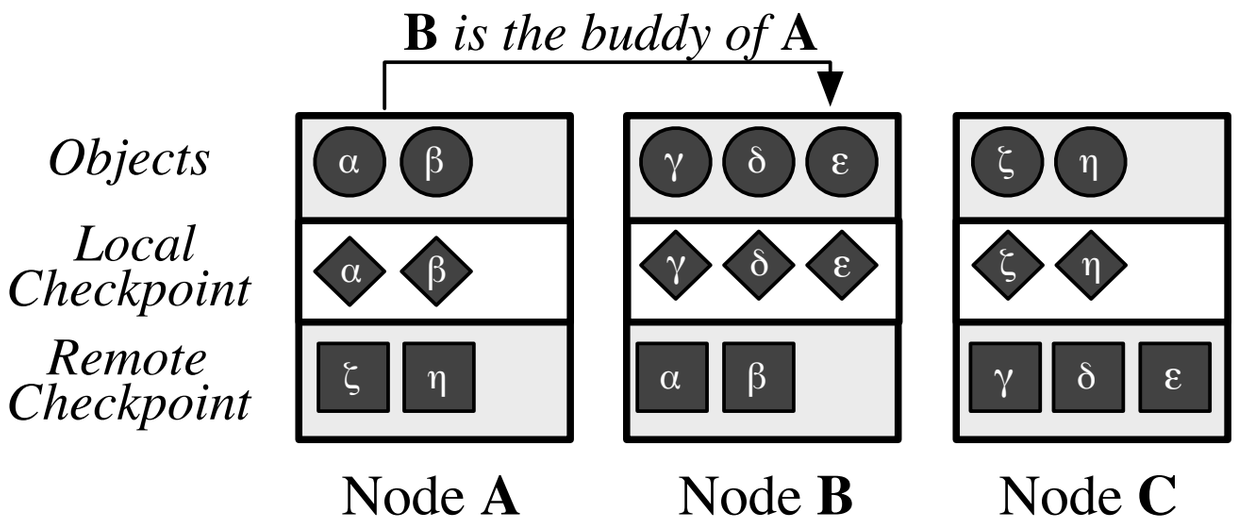


LeanMD

Part 3

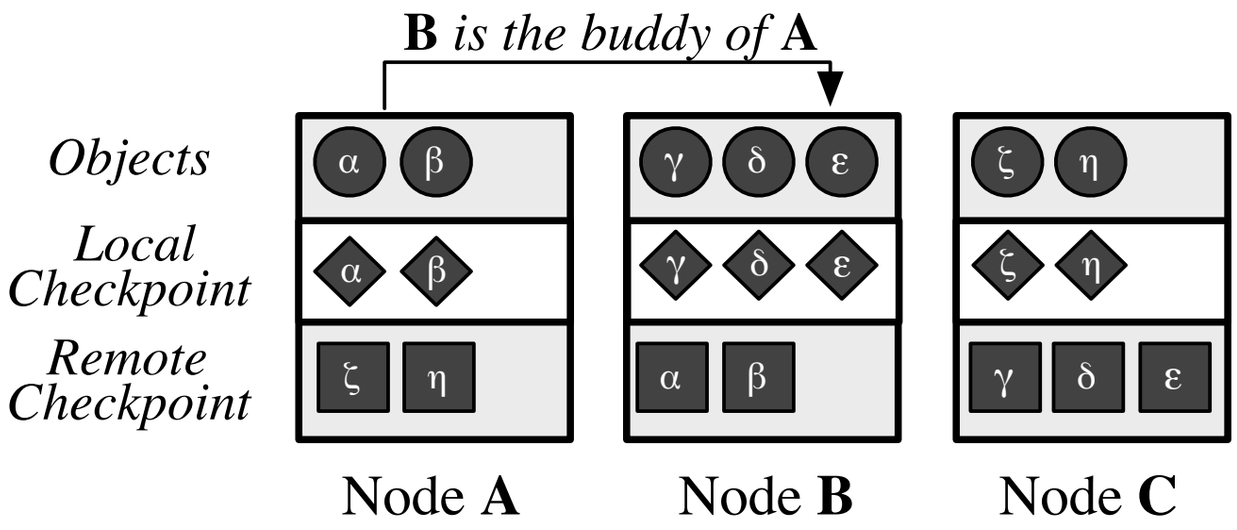
Memory Limitations

Checkpointing to SSD



- Hard to fit application data and checkpoints in memory at the same time
- Full SSD strategy: store both local and remote checkpoints in SSD
- Half SSD strategy: store only remote checkpoint in SSD

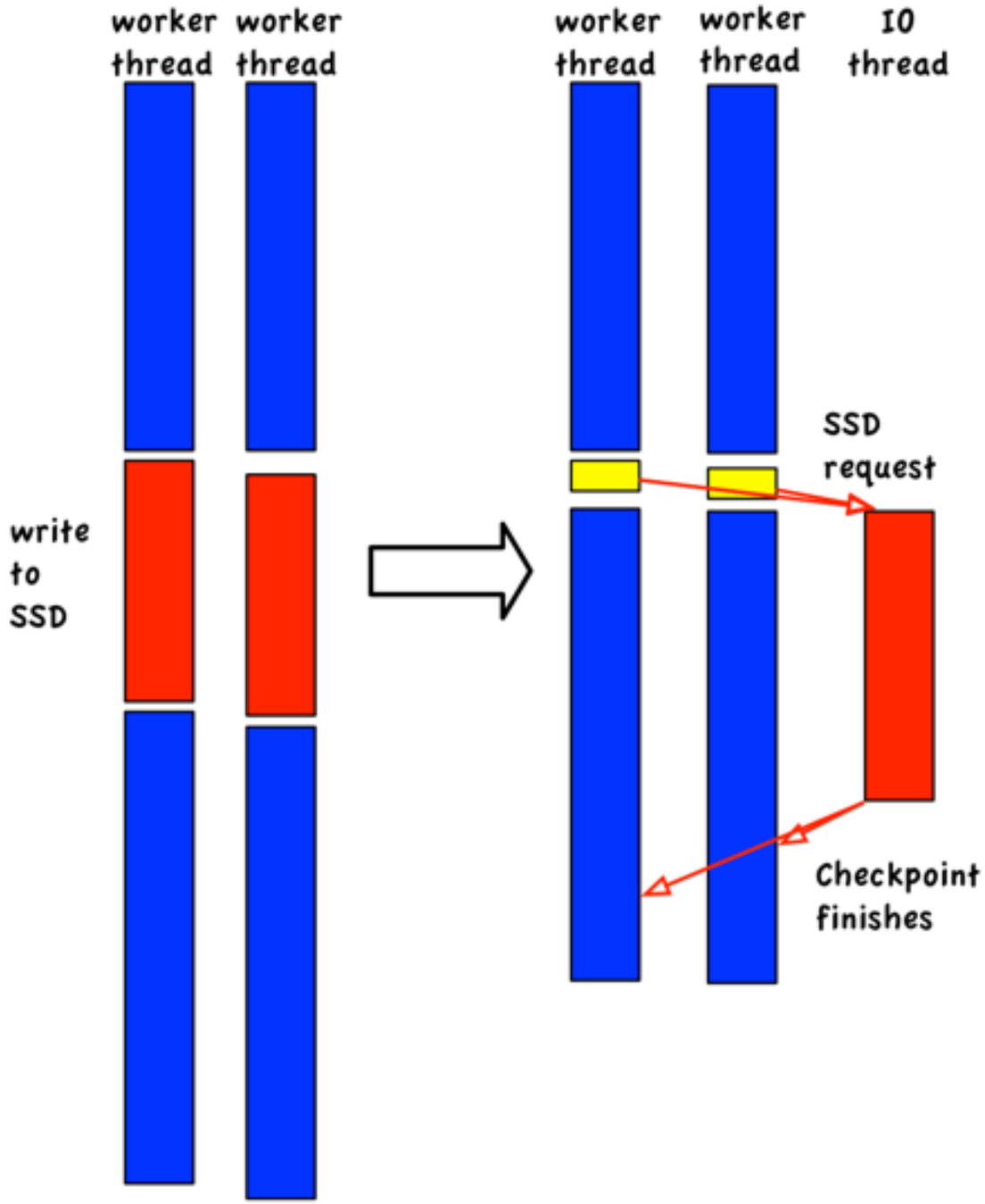
Checkpointing to SSD



- Hard to fit application data and checkpoints in memory at the same time
- Full SSD strategy: store both local and remote checkpoints in SSD
- Half SSD strategy: store only remote checkpoint in SSD

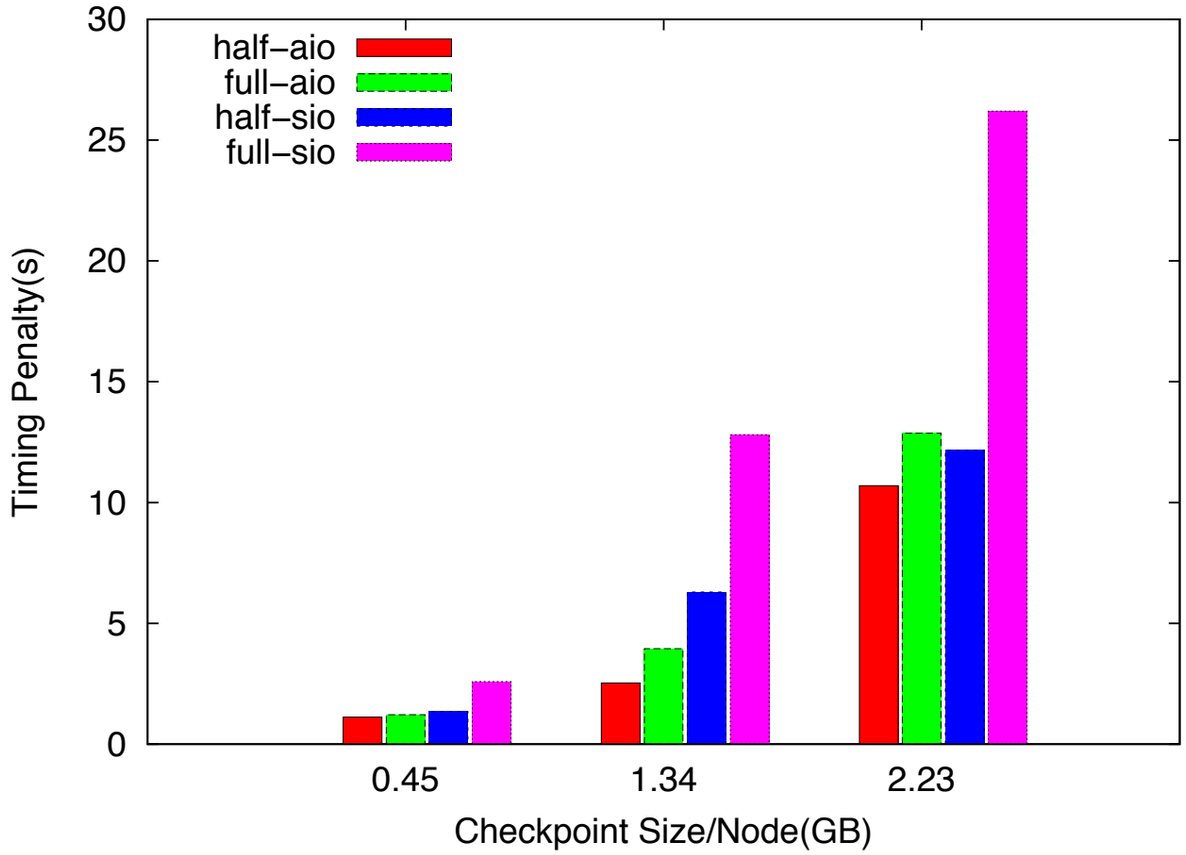
faster checkpoint&restart

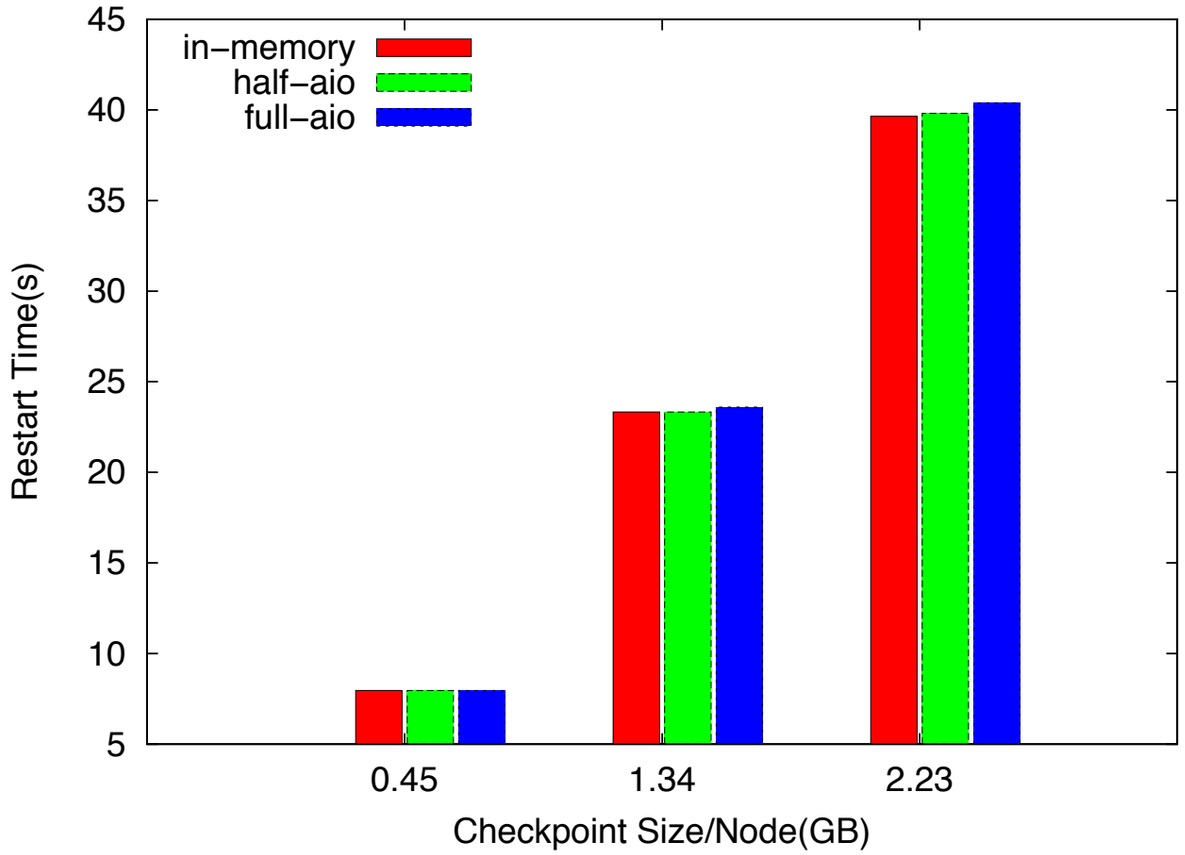
Asynchronous Checkpointing



- IO threads
  - Write checkpoint to/Read checkpoint from SSD when receiving requests from worker threads.
  - Notify worker threads when SSD is done with certain requests.

Checkpoint/Restart on SSD





- Half SSD strategy with asynchronous IO much reduces the checkpoint overhead.
- Restarting from SSD does not incur extra overhead.

aio	asynchronous IO
sio	synchronous IO