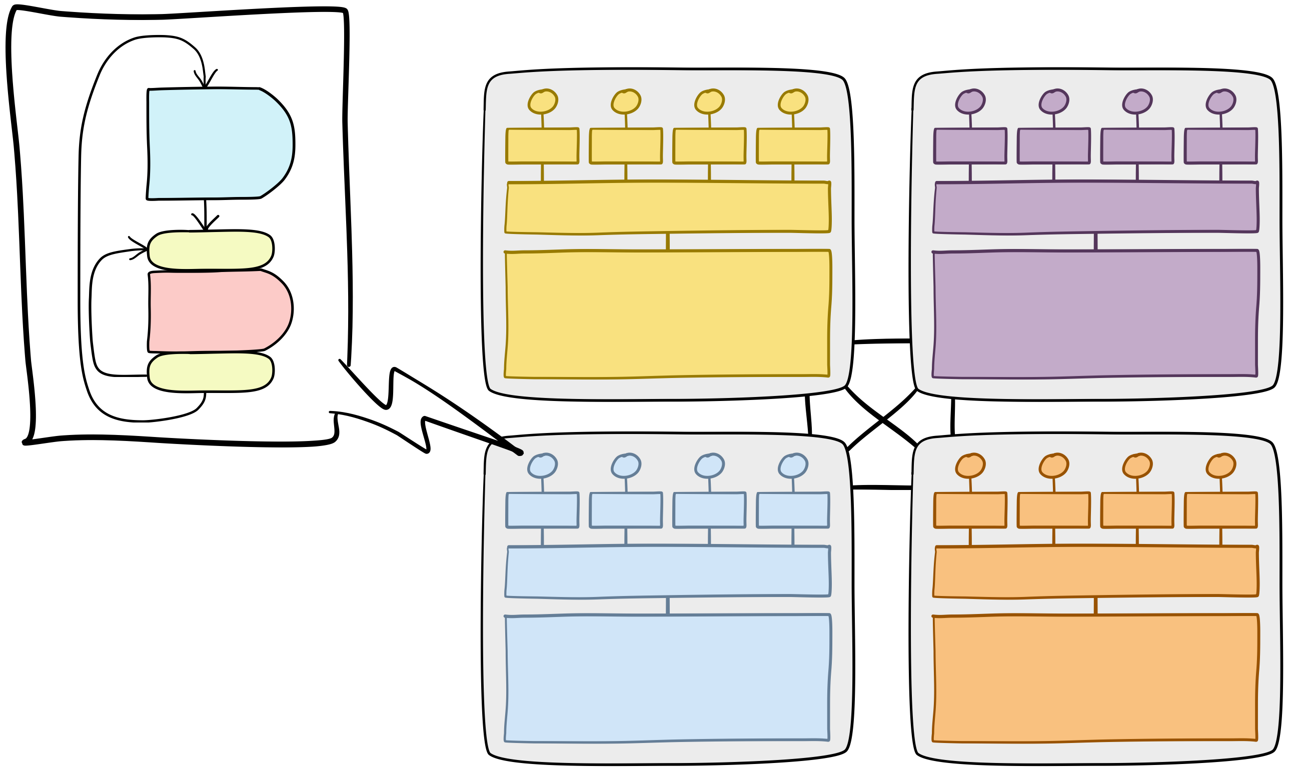
Unfair Scheduling Patterns *in* NUMA Architectures

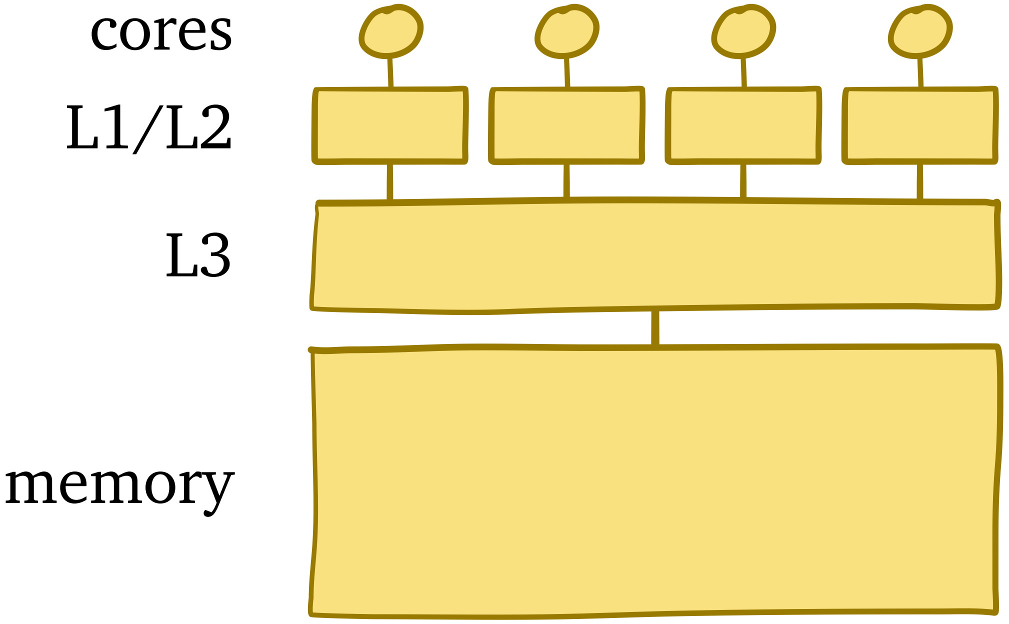
Naama Ben-David
Ziv Scully
Guy Blelloch

Carnegie Mellon University



NUMA Architectures
+
Concurrent Programs

NUMA Architectures



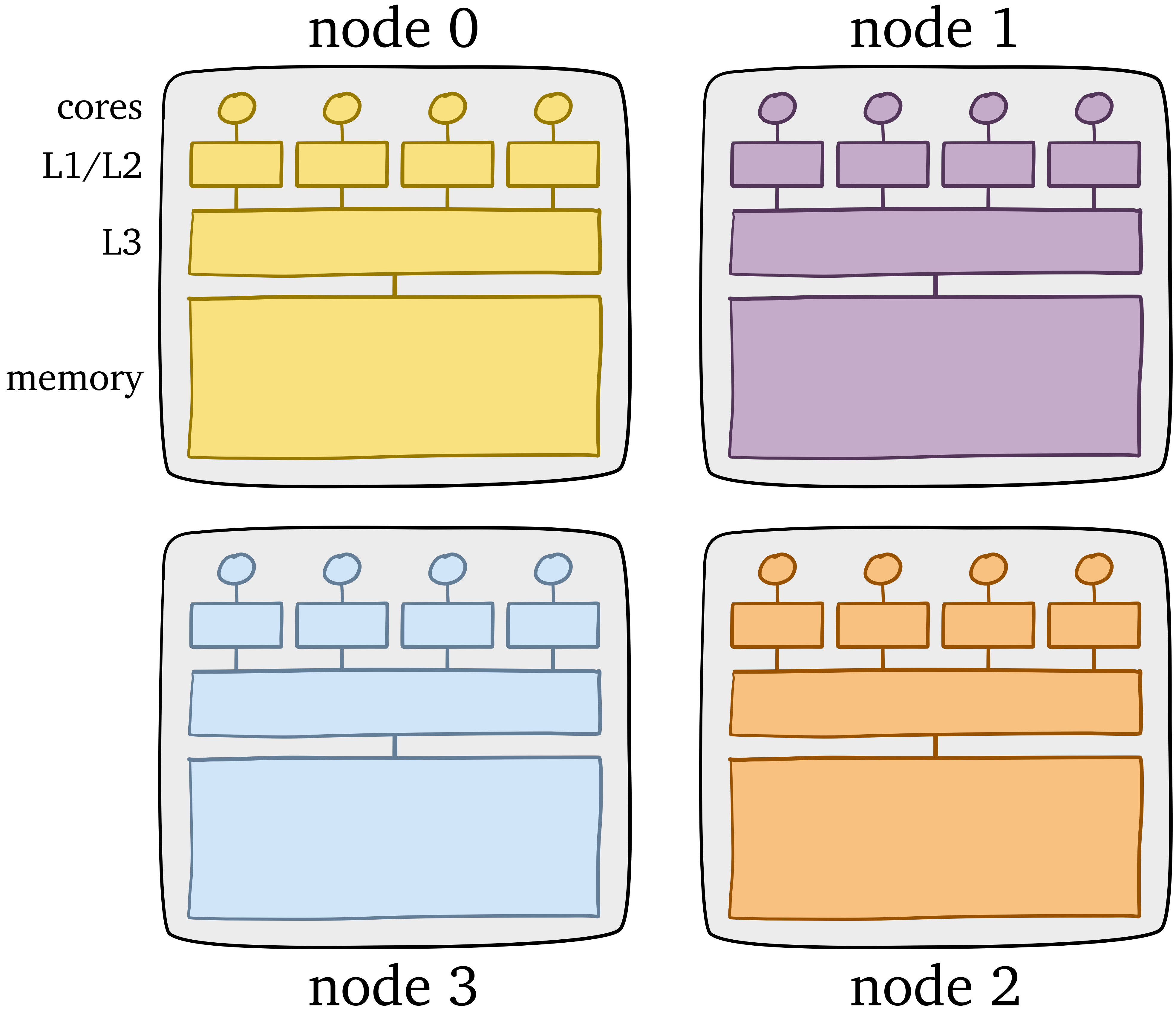
cores

L1/L2

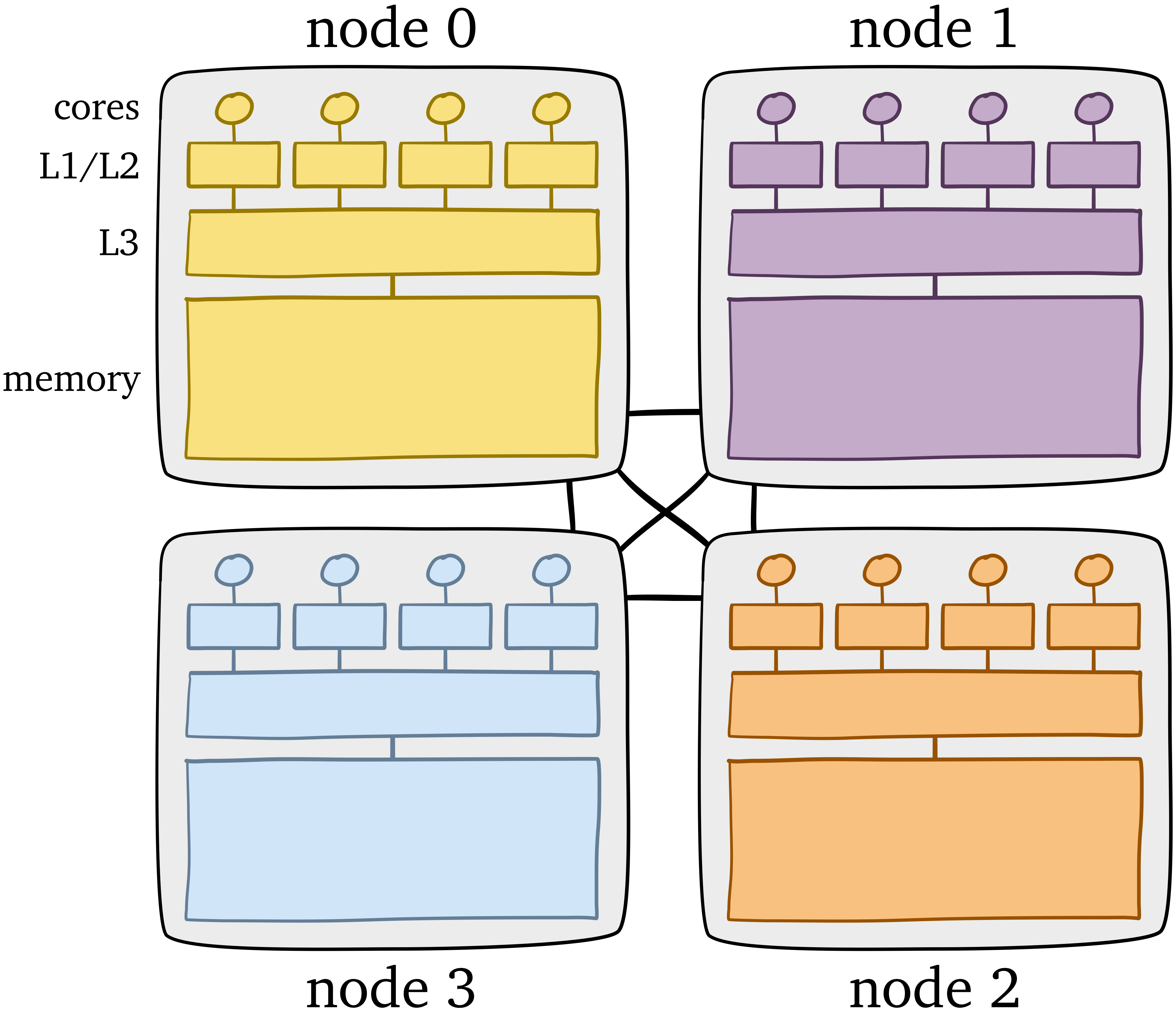
L3

memory

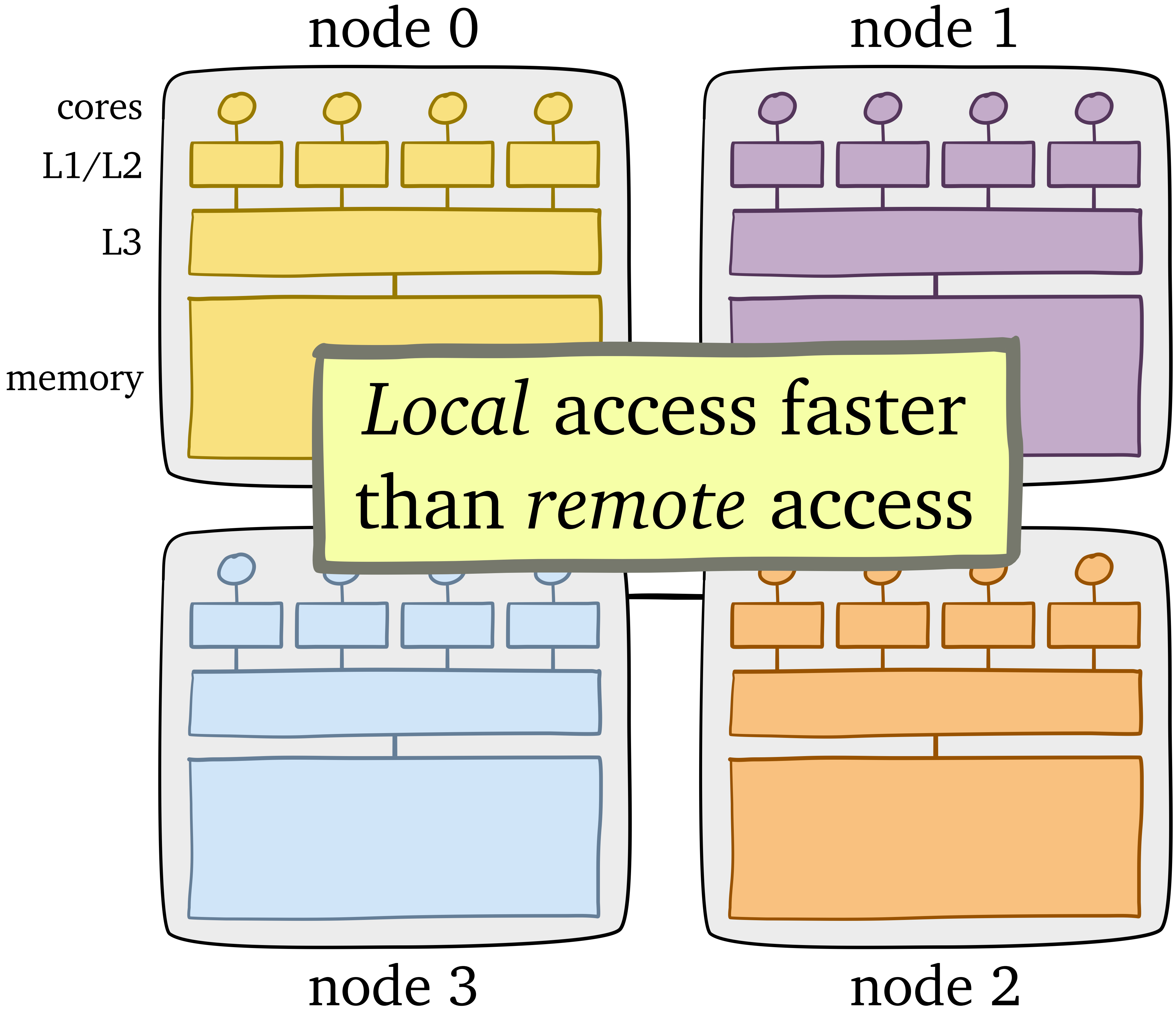
NUMA Architectures



NUMA Architectures

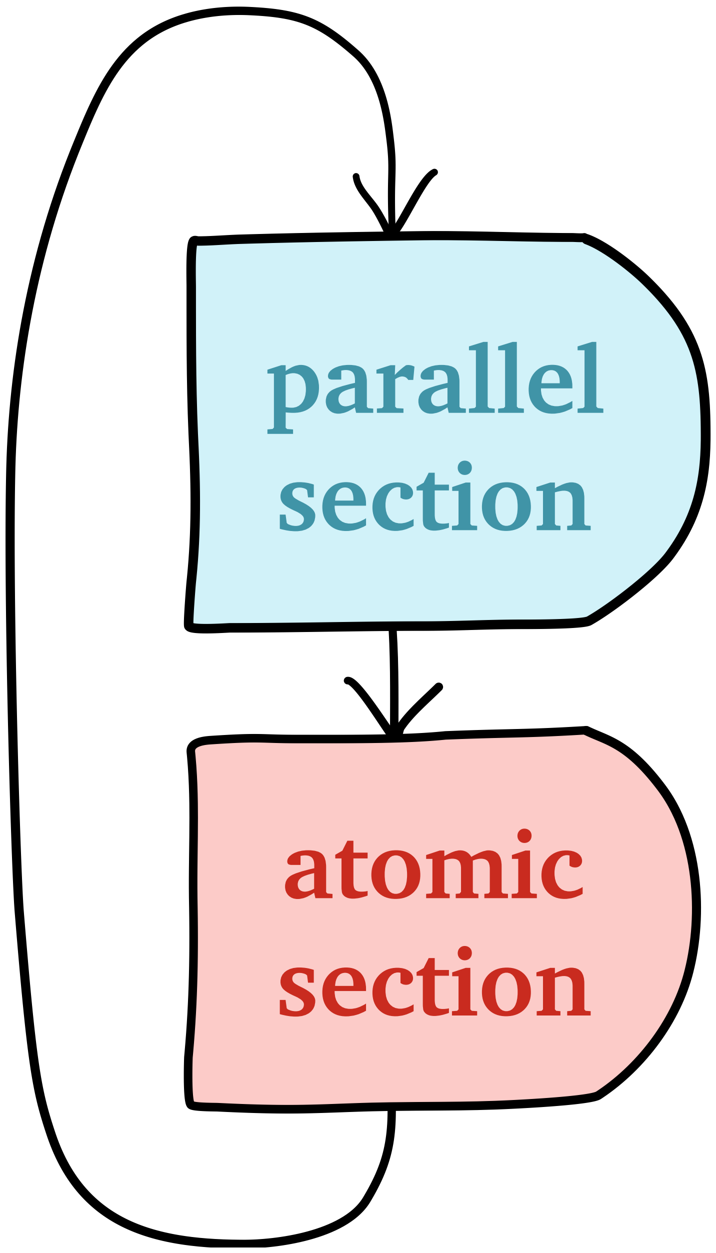


NUMA Architectures

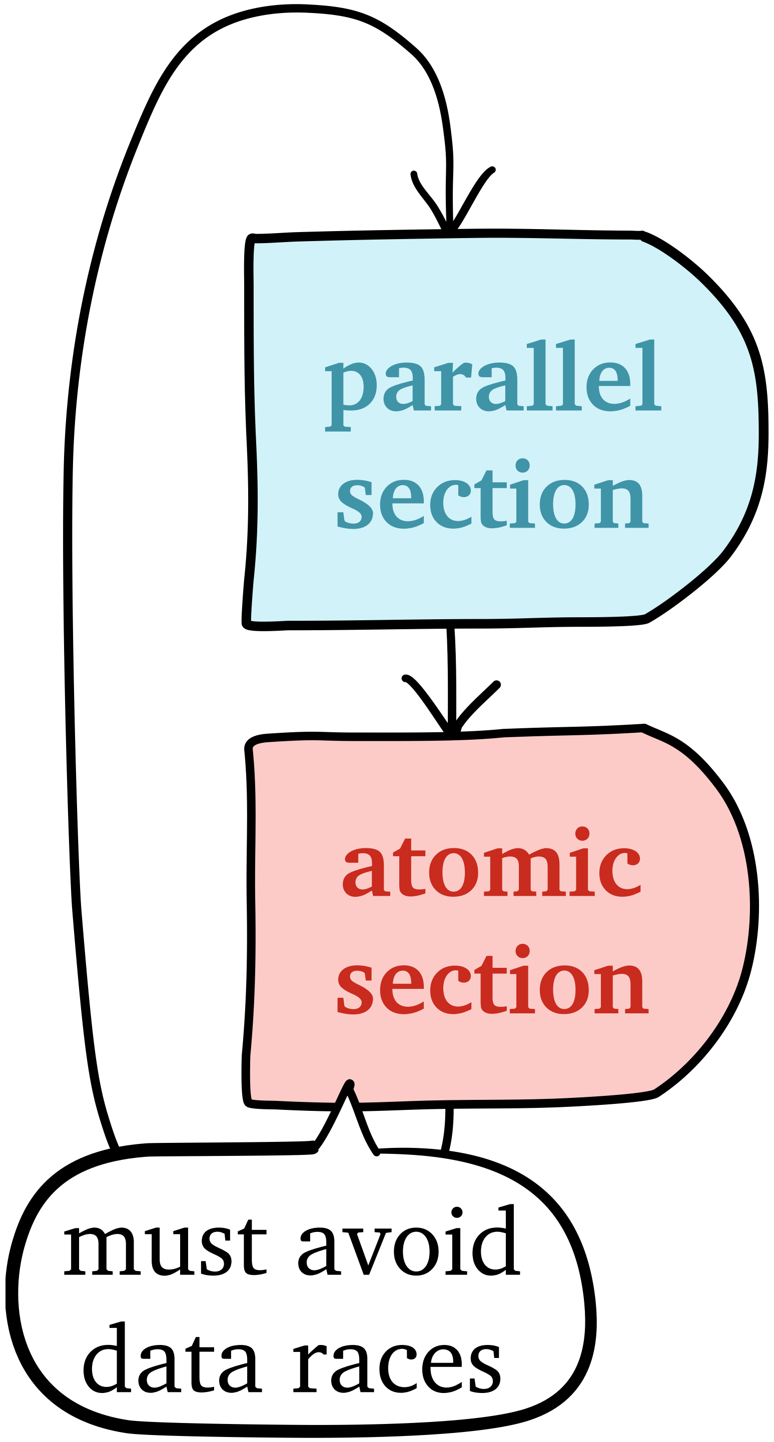


Local access faster than *remote* access

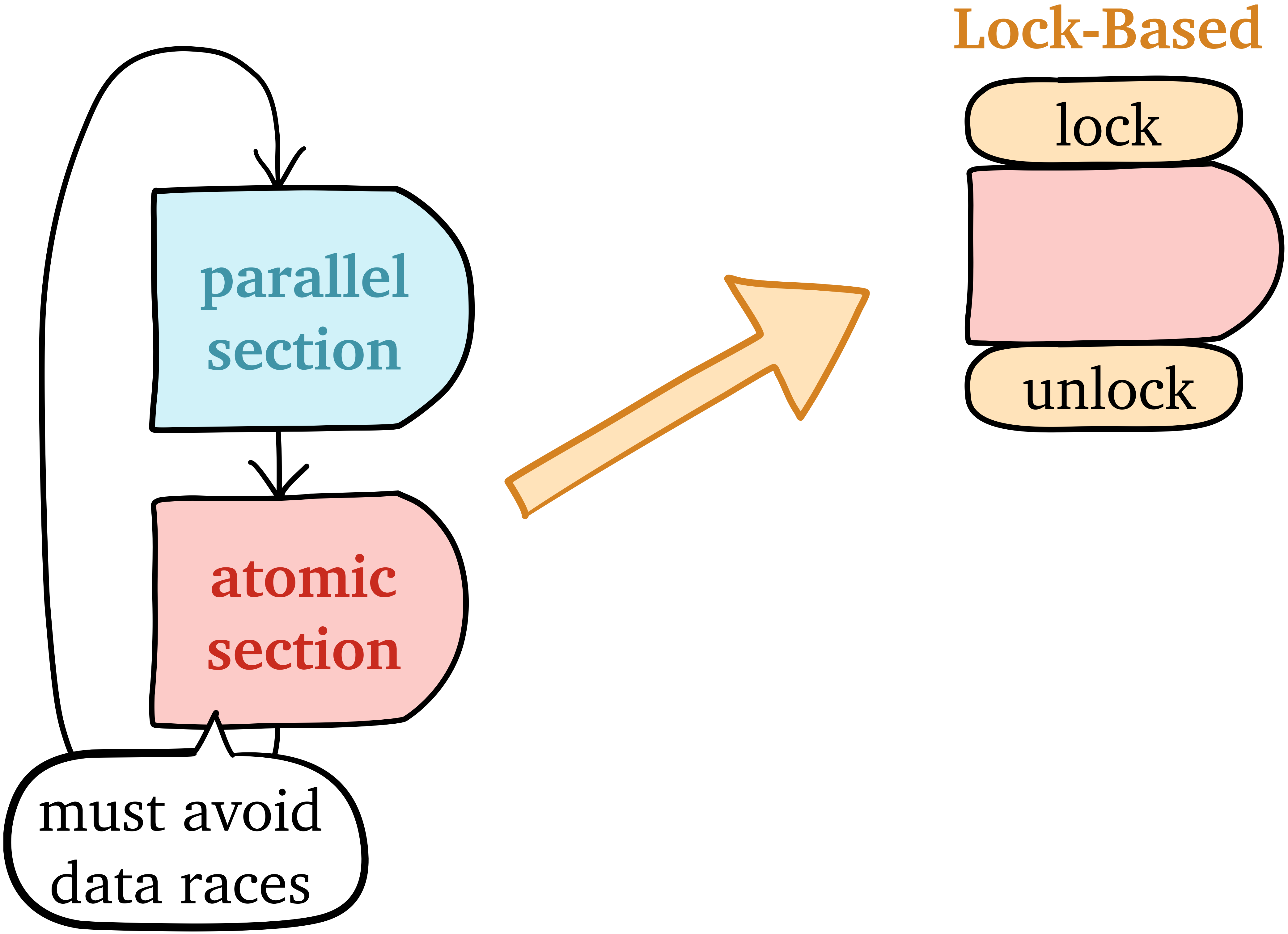
Concurrent Programs



Concurrent Programs



Concurrent Programs



Lock-Based

lock

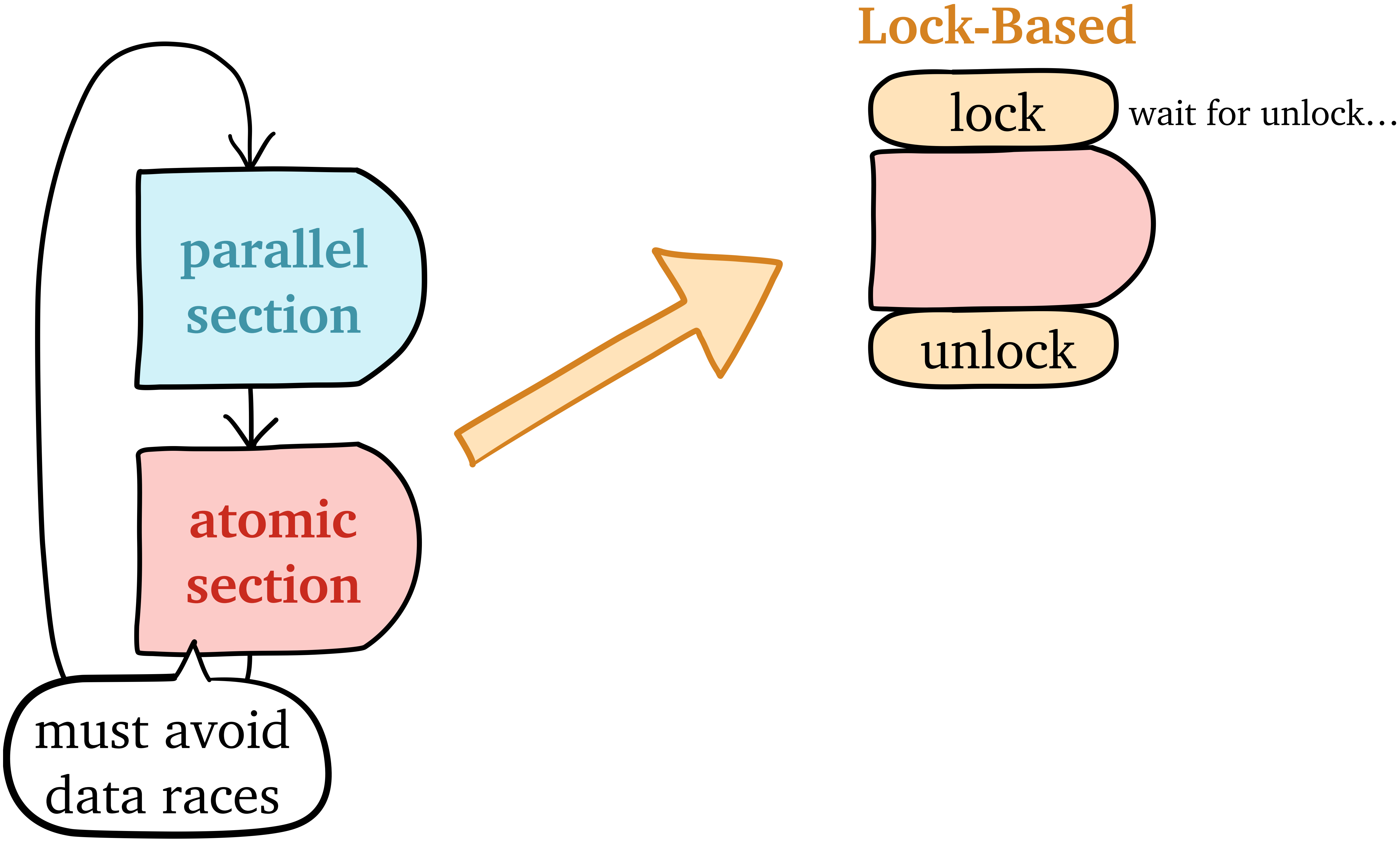
unlock

parallel section

atomic section

must avoid data races

Concurrent Programs



parallel section

atomic section

must avoid data races

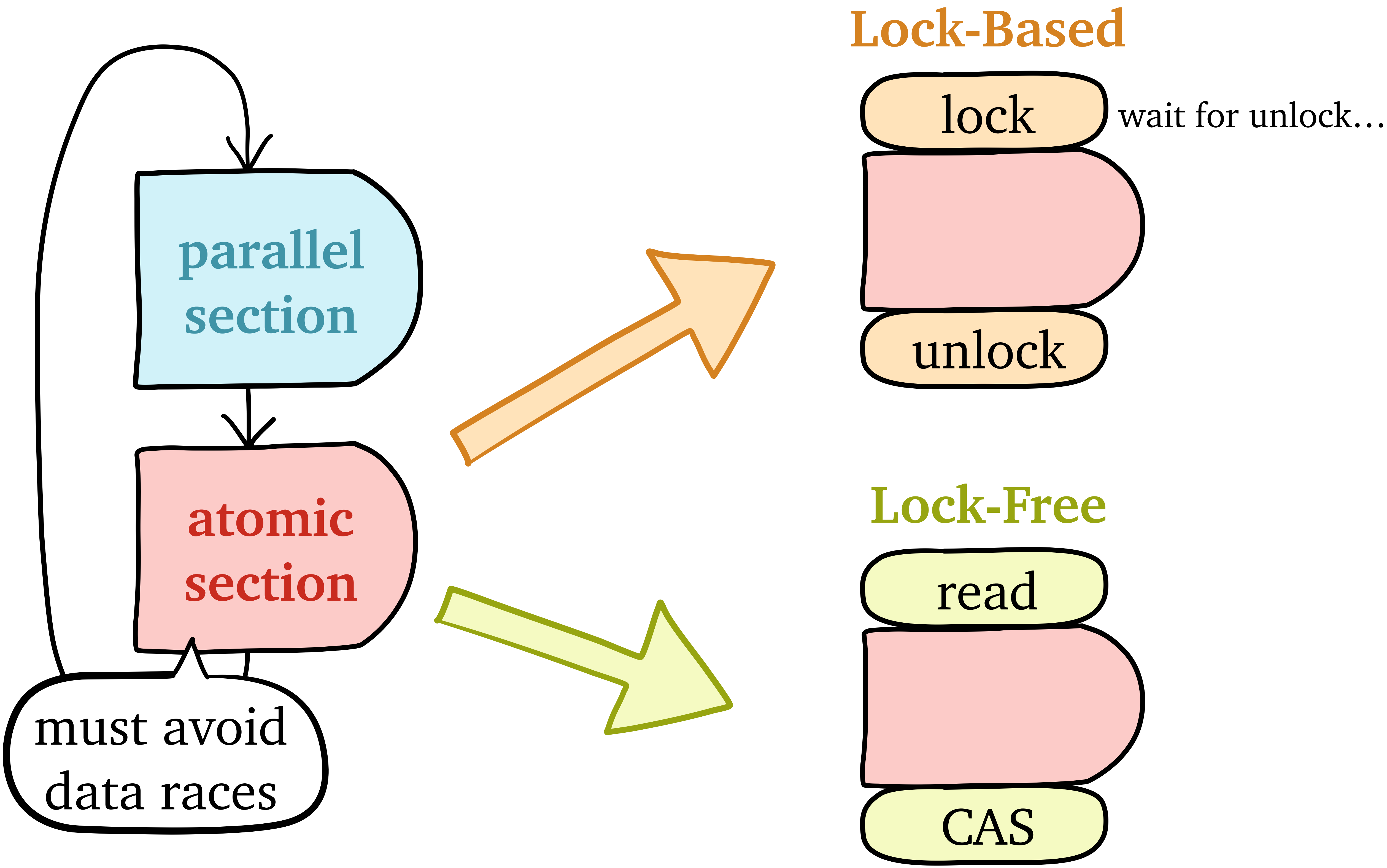
Lock-Based

lock

wait for unlock…

unlock

Concurrent Programs



parallel section

atomic section

must avoid data races

Lock-Based

lock

wait for unlock…

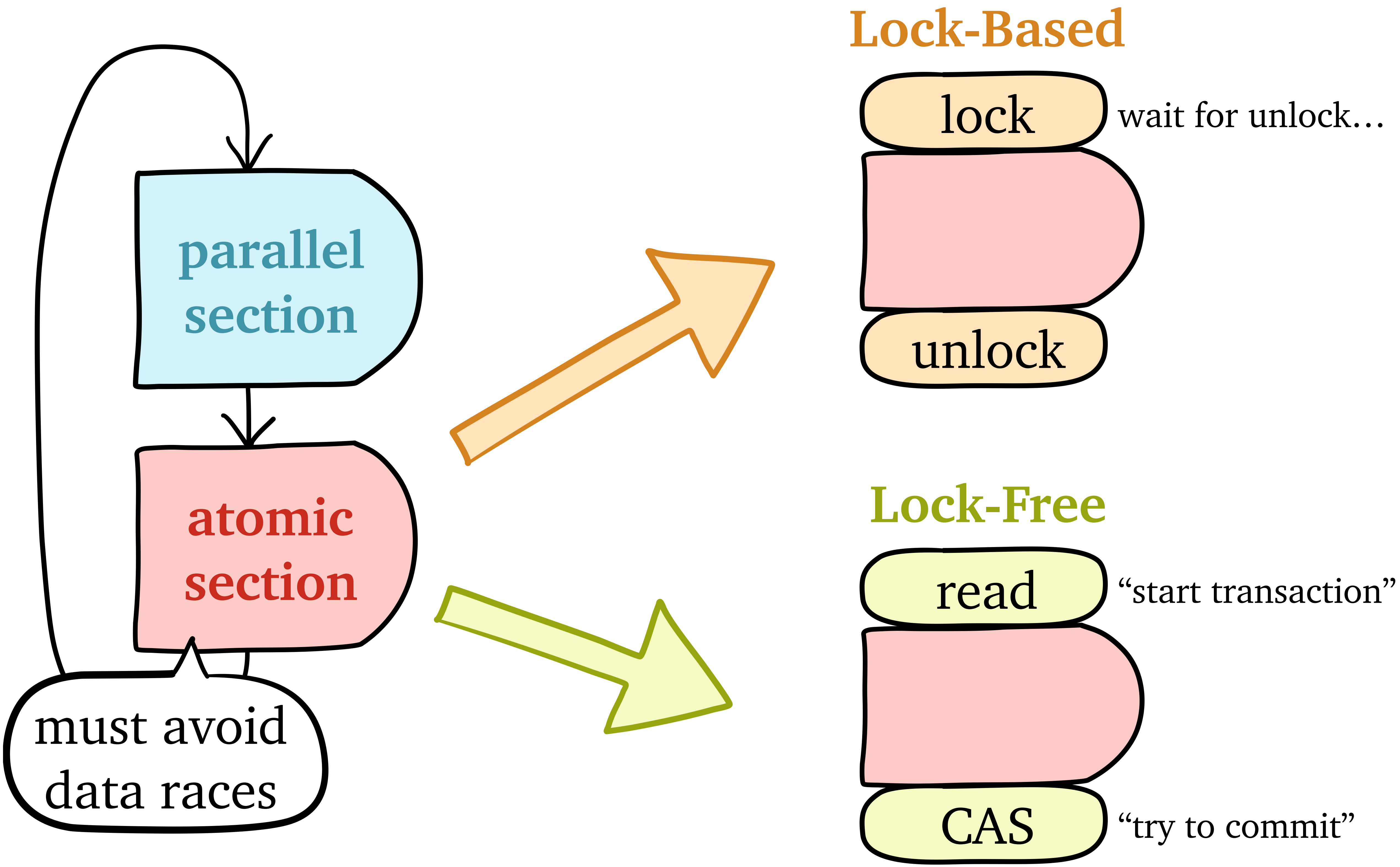
unlock

Lock-Free

read

CAS

Concurrent Programs



parallel section

atomic section

must avoid data races

Lock-Based

lock

wait for unlock…

unlock

Lock-Free

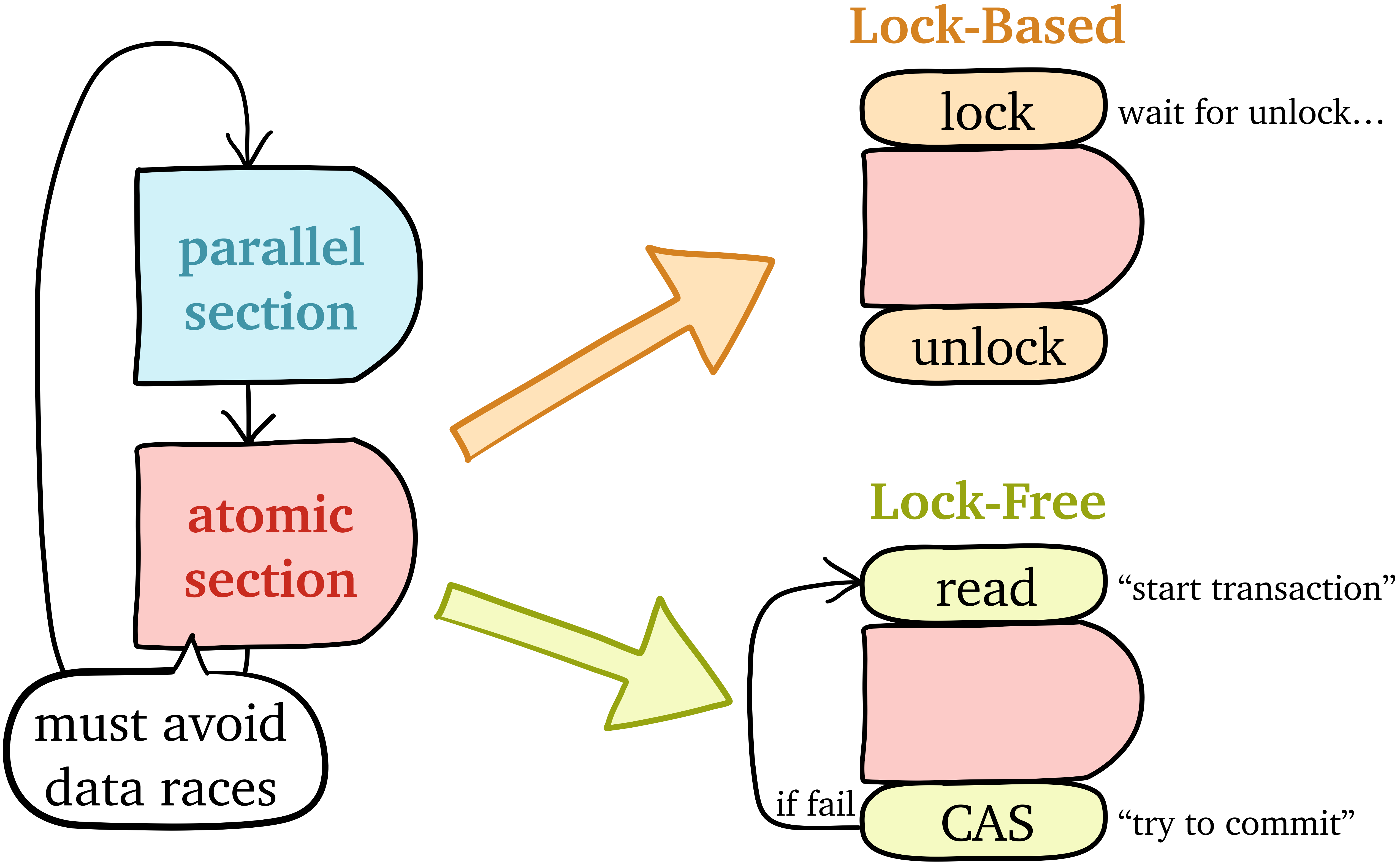
read

“start transaction”

CAS

“try to commit”

Concurrent Programs



parallel section

atomic section

must avoid data races

Lock-Based

lock

wait for unlock…

unlock

Lock-Free

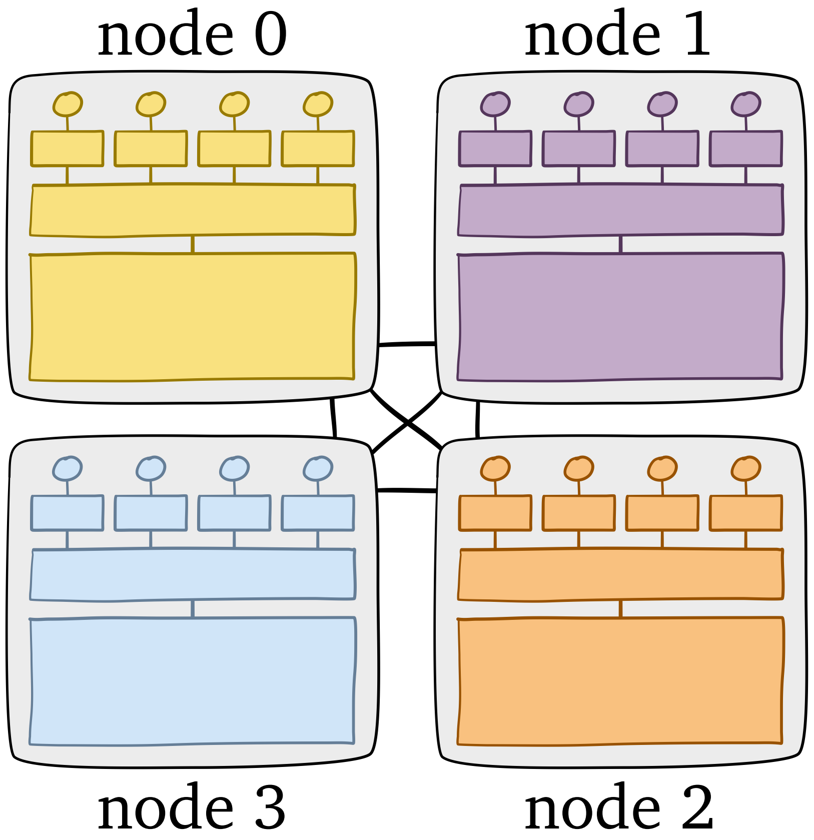
read

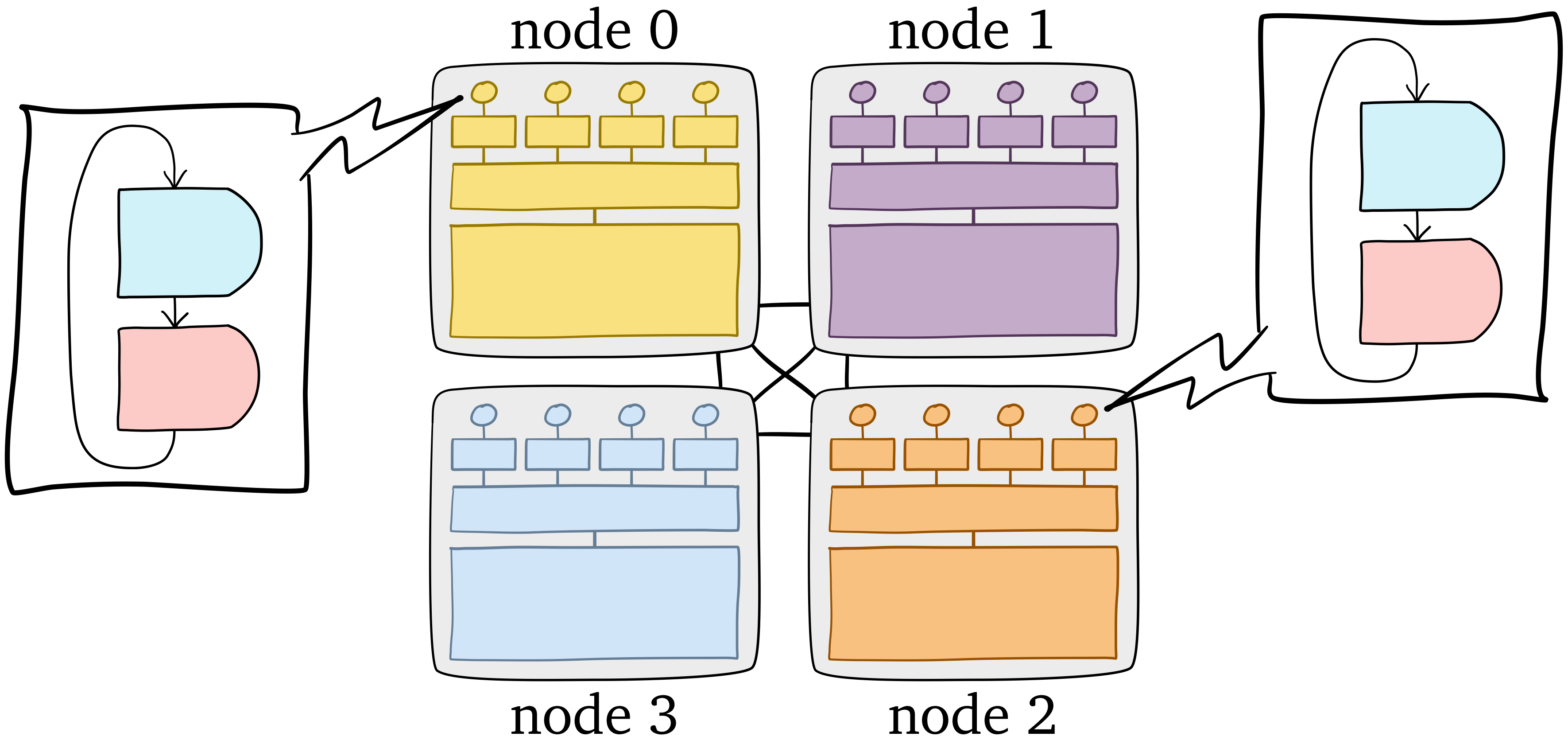
“start transaction”

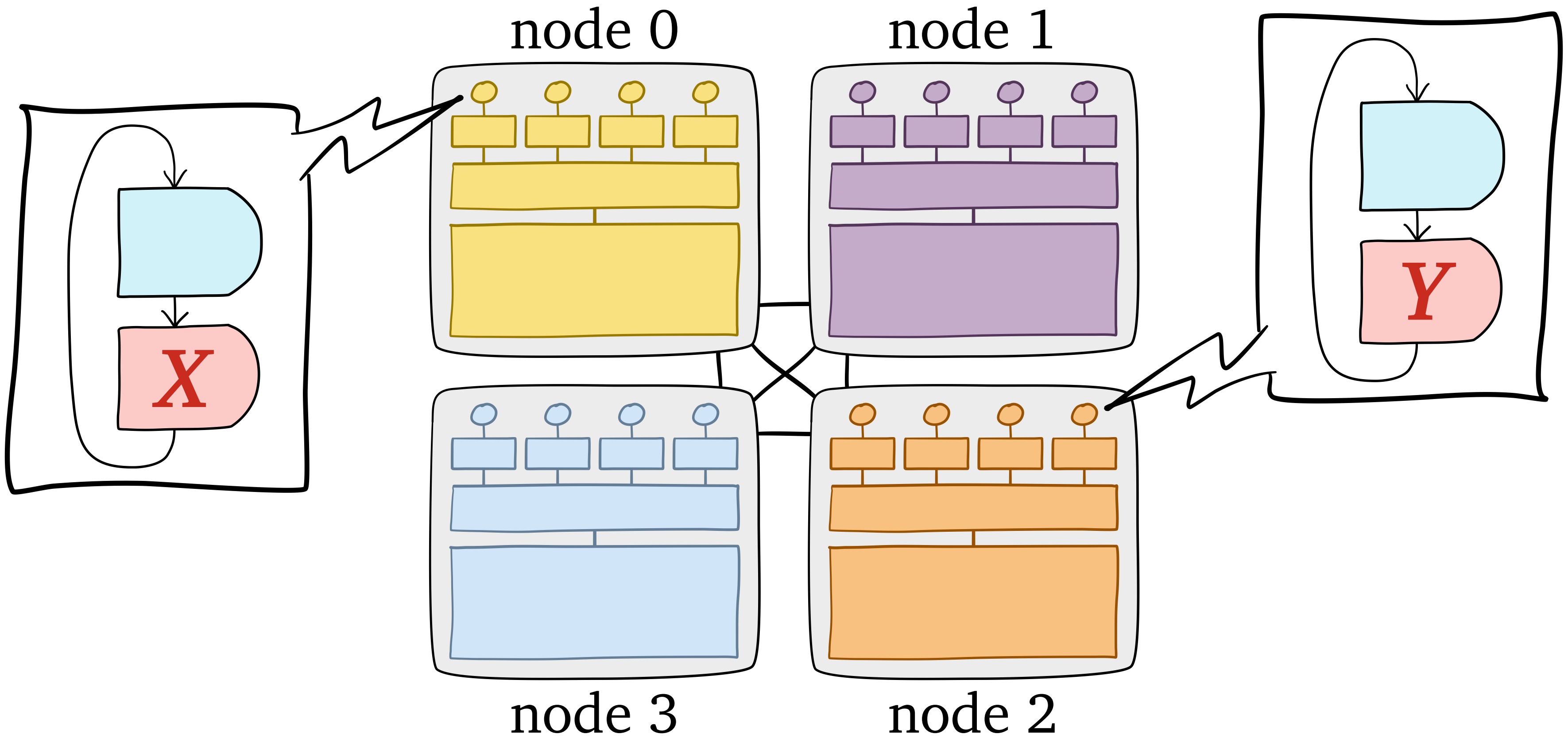
if fail

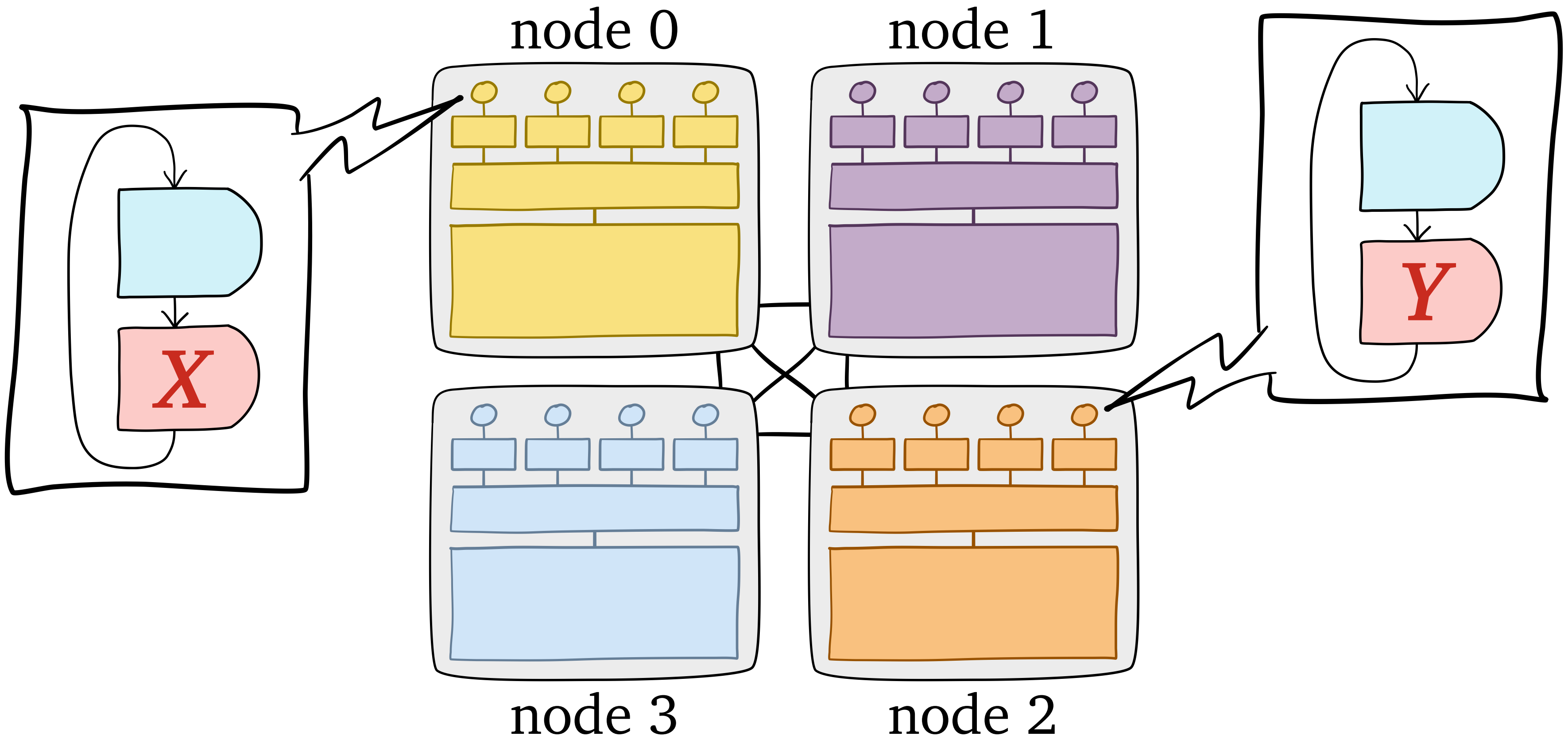
CAS

“try to commit”

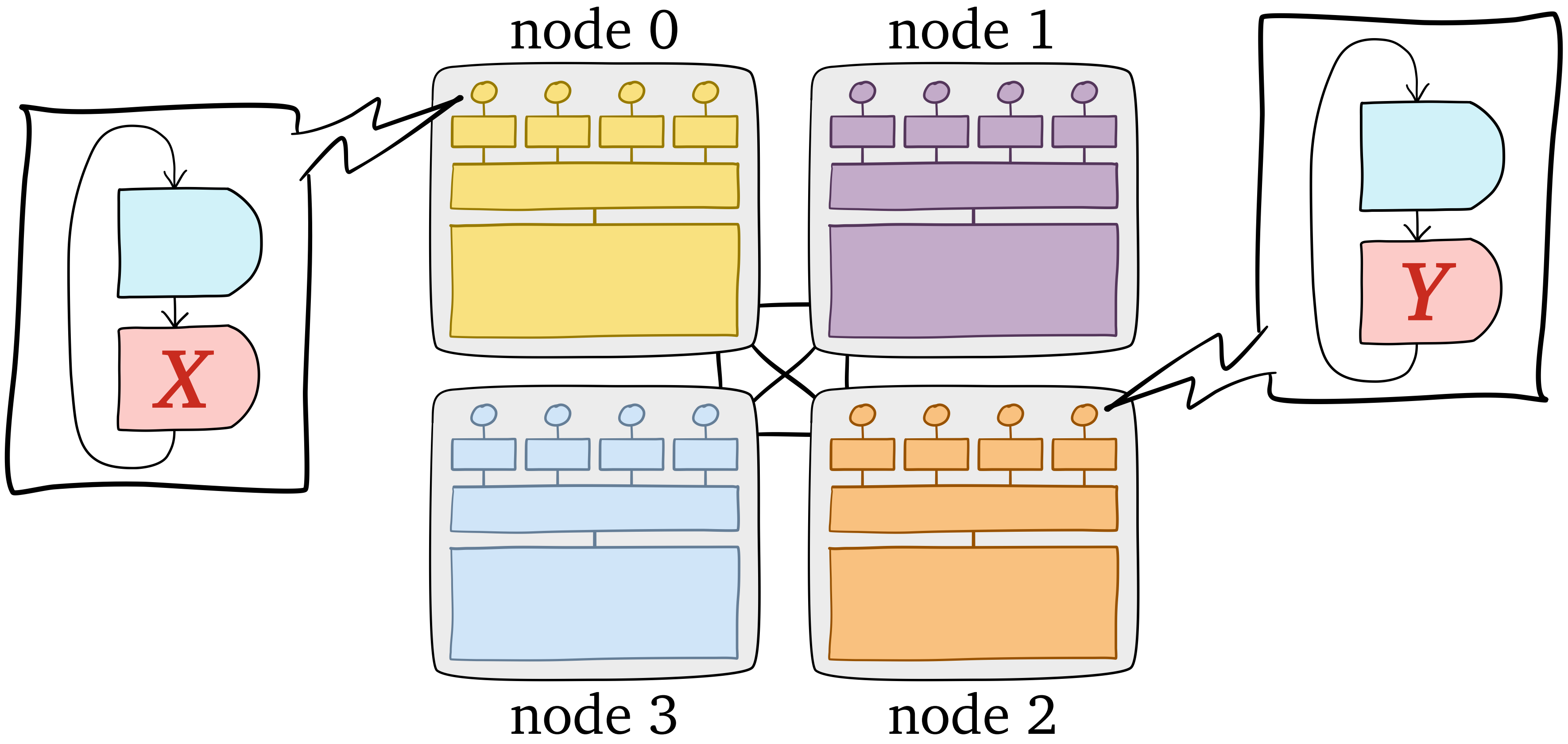

node 0
node 1
node 3
node 2


node 0
node 1
node 3
node 2


node 0
node 1
X
Y
node 3
node 2

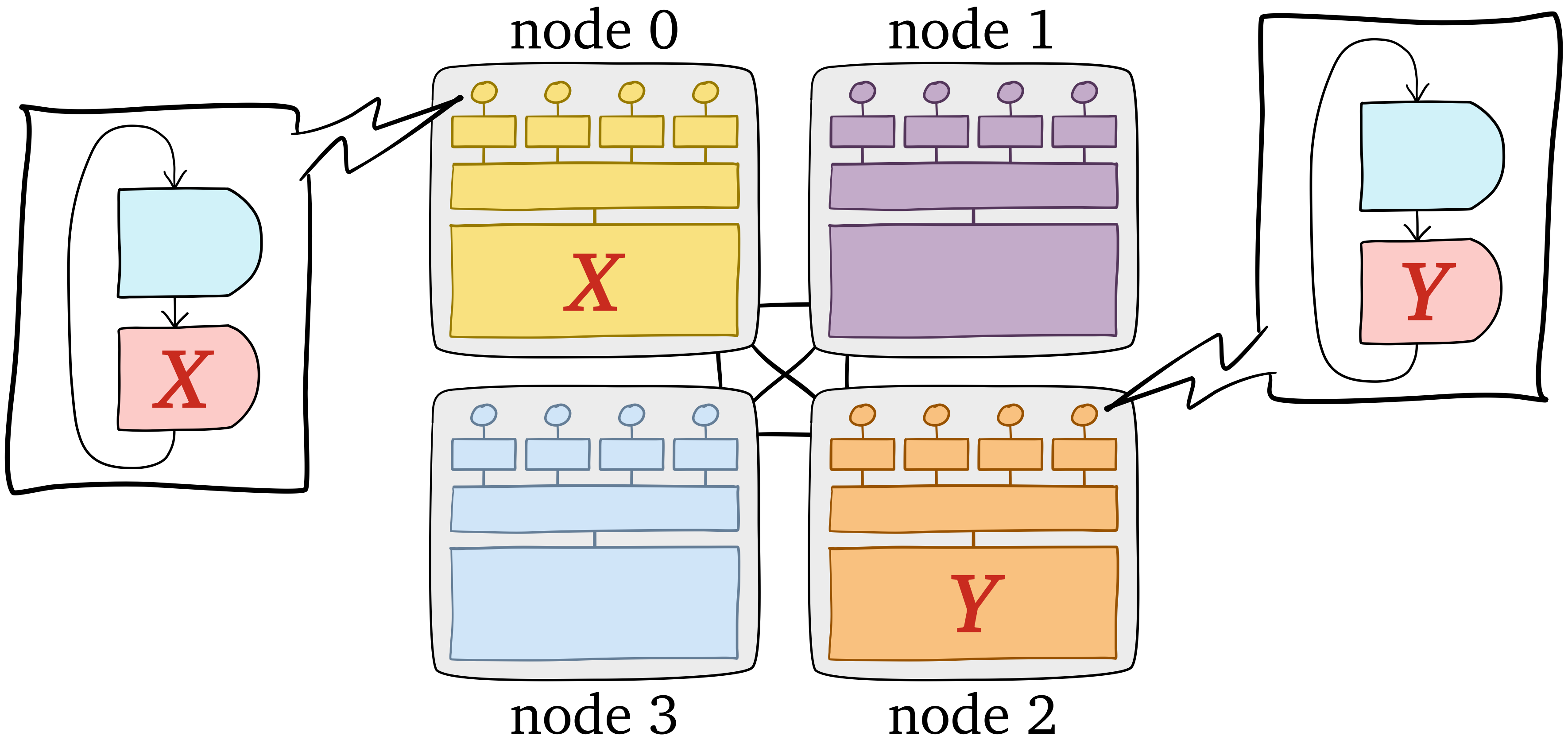


Question: where should we allocate shared data?



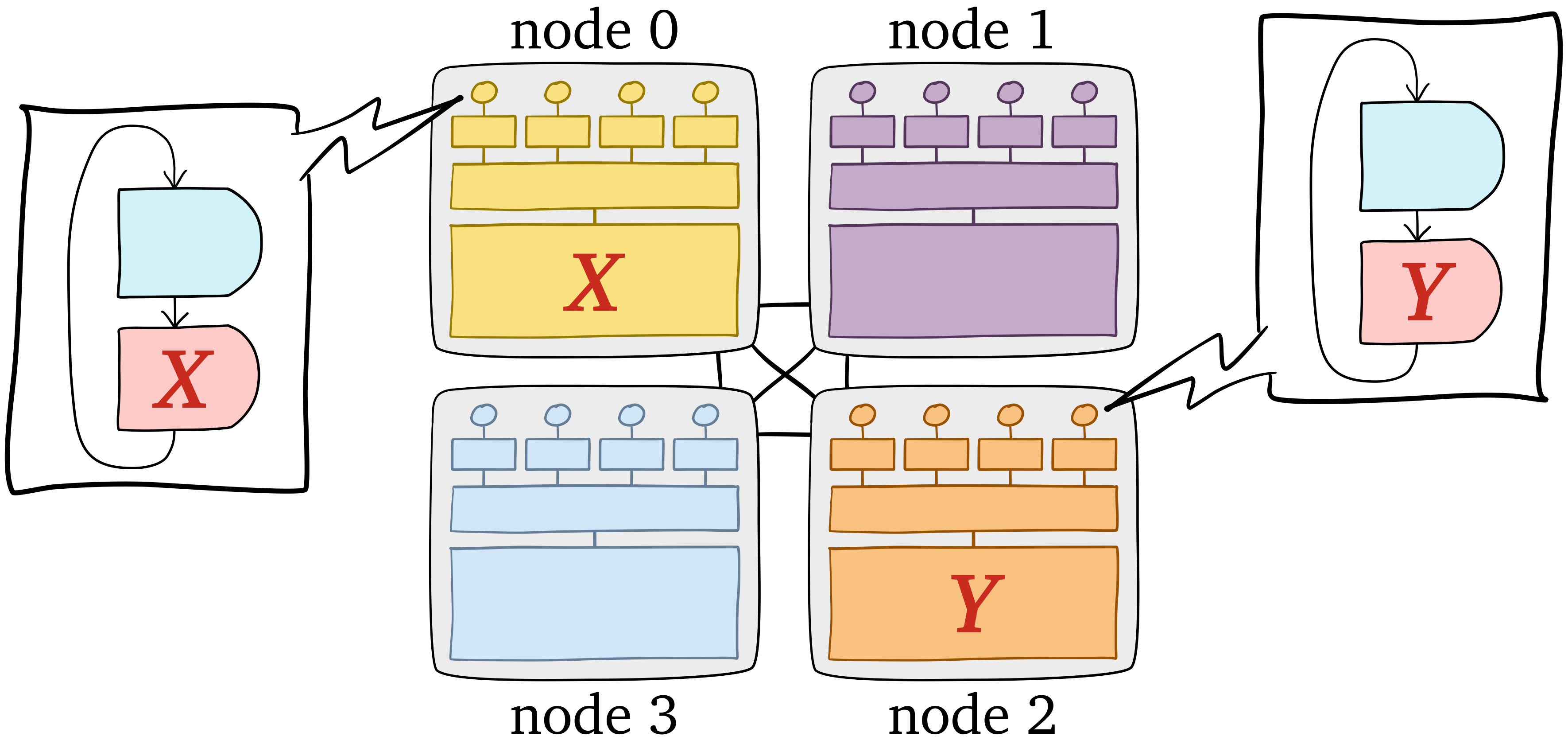
Question: where should we allocate shared data?

Conventional wisdom: put data near computation



Question: where should we allocate shared data?

Conventional wisdom: put data near computation



Question: where should we allocate shared data?

Conventional wisdom: put data near computation

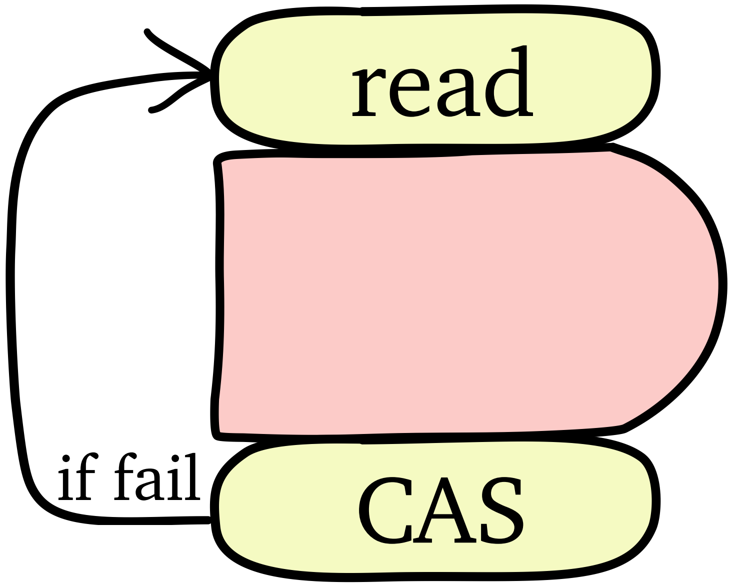
Problem: conventional wisdom is for lock-based algorithms

NUMA Architectures
+
Concurrent Programs

NUMA Architectures

+

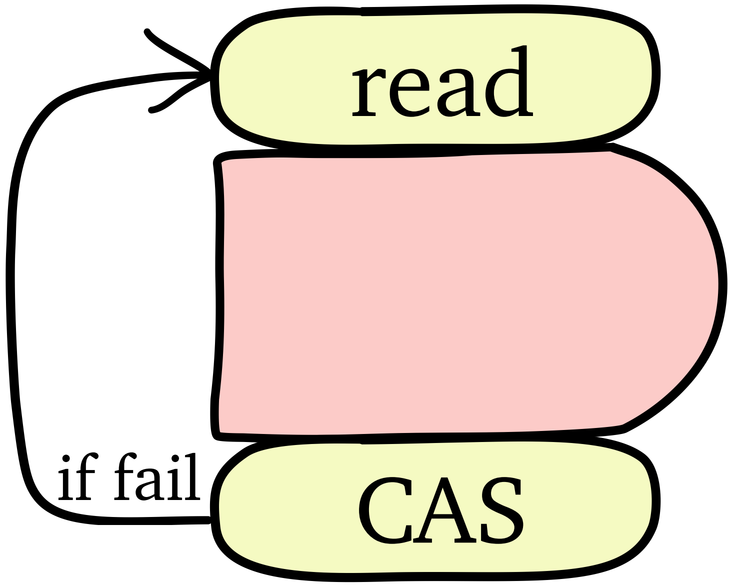
Lock-Free Algorithms



NUMA Architectures

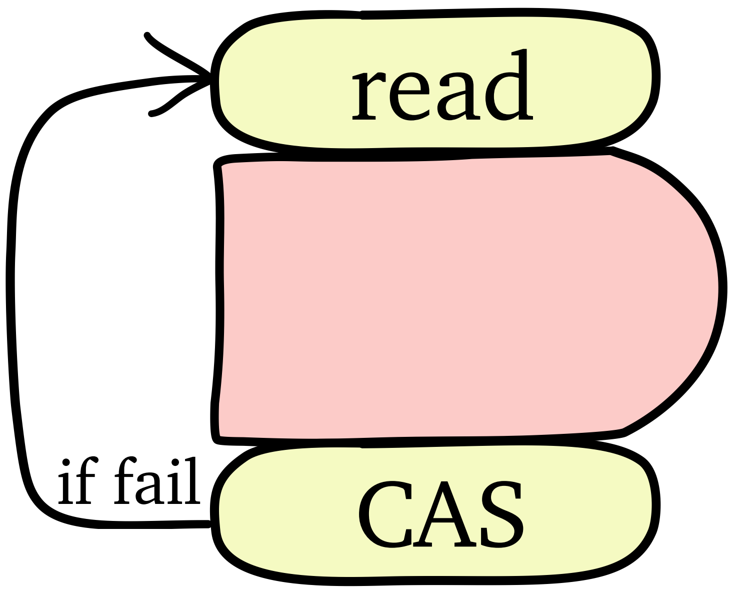
+

Lock-Free Algorithms



Question: where should we allocate shared data?

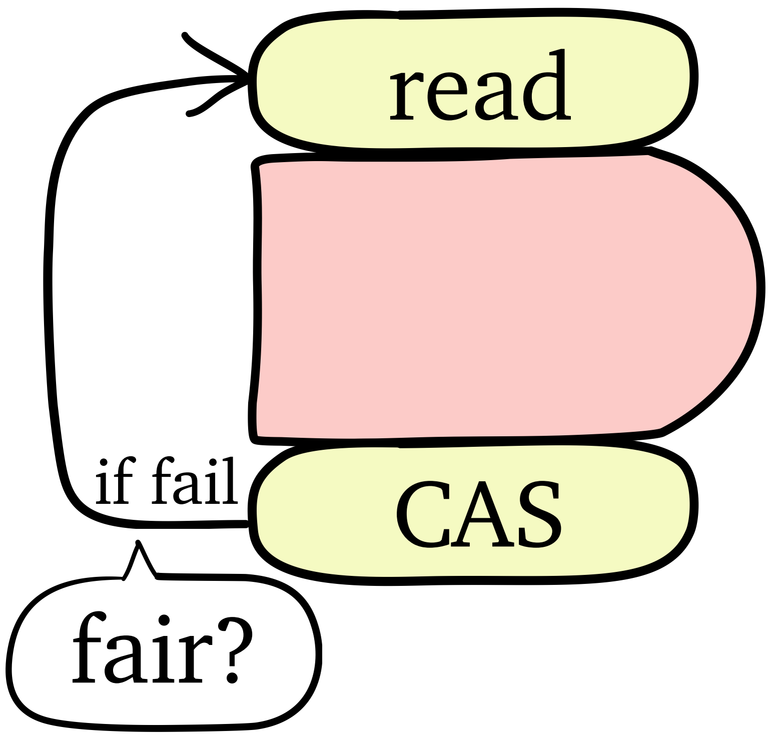
NUMA Architectures
+
Lock-Free Algorithms



Question: where should we allocate shared data?

Question: does NUMA treat remote cores fairly?

NUMA Architectures
+
Lock-Free Algorithms



Question: where should we allocate shared data?

Question: does NUMA treat remote cores fairly?

Our Contributions

Our Contributions

1. *New tool* revealing NUMA's effects on **lock-free** algorithms

Our Contributions

Severus

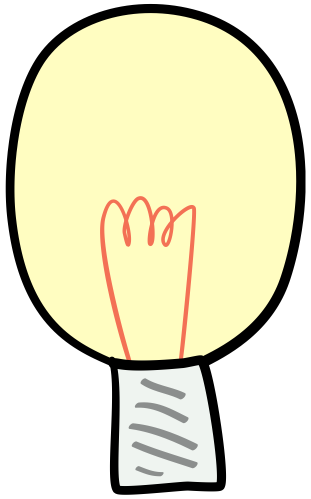
1. *New tool* revealing NUMA's effects on **lock-free** algorithms

Our Contributions

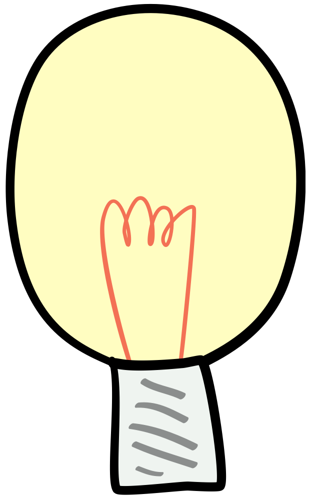
Severus

1. *New tool* revealing NUMA's effects on **lock-free** algorithms

2. *Case studies* of two machines:
   - AMD Opteron 6278 (Interlagos)
   - Intel Xeon E7-8867 v4 (Broadwell-EX)



Idea: look at *schedule* of memory accesses



Idea: look at *schedule* of memory accesses

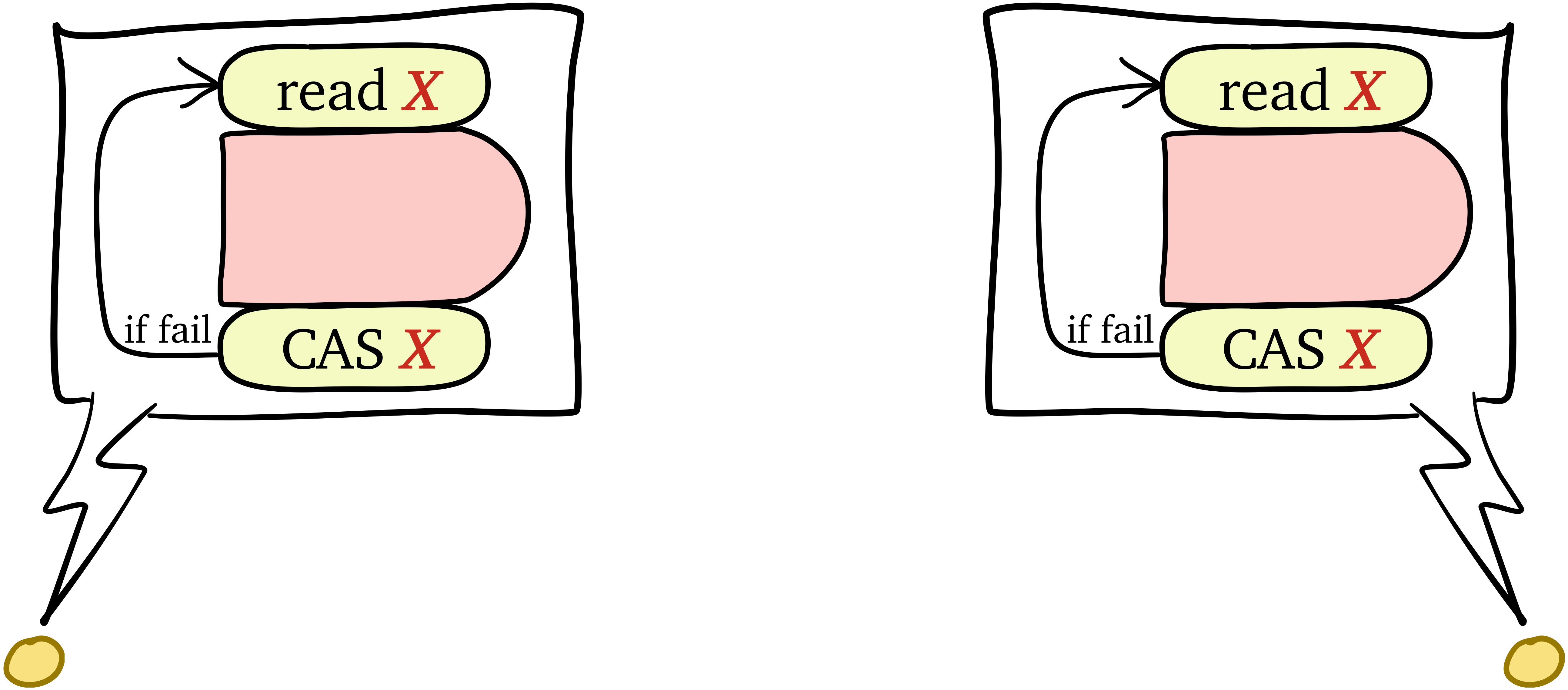
ordering

Schedule Matters

ordering

Schedule Matters

ordering



read *X*

if fail

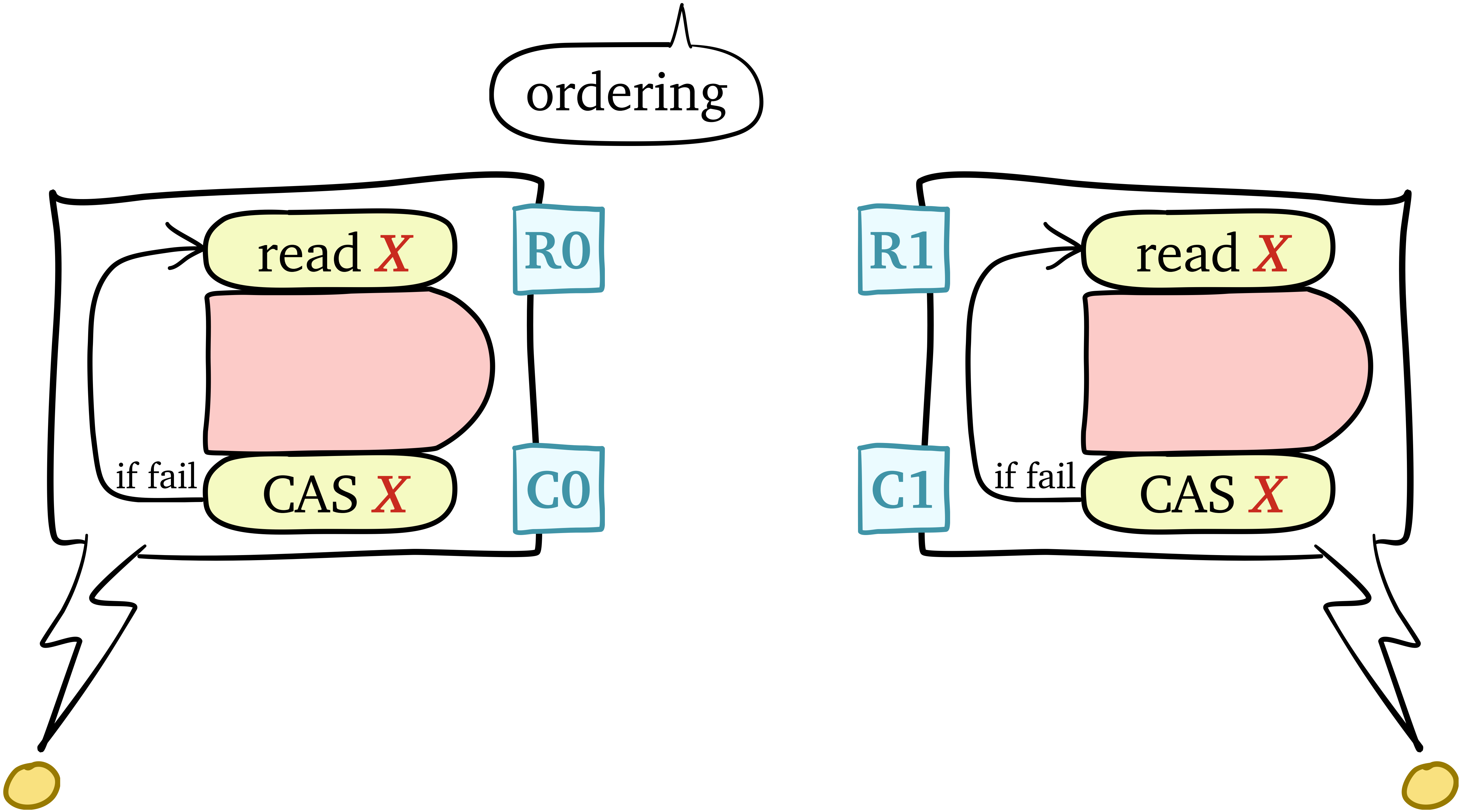
CAS *X*

read *X*

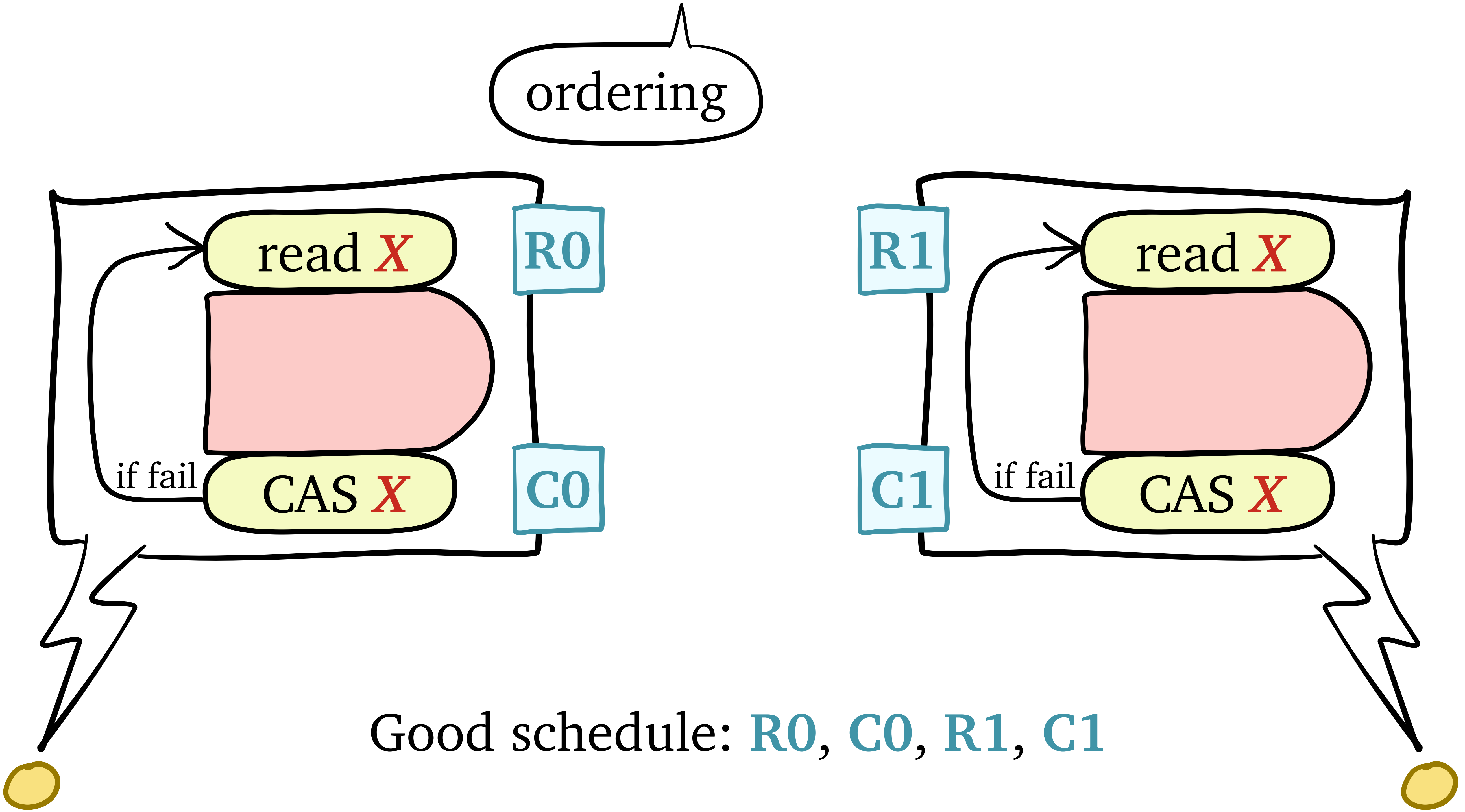
if fail

CAS *X*

Schedule Matters

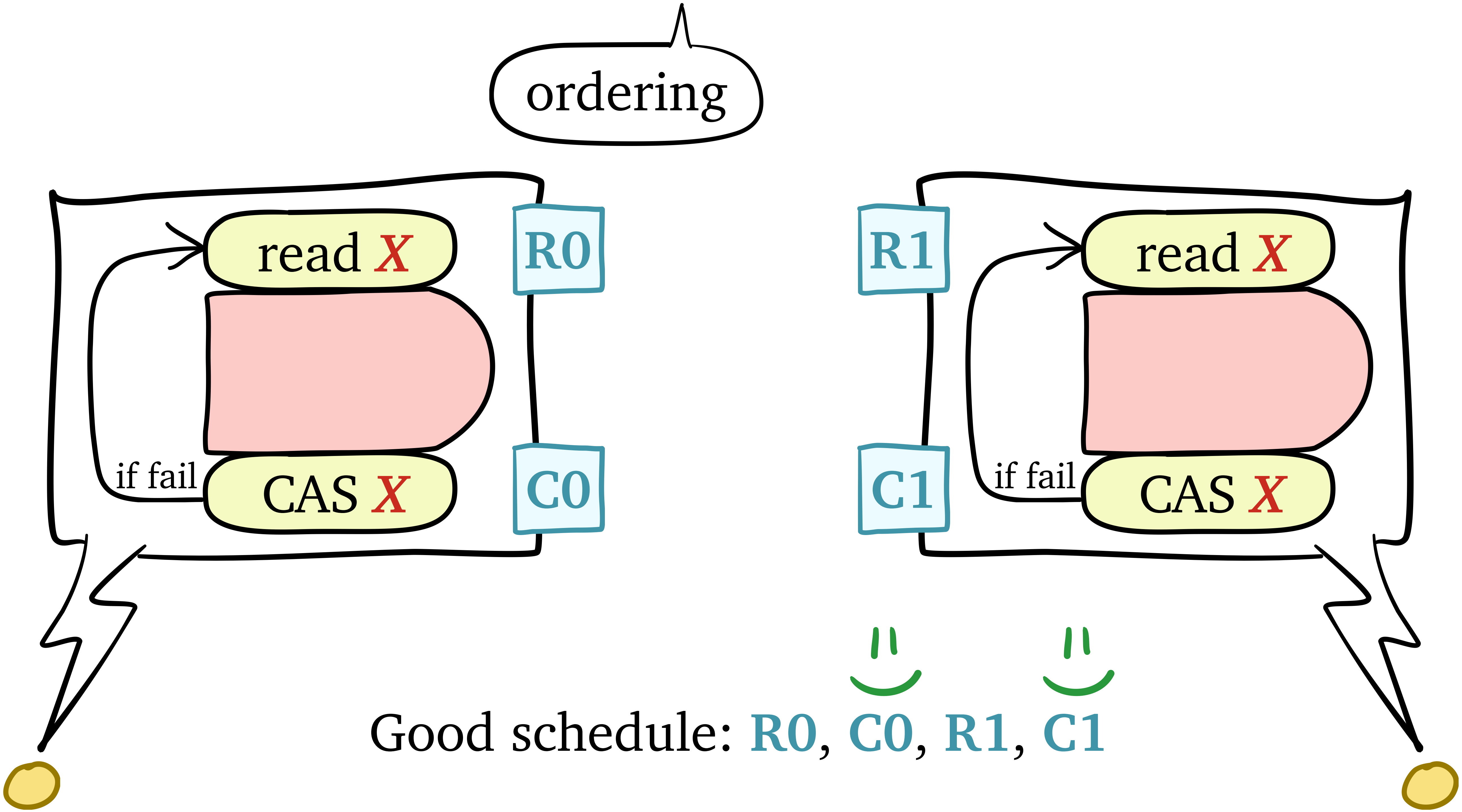


Schedule Matters



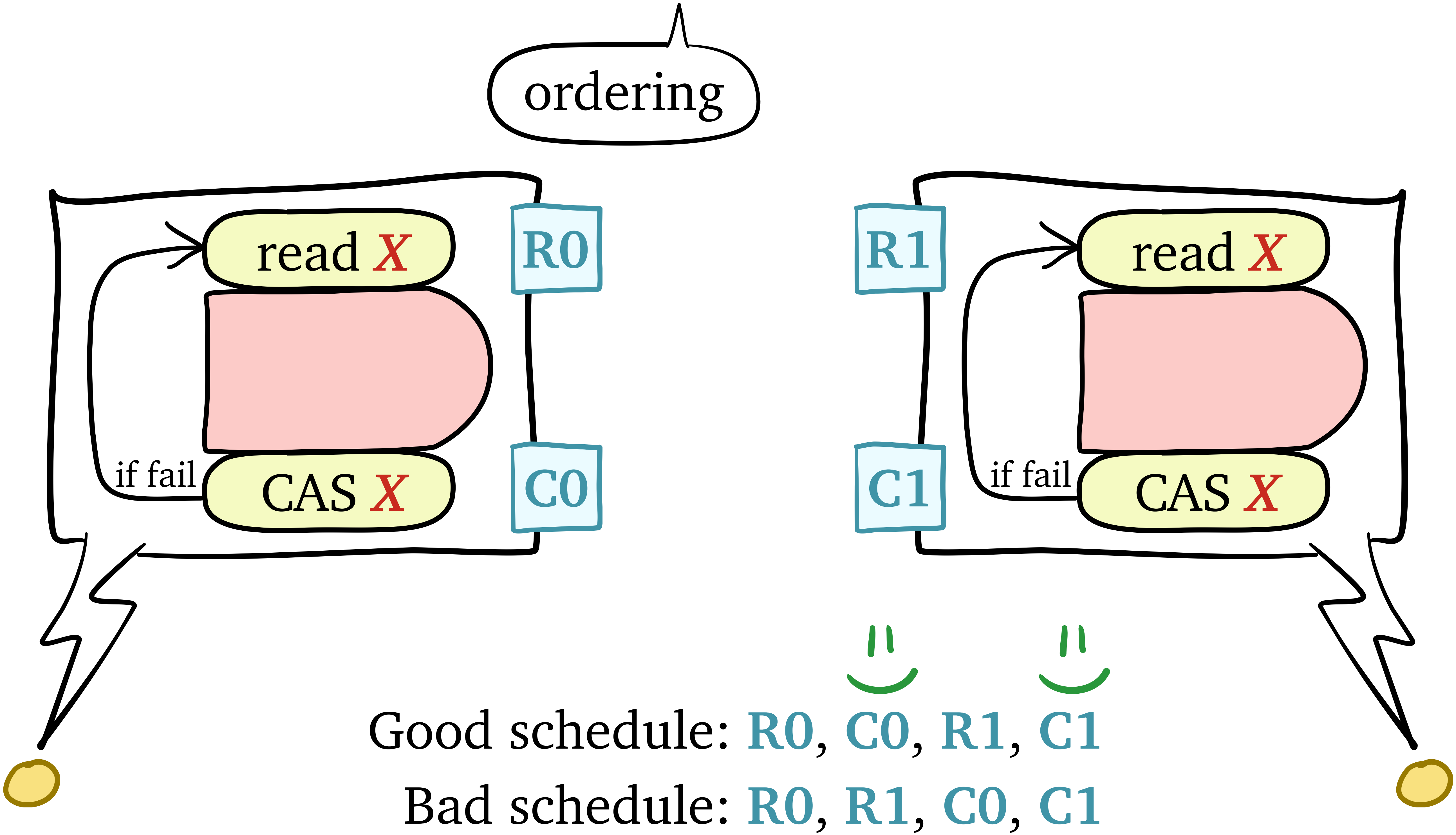
Good schedule: **R0**, **C0**, **R1**, **C1**

Schedule Matters



Good schedule: R0, C0, R1, C1

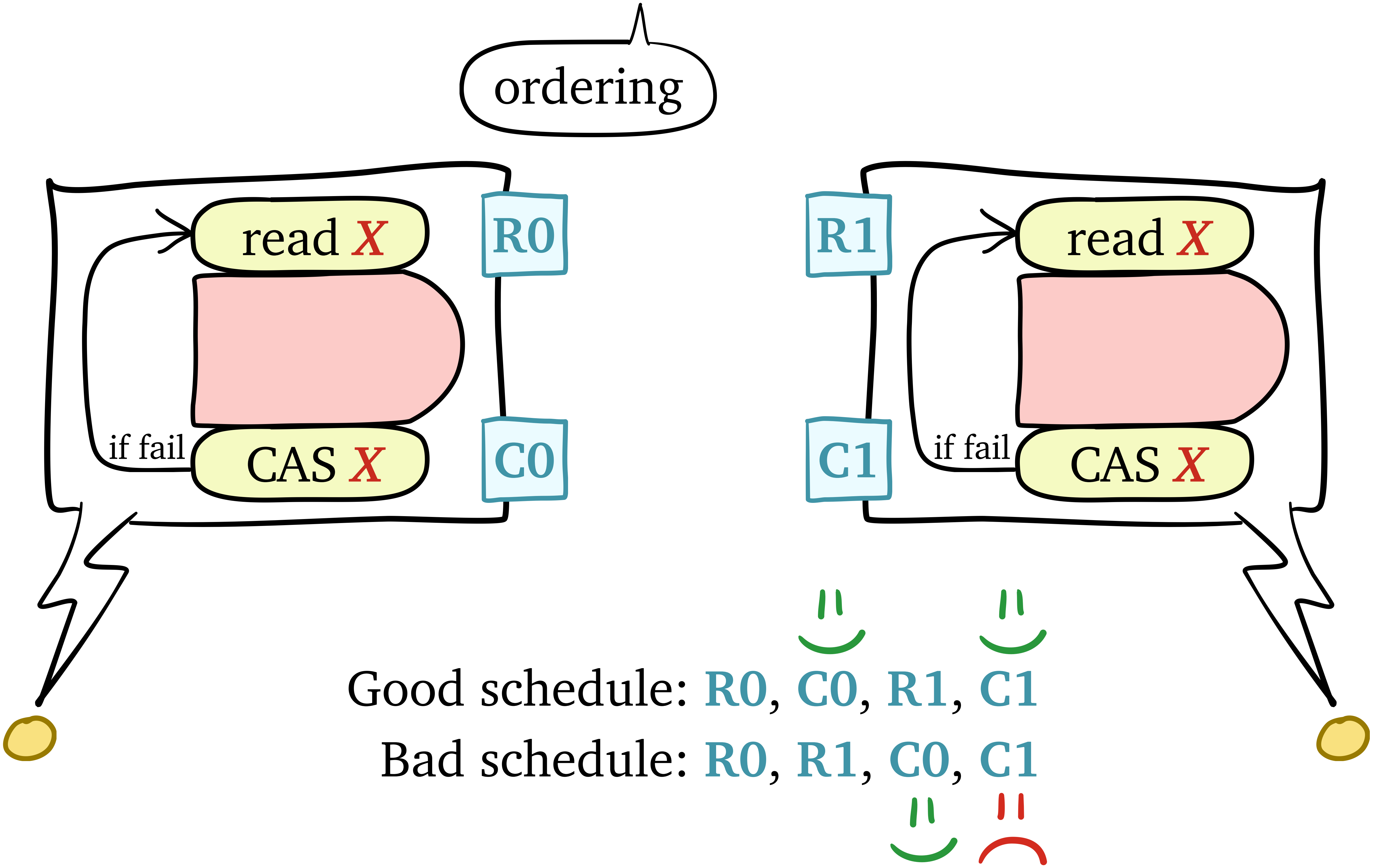
Schedule Matters



Good schedule: R0, C0, R1, C1

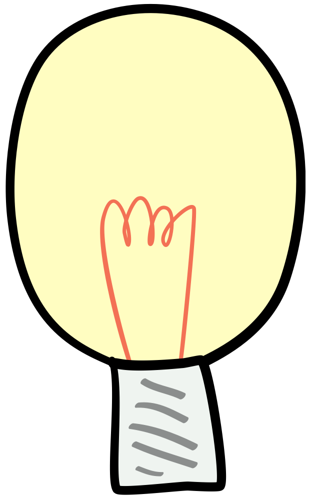
Bad schedule: R0, R1, C0, C1

Schedule Matters



Good schedule: R0, C0, R1, C1

Bad schedule: R0, R1, C0, C1



ordering

Idea: look at *schedule* of memory accesses

ordering

Idea: look at *schedule* of memory accesses

Problem: schedule depends on *complex hardware details*

ordering

Idea: look at *schedule* of memory accesses

Problem: schedule depends on *complex hardware details*

- cache coherence protocol

ordering

Idea: look at *schedule* of memory accesses

Problem: schedule depends on *complex hardware details*

- cache coherence protocol
- interconnect routing policy

New Tool: **Severus**

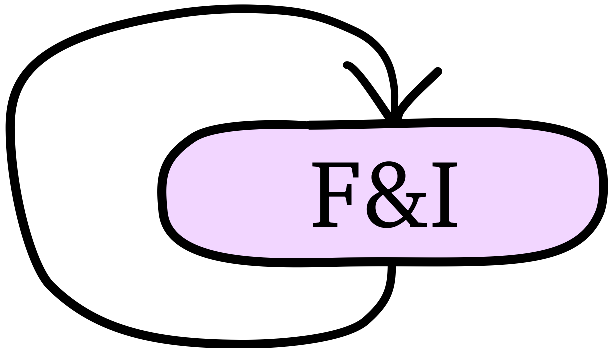
New Tool: **Severus**

Goal #1: *reveal schedule* of memory accesses

New Tool: Severus

Goal #1: *reveal schedule* of memory accesses

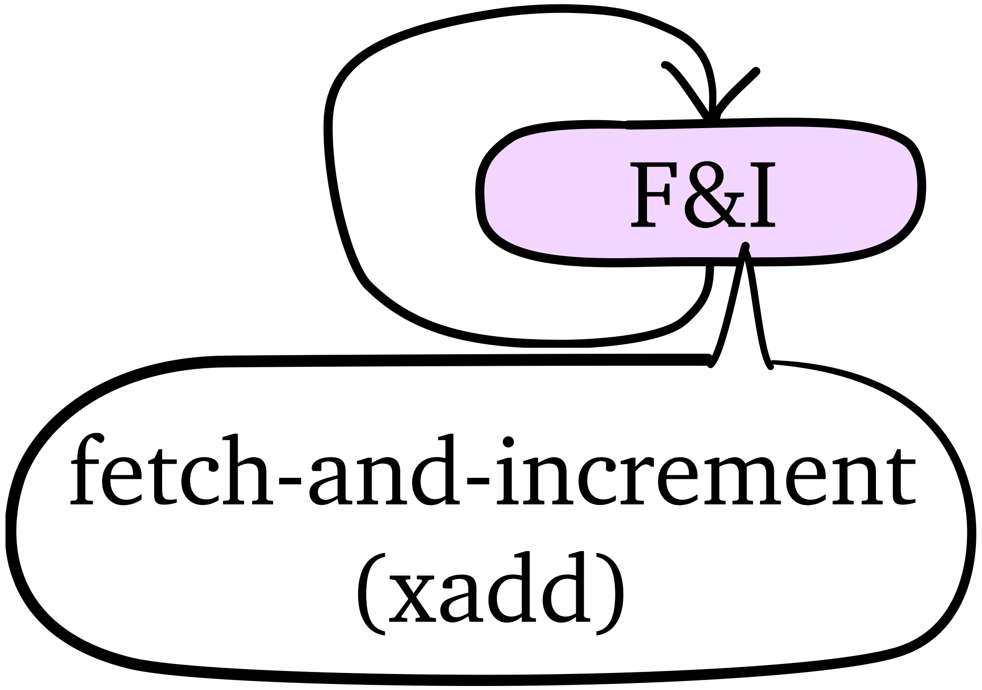
F&I Experiments



New Tool: **Severus**

Goal **#1**: *reveal schedule* of memory accesses

F&I Experiments

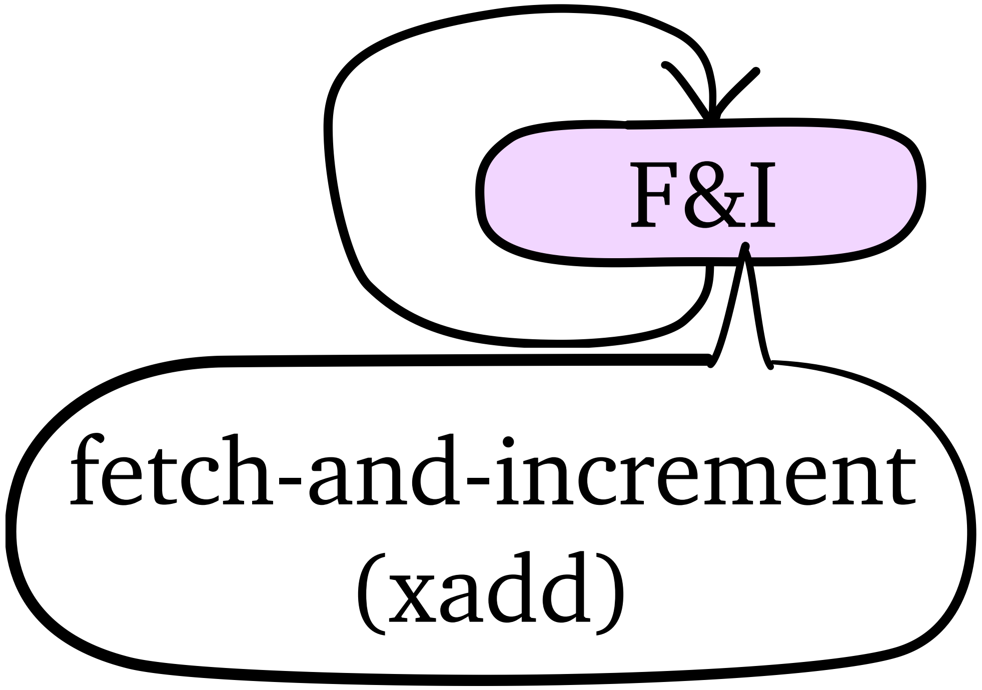


New Tool: Severus

Goal #1: *reveal schedule* of memory accesses

Goal #2: *simulate* lock-free algorithms

F&I Experiments

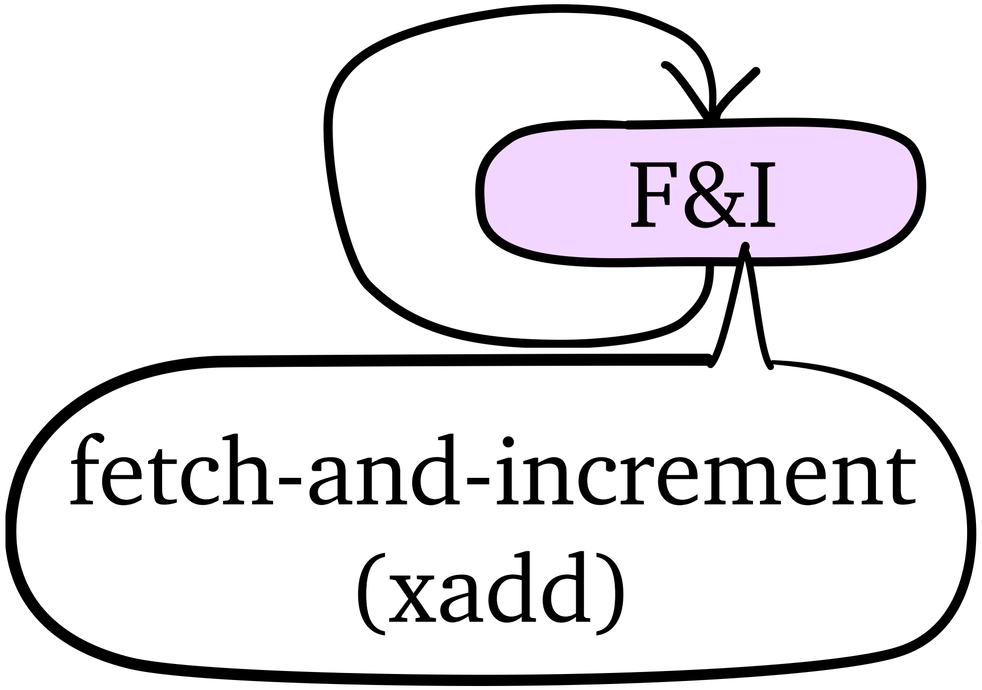


New Tool: Severus

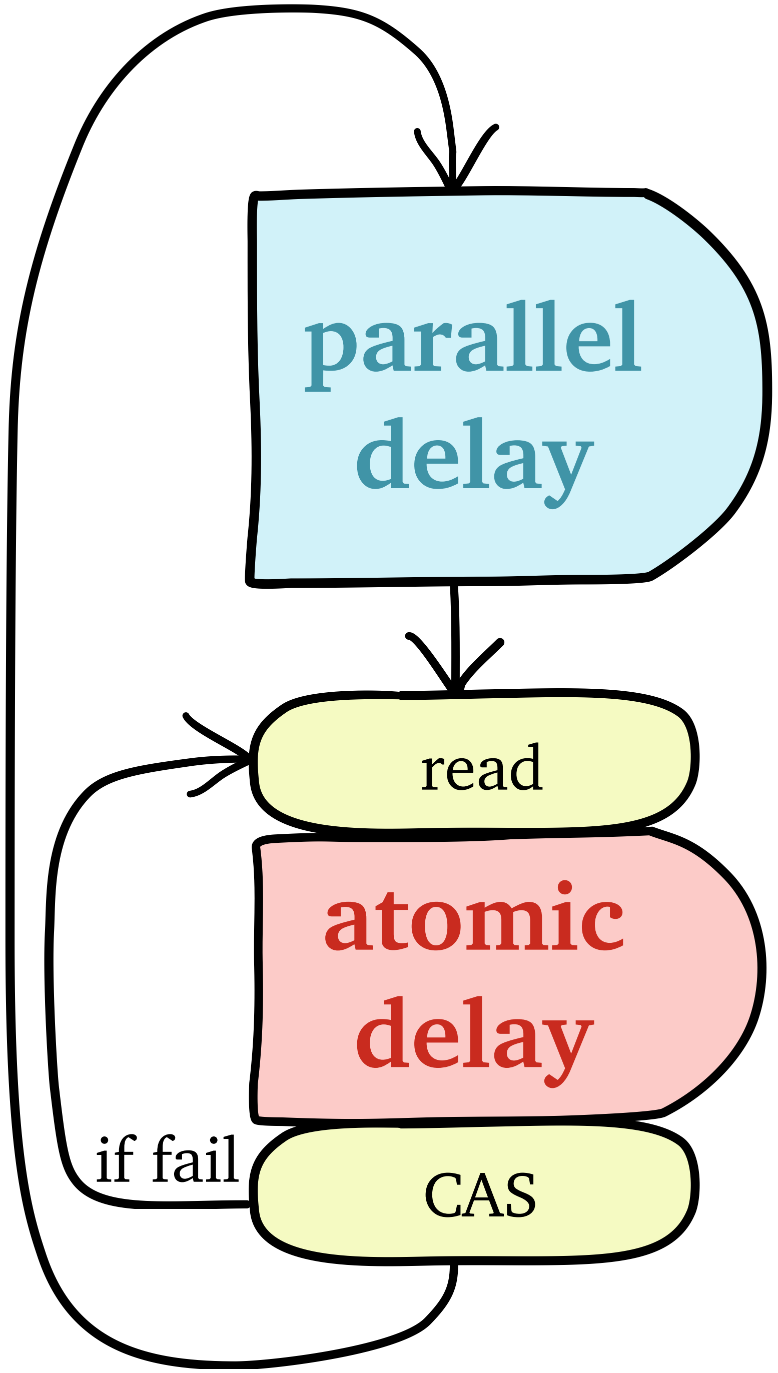
Goal #1: *reveal schedule* of memory accesses

Goal #2: *simulate* lock-free algorithms

F&I Experiments



Read-CAS Experiments

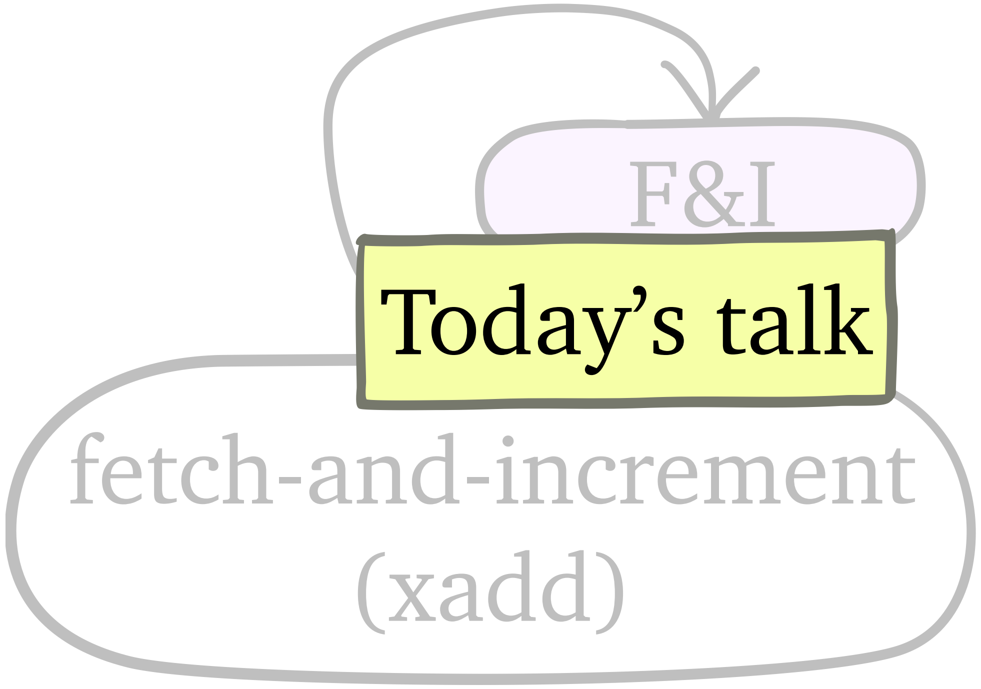


New Tool: Severus

Goal #1: *reveal schedule* of memory accesses

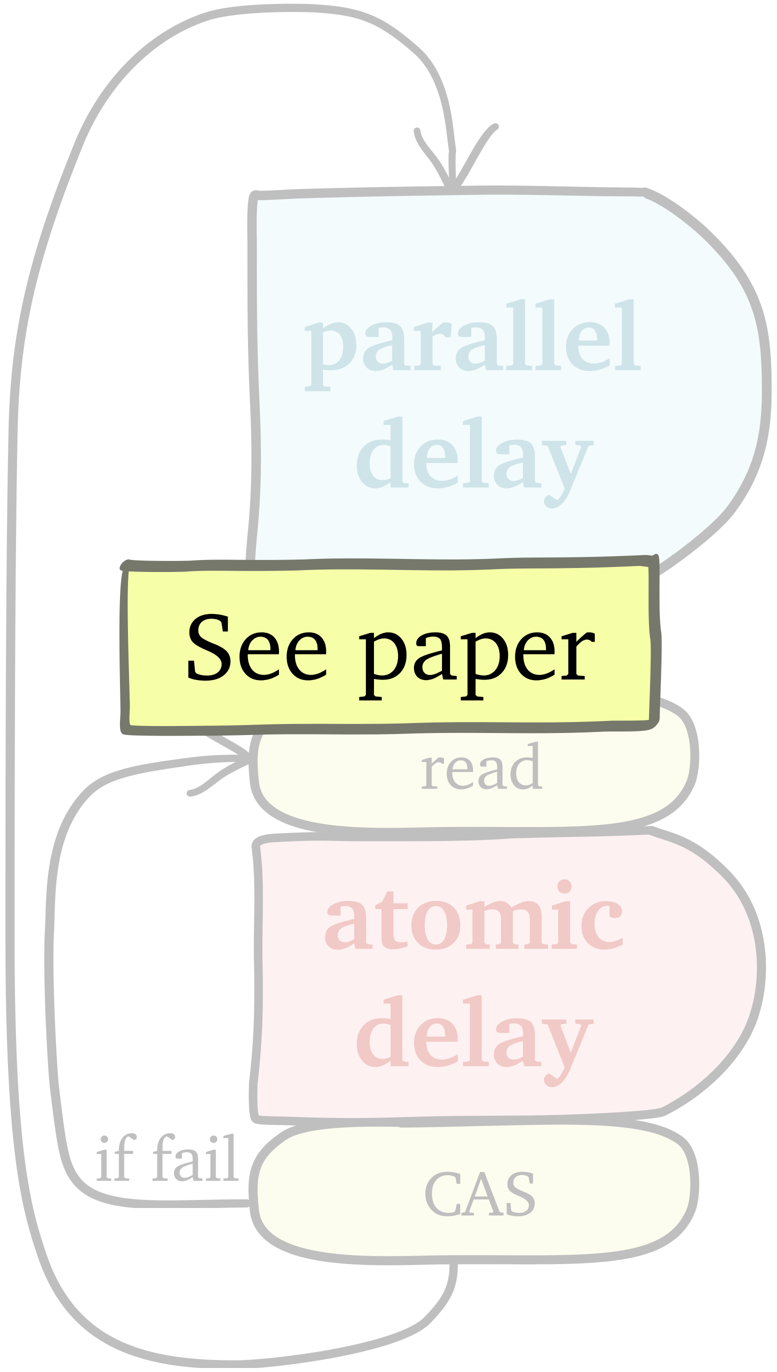
Goal #2: *simulate* lock-free algorithms

F&I Experiments



Today's talk

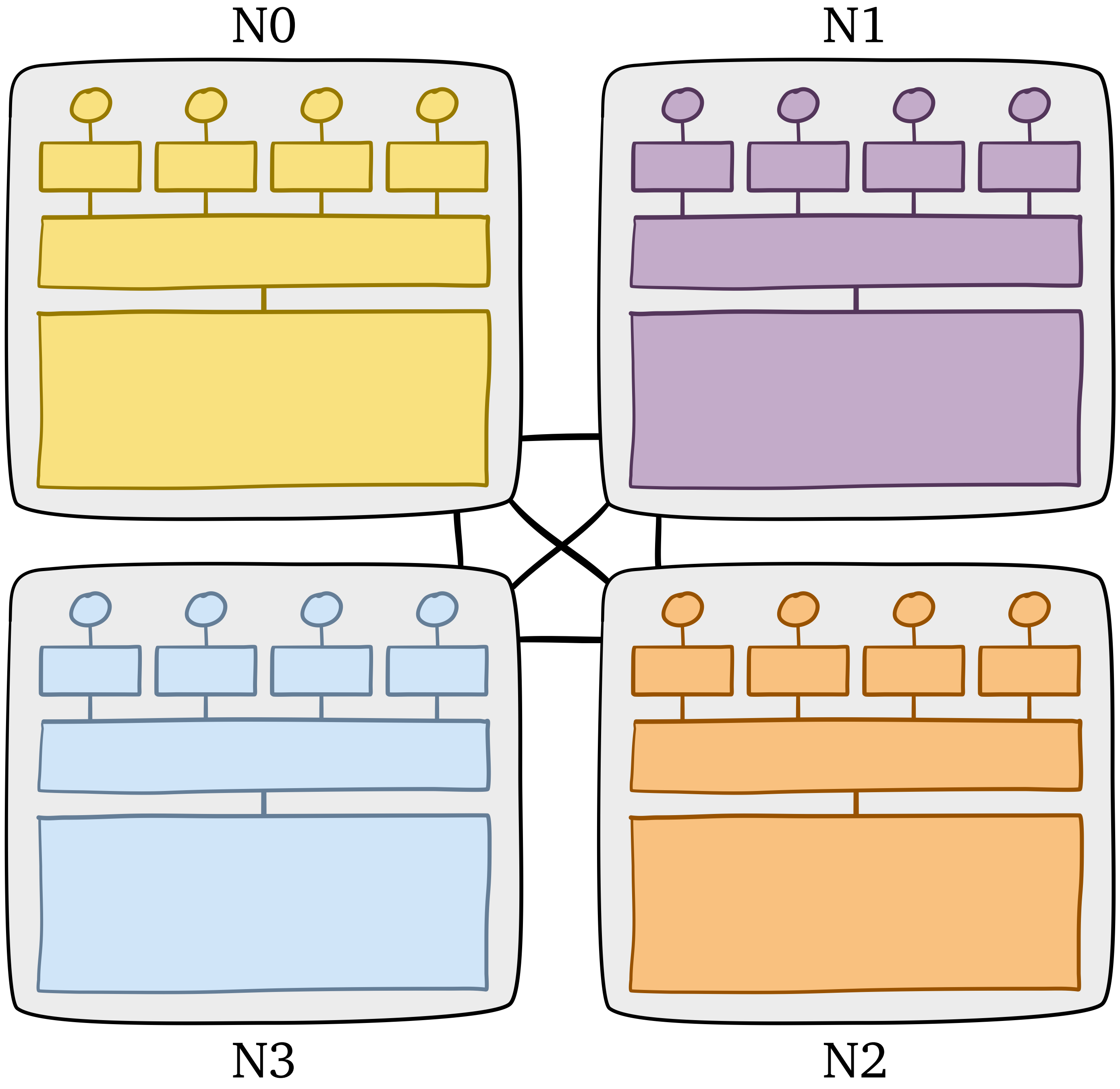
Read-CAS Experiments



See paper

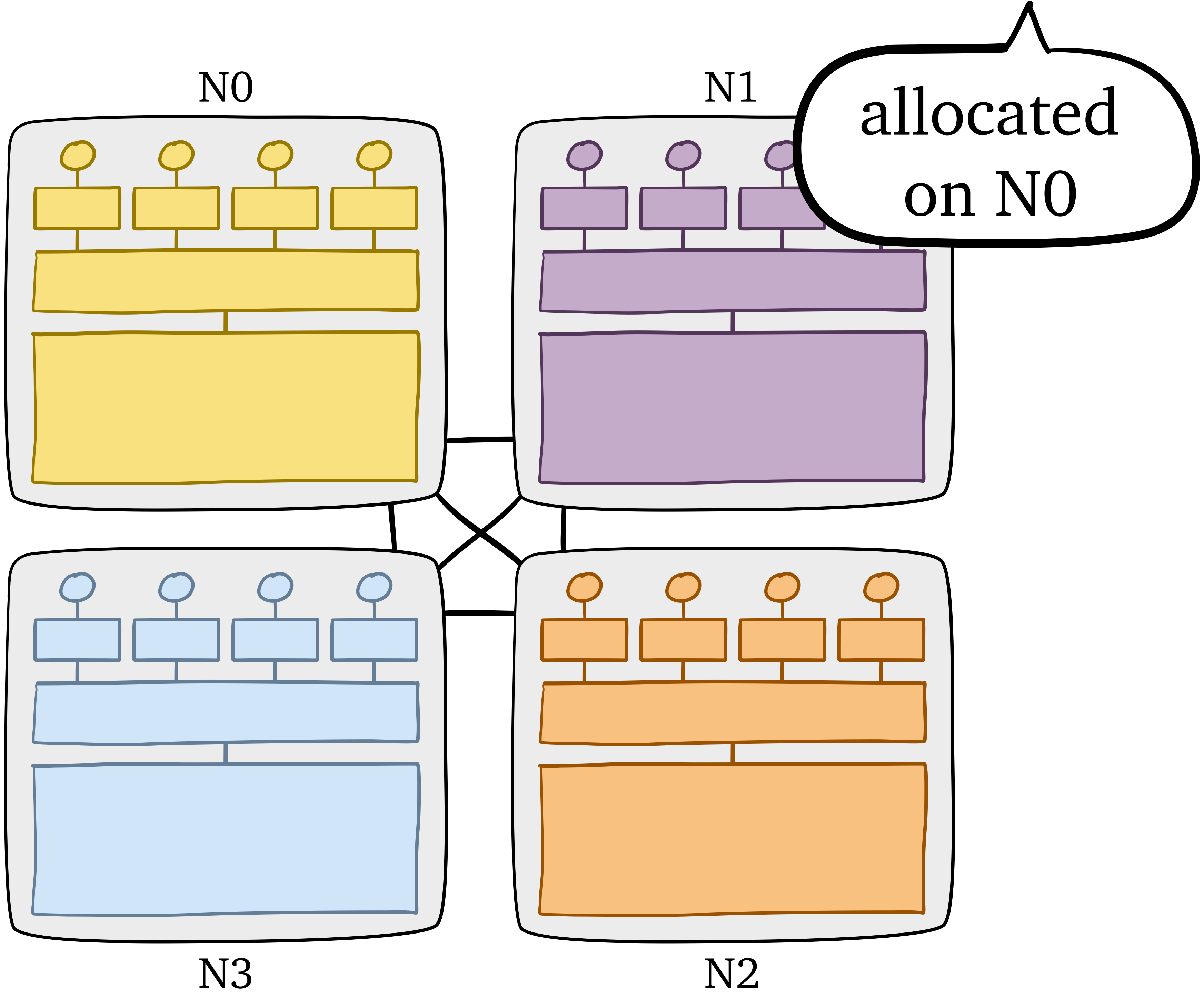
F&I Experiments

All cores F&I same **target** location



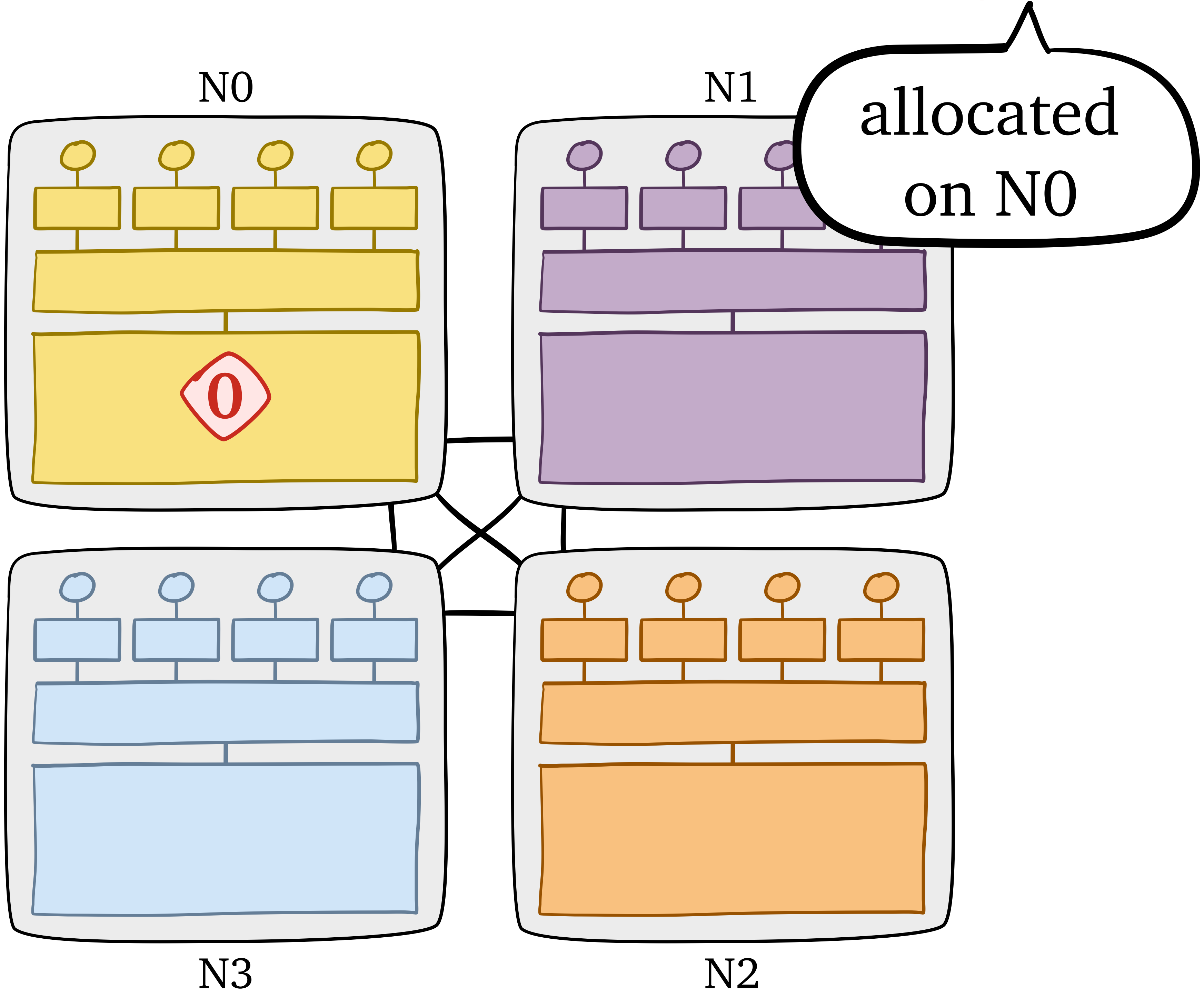
F&I Experiments

All cores F&I same **target** location



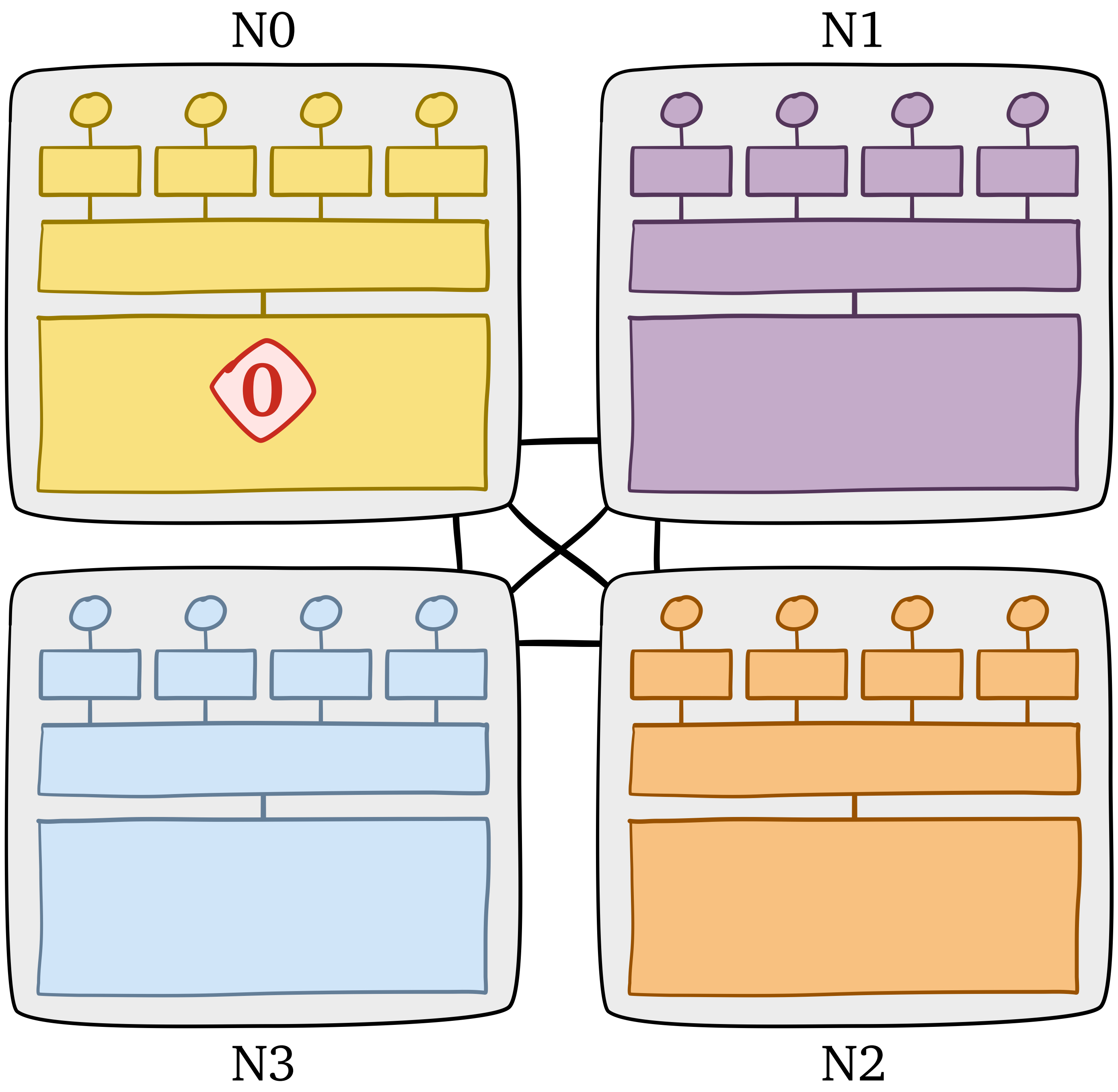
F&I Experiments

All cores F&I same **target** location



F&I Experiments

All cores F&I same **target** location

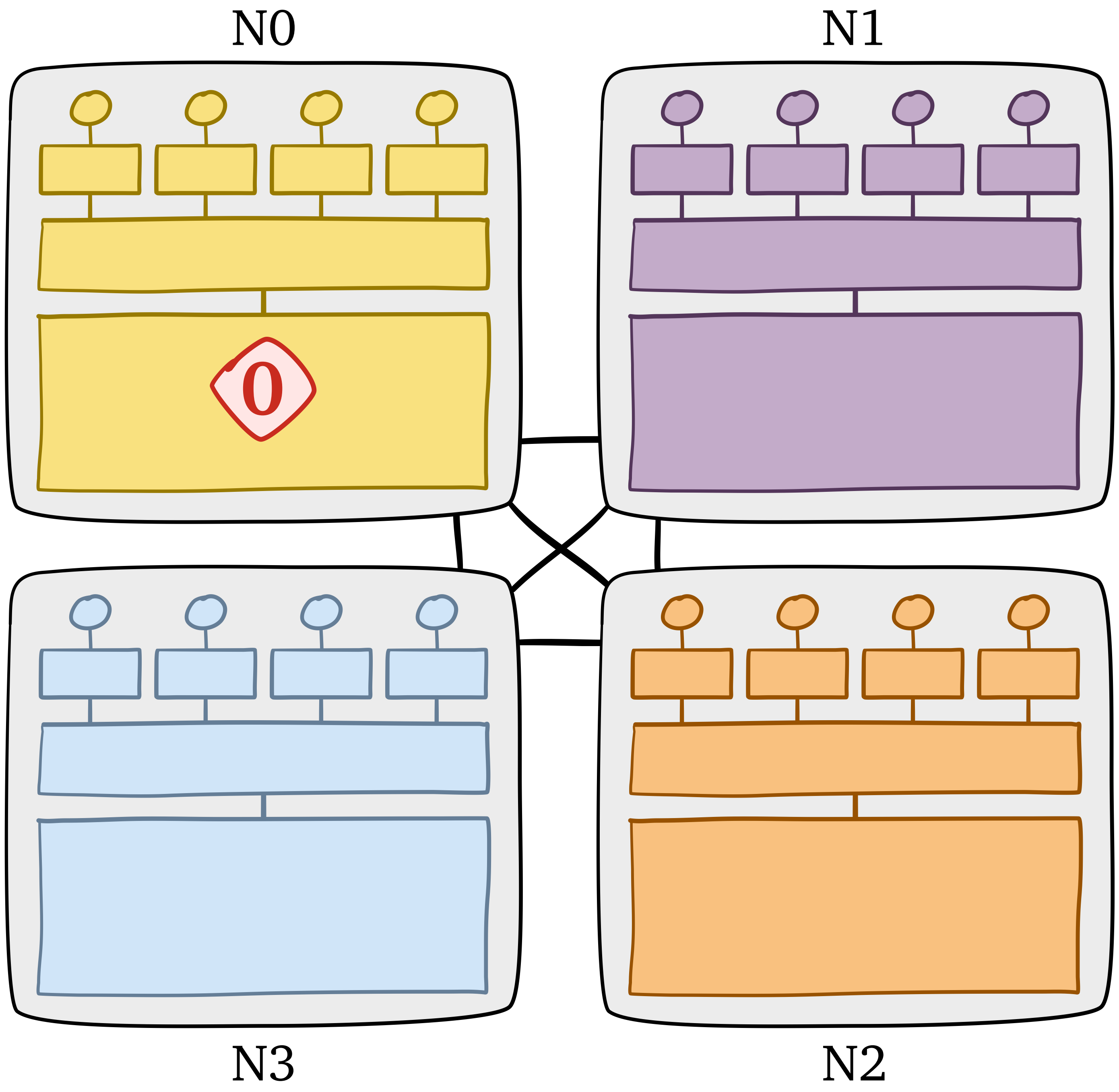


Schedule

thread ID	value

F&I Experiments

All cores F&I same **target** location

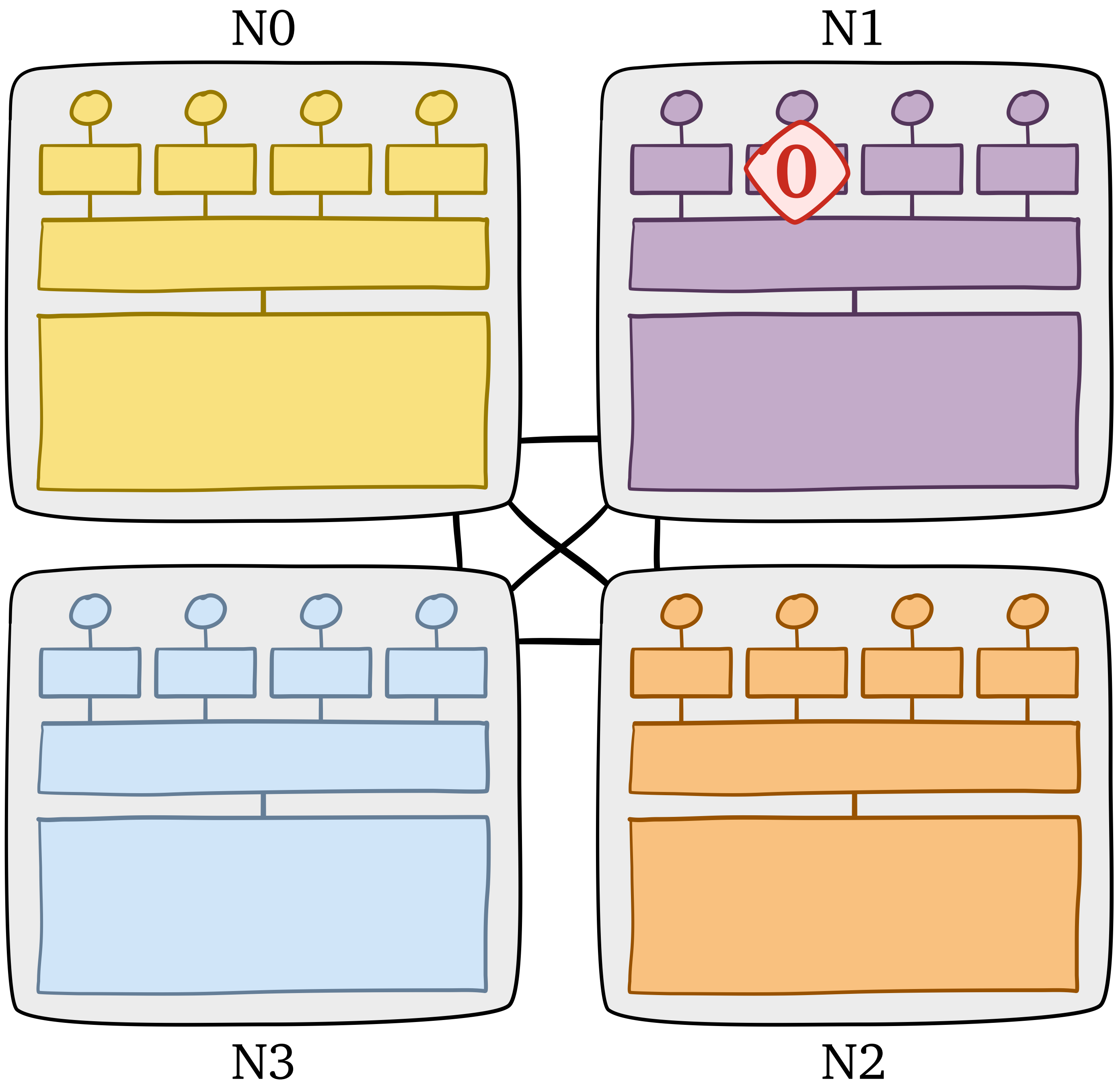


Schedule

thread ID	value
5	

F&I Experiments

All cores F&I same **target** location

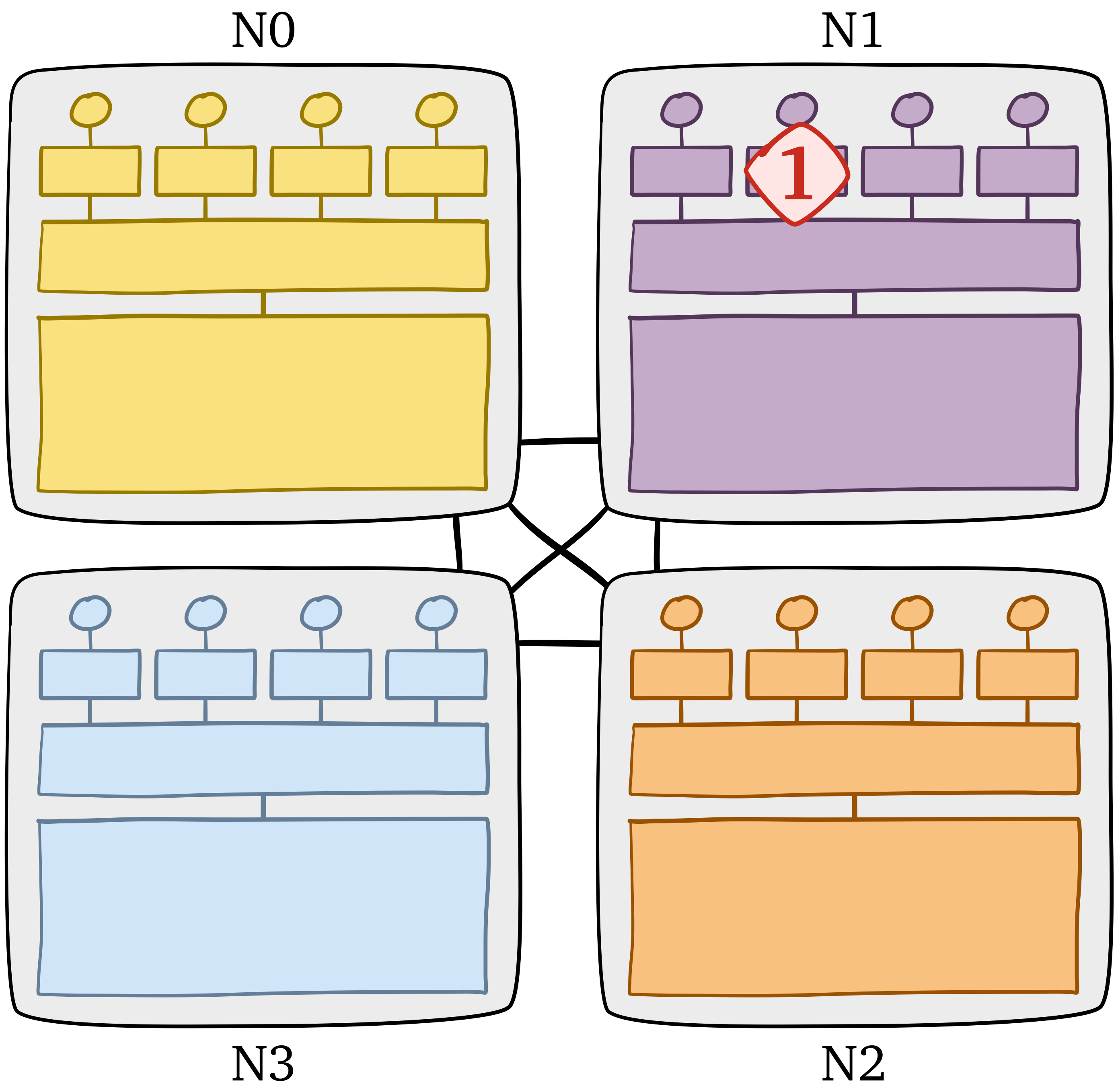


Schedule

thread ID	value
5	

F&I Experiments

All cores F&I same **target** location

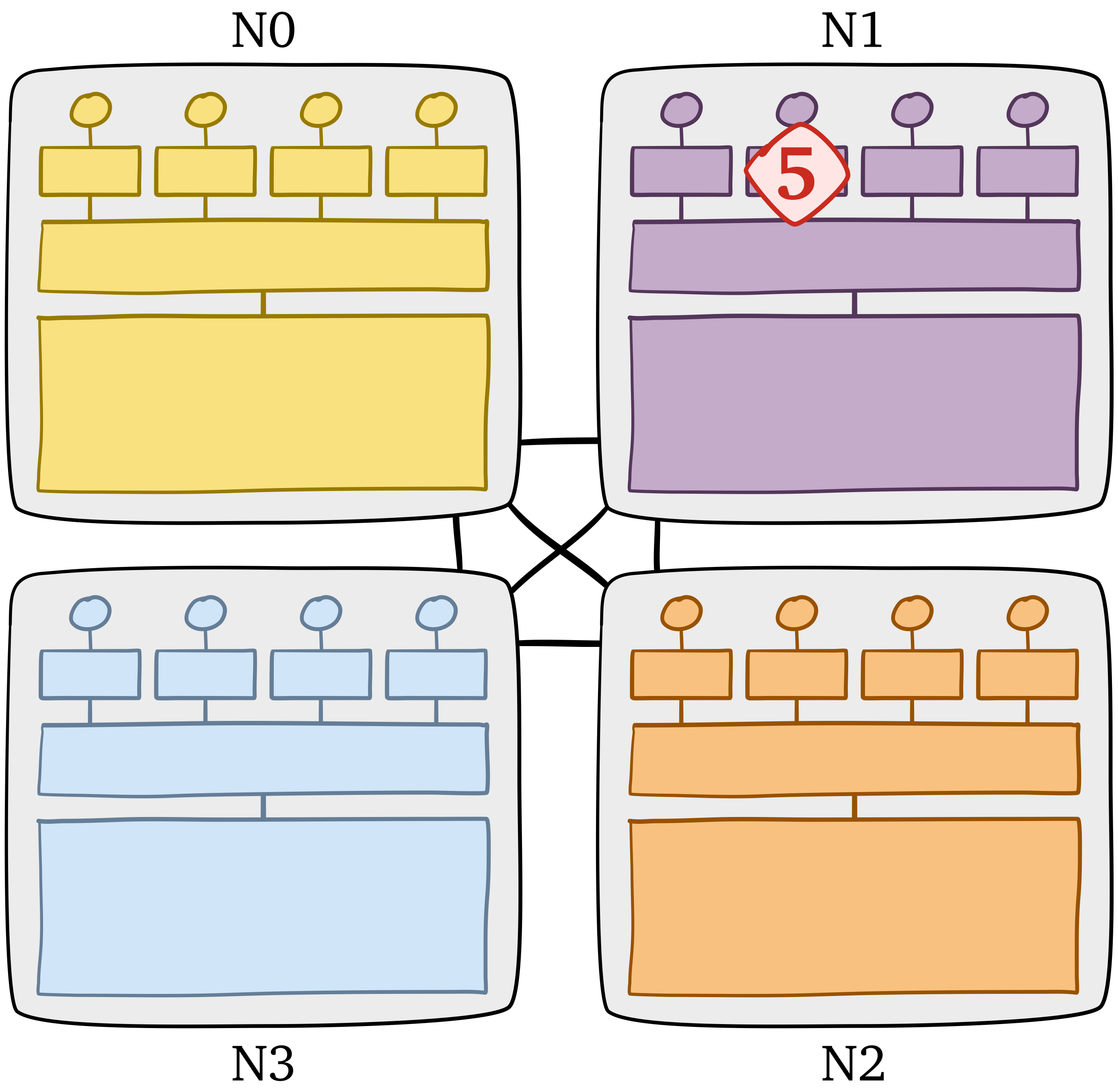


Schedule

thread ID	value
5	**0→1**

F&I Experiments

All cores F&I same **target** location



Schedule

thread ID	value
5	0→1
8	1→2
0	2→3
8	3→4
5	4→5

F&I Experiments

All cores F&I same **target** location

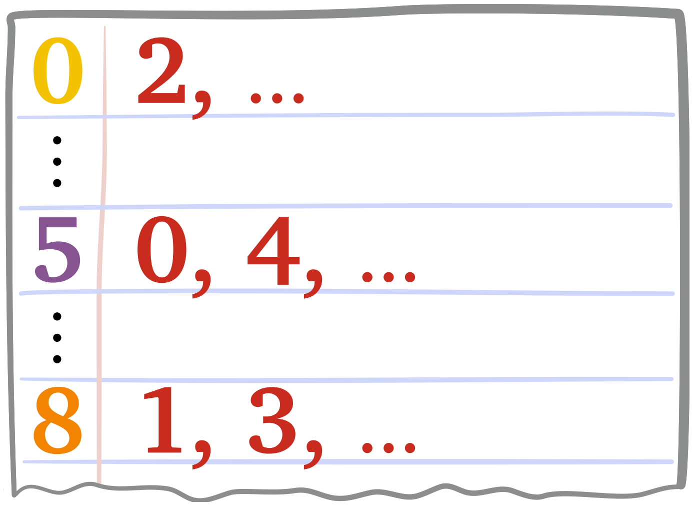
Schedule

thread ID	value
5	0→1
8	1→2
0	2→3
8	3→4
5	4→5

F&I Experiments

All cores F&I same **target** location

Local Logs



0	2, …
⋮	
5	0, 4, …
⋮	
8	1, 3, …

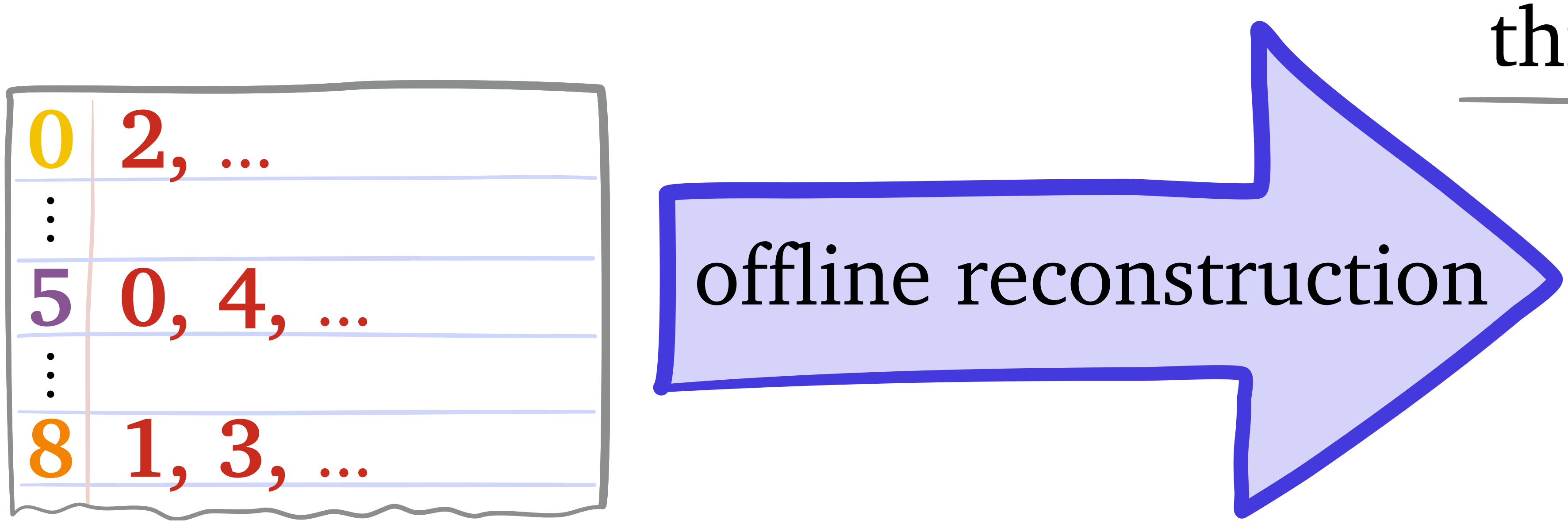
Schedule

thread ID	value
5	0→1
8	1→2
0	2→3
8	3→4
5	4→5

F&I Experiments

All cores F&I same **target** location

Local Logs



0	2, …
⋮	
5	0, 4, …
⋮	
8	1, 3, …

offline reconstruction

Schedule

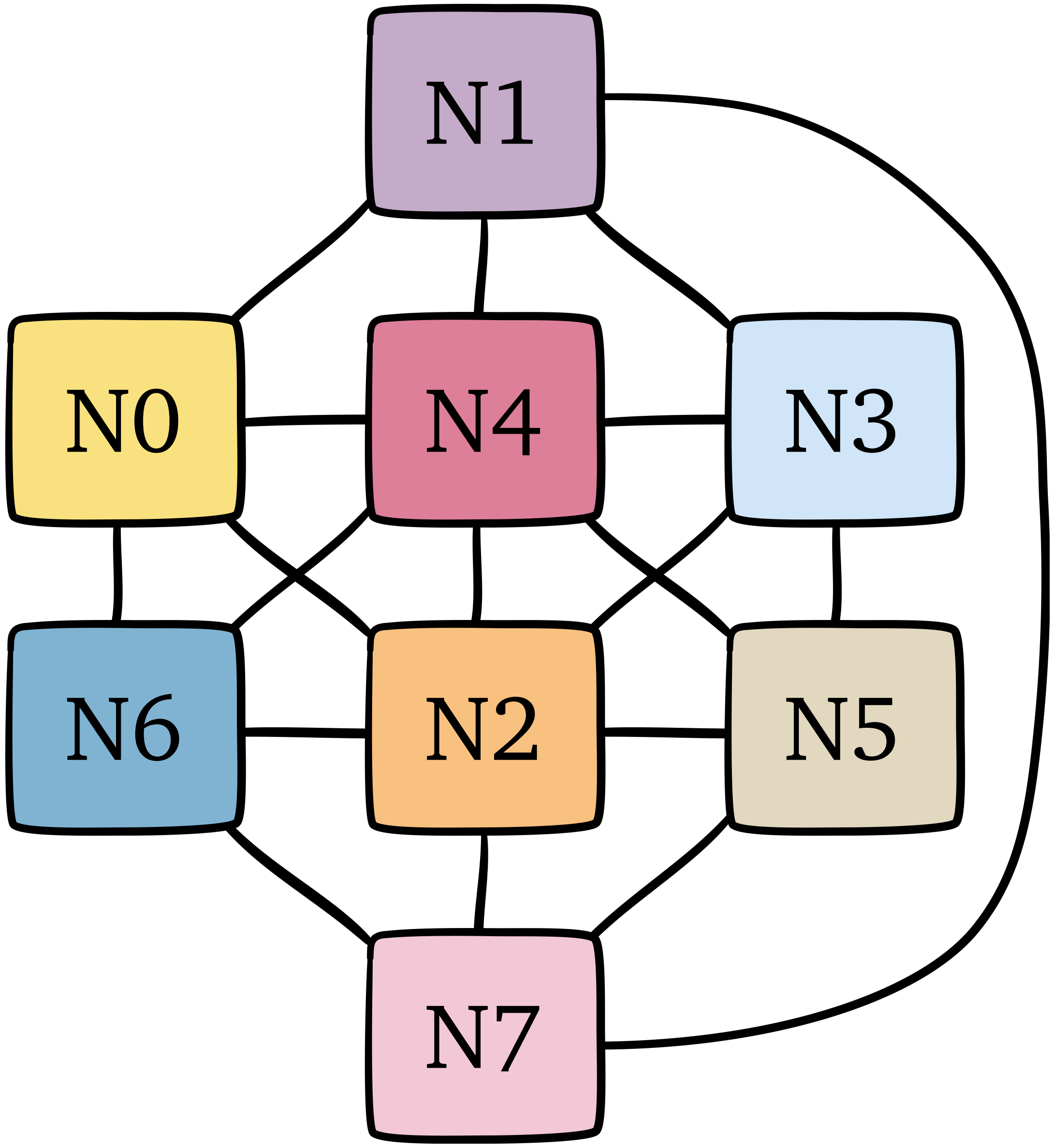
thread ID	value
5	0→1
8	1→2
0	2→3
8	3→4
5	4→5

AMD Interlagos
F&I Experiments

Interlagos Setup

AMD Opteron 6278

8 nodes



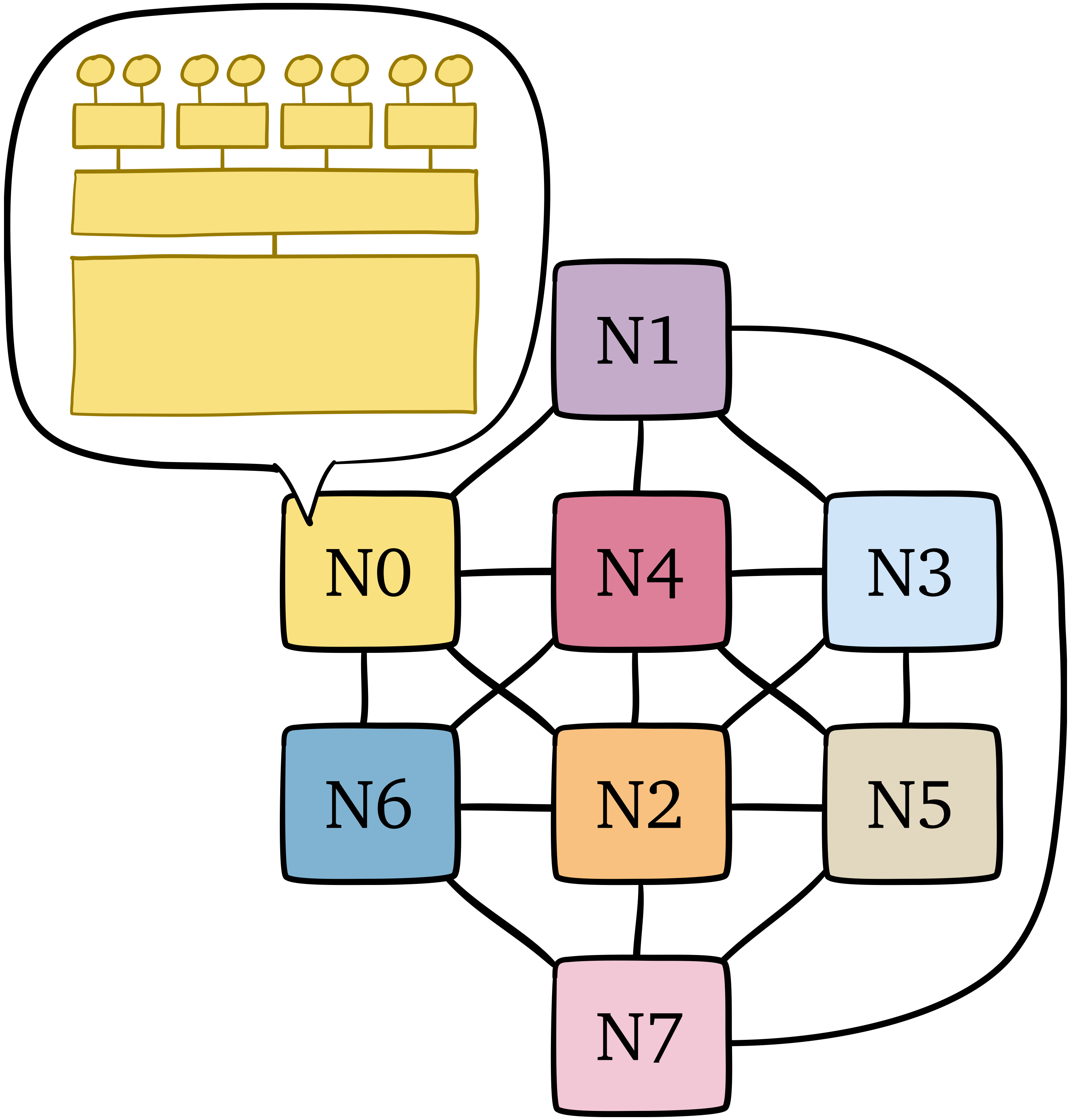
Interlagos Setup

AMD Opteron 6278

8 nodes

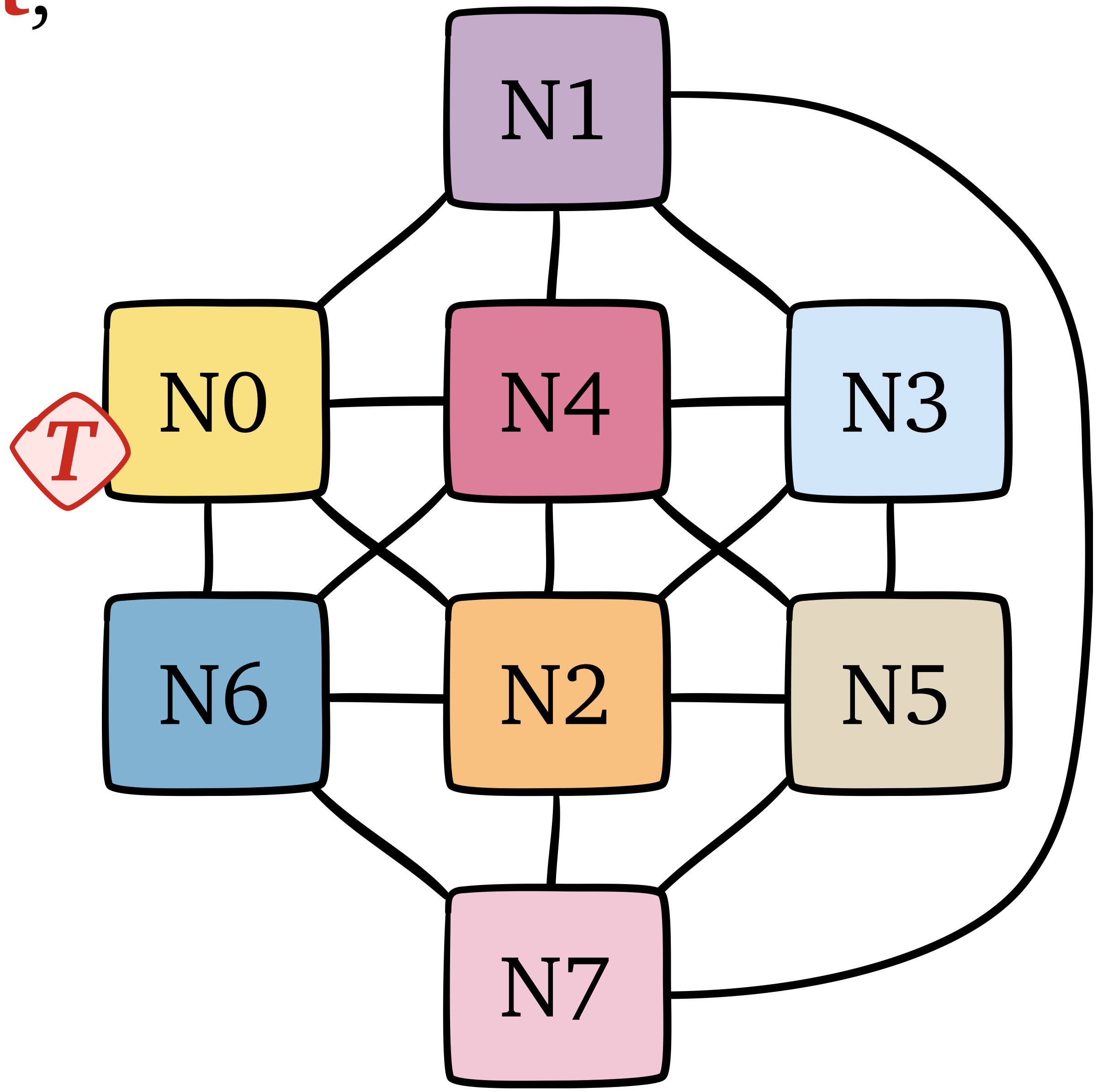
4 modules/node

2 cores/module
(shared L1)



Interlagos Setup

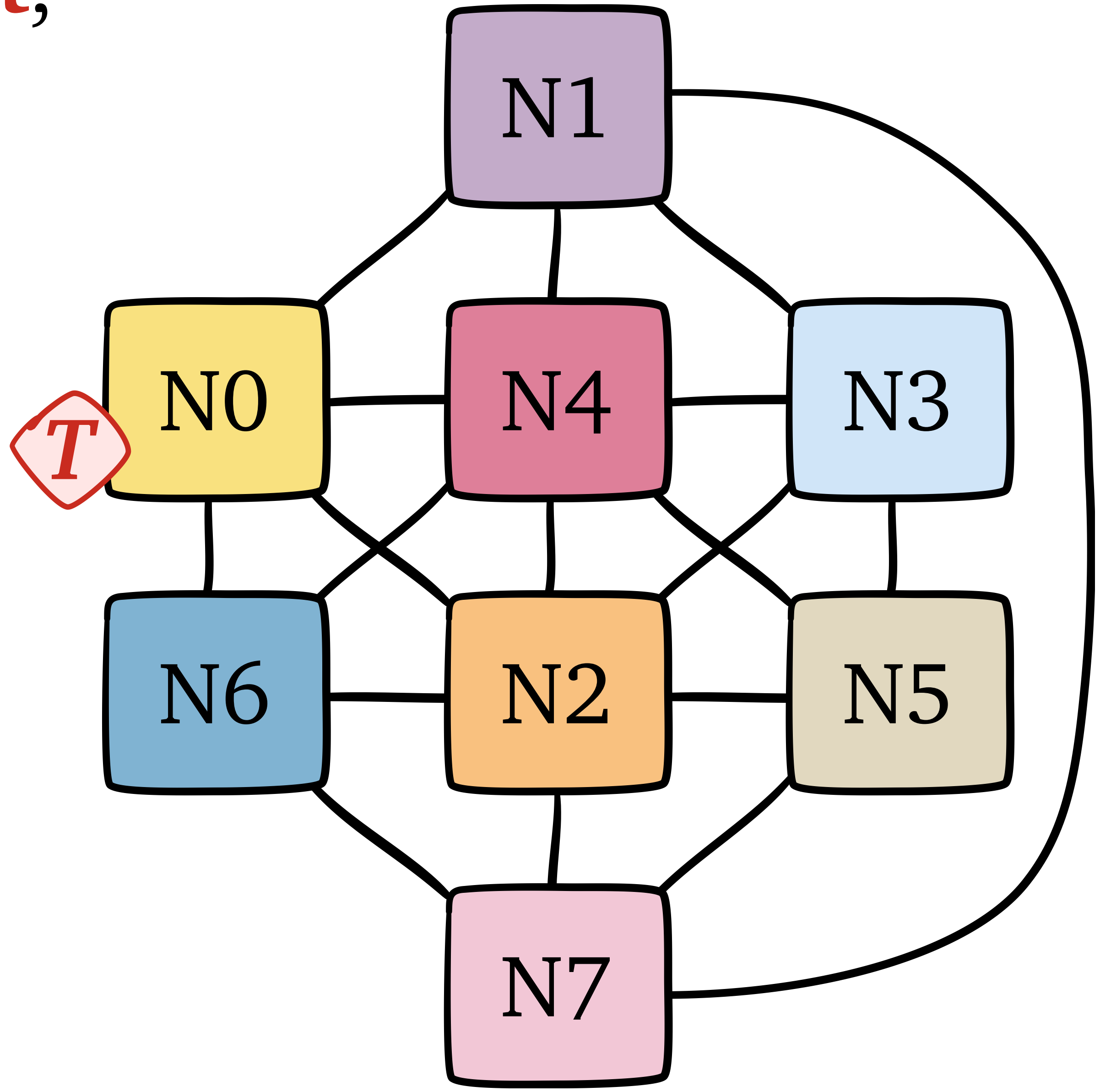
Setup: all cores F&I **target**, which is allocated on N0



Interlagos Setup

Setup: all cores F&I **target**, which is allocated on N0

Question: which nodes' cores do most F&I/sec?

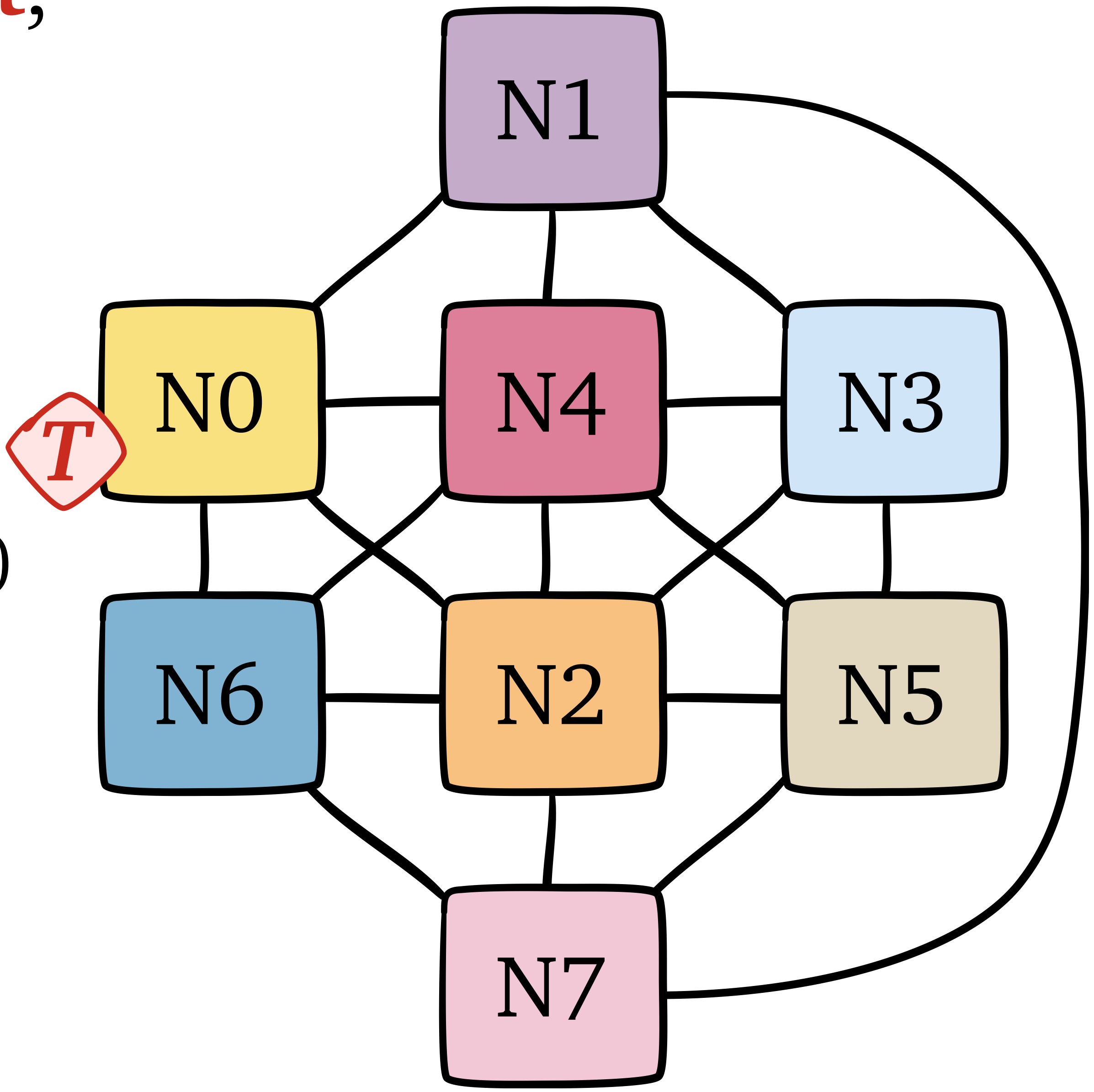


Interlagos Setup

Setup: all cores F&I **target**, which is allocated on N0

Question: which nodes' cores do most F&I/sec?

A. distance 0 (N0)
B. distance 1 (N1, N2, N4, N6)
C. distance 2 (N3, N5, N7)
D. all equal
E. something else



Interlagos Setup

Setup: all cores F&I **target**, which is allocated on N0

Question: which nodes' cores do most F&I/sec?

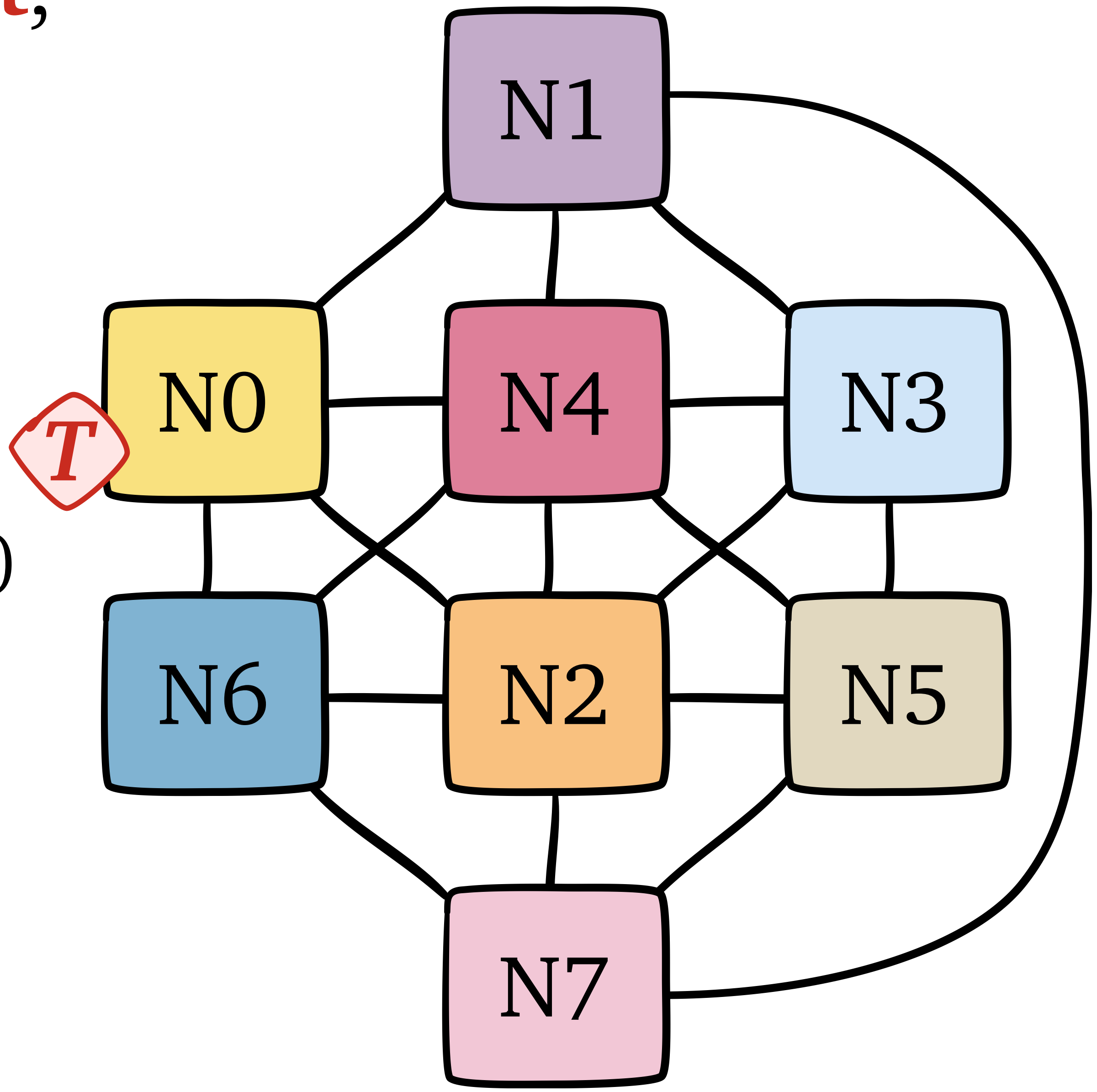
A. distance 0 (N0)

B. distance 1 (N1, N2, N4, N6)

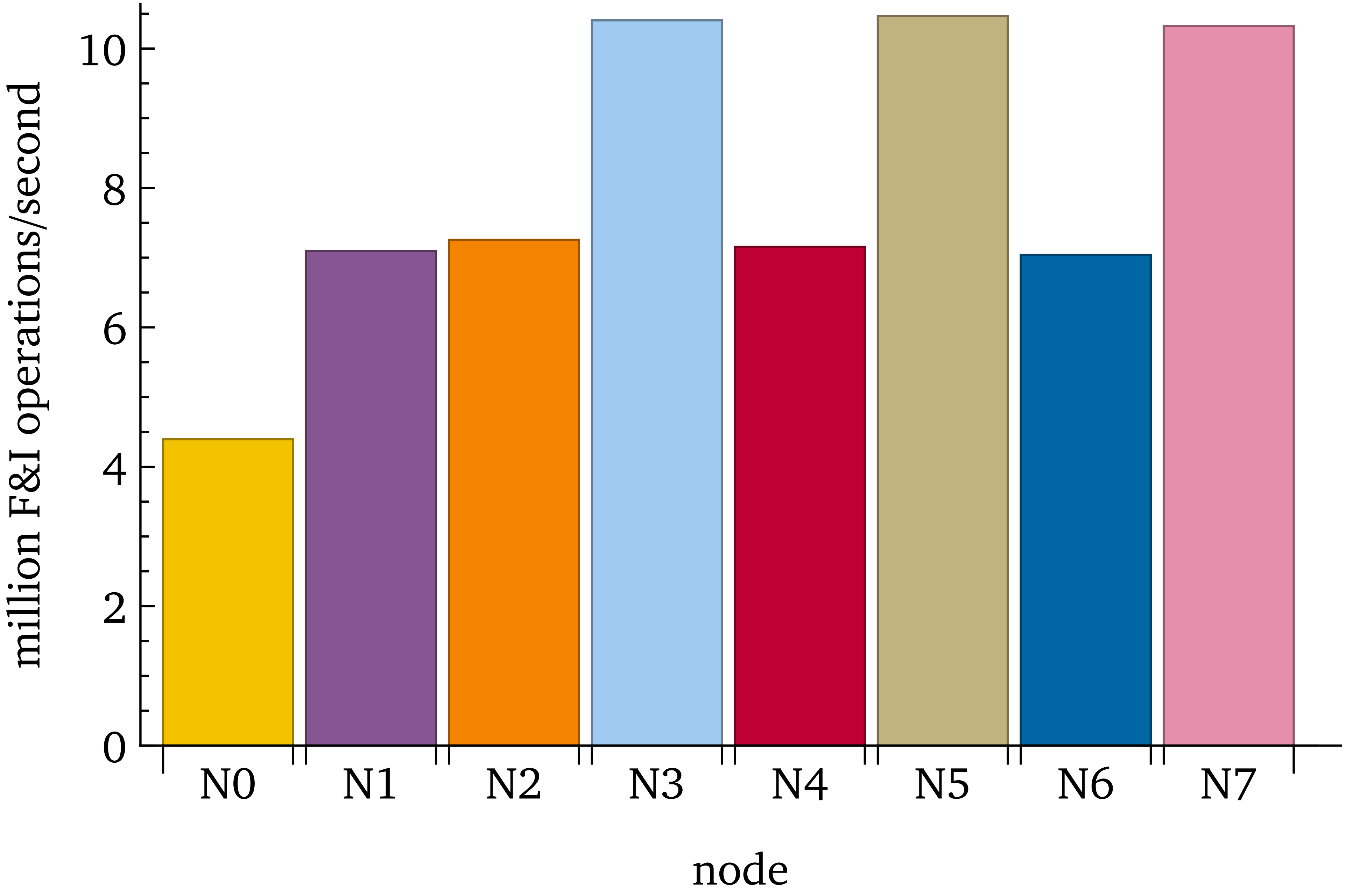
[✓ C. distance 2 (N3, N5, N7)]

D. all equal

E. something else

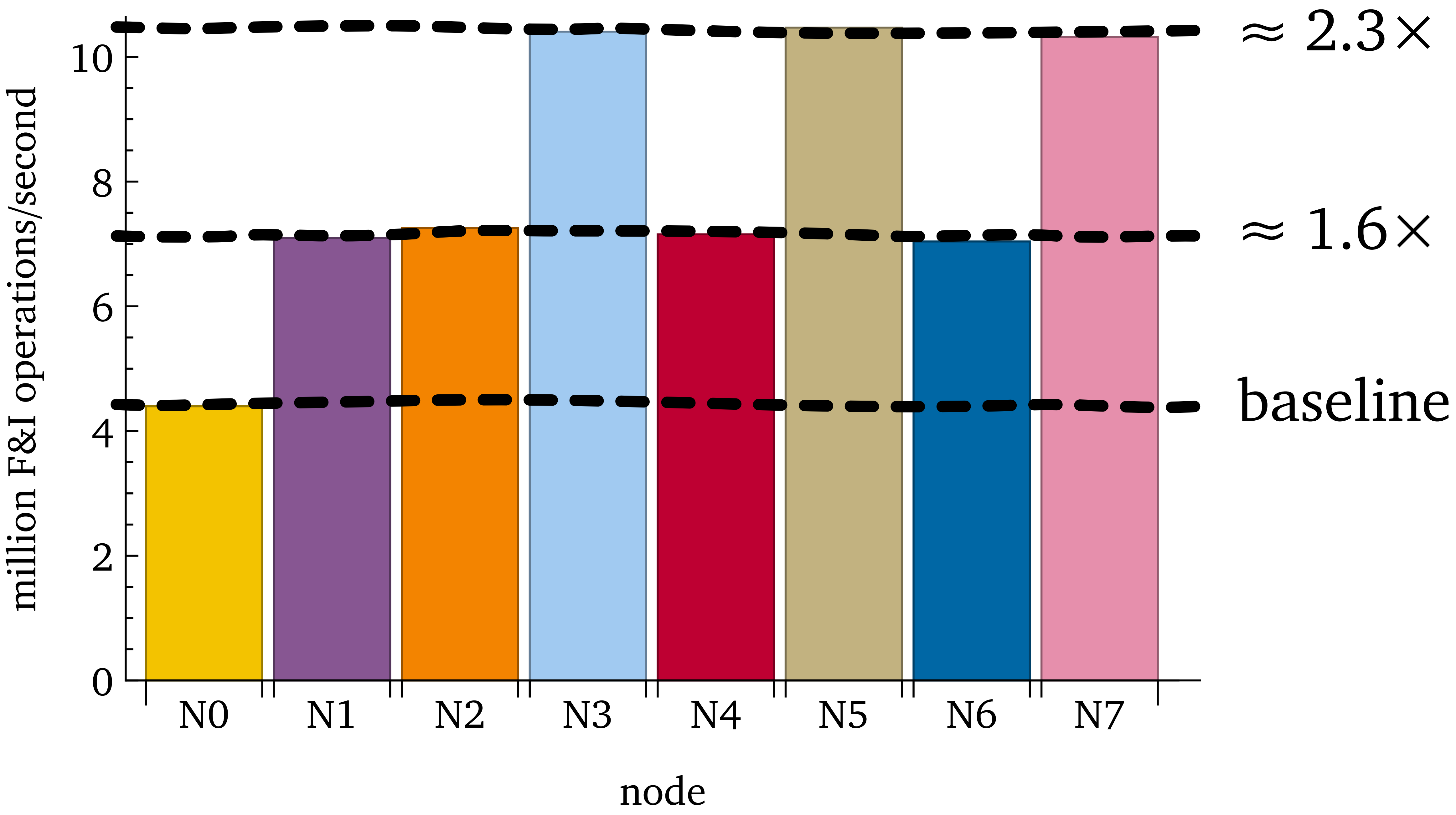


Interlagos F&I Throughput



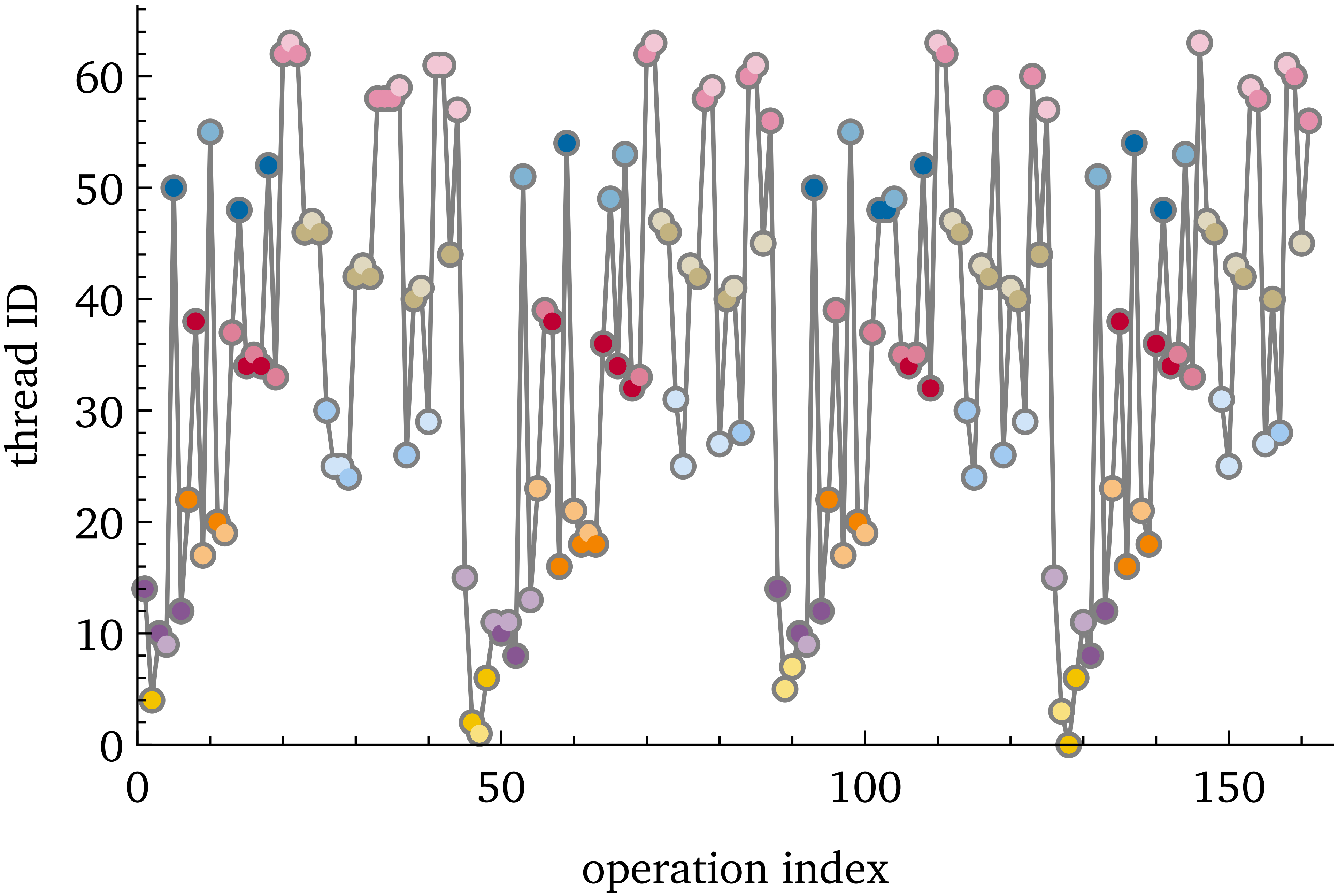
Setup: all nodes running, target on N0

Interlagos **F&I** Throughput



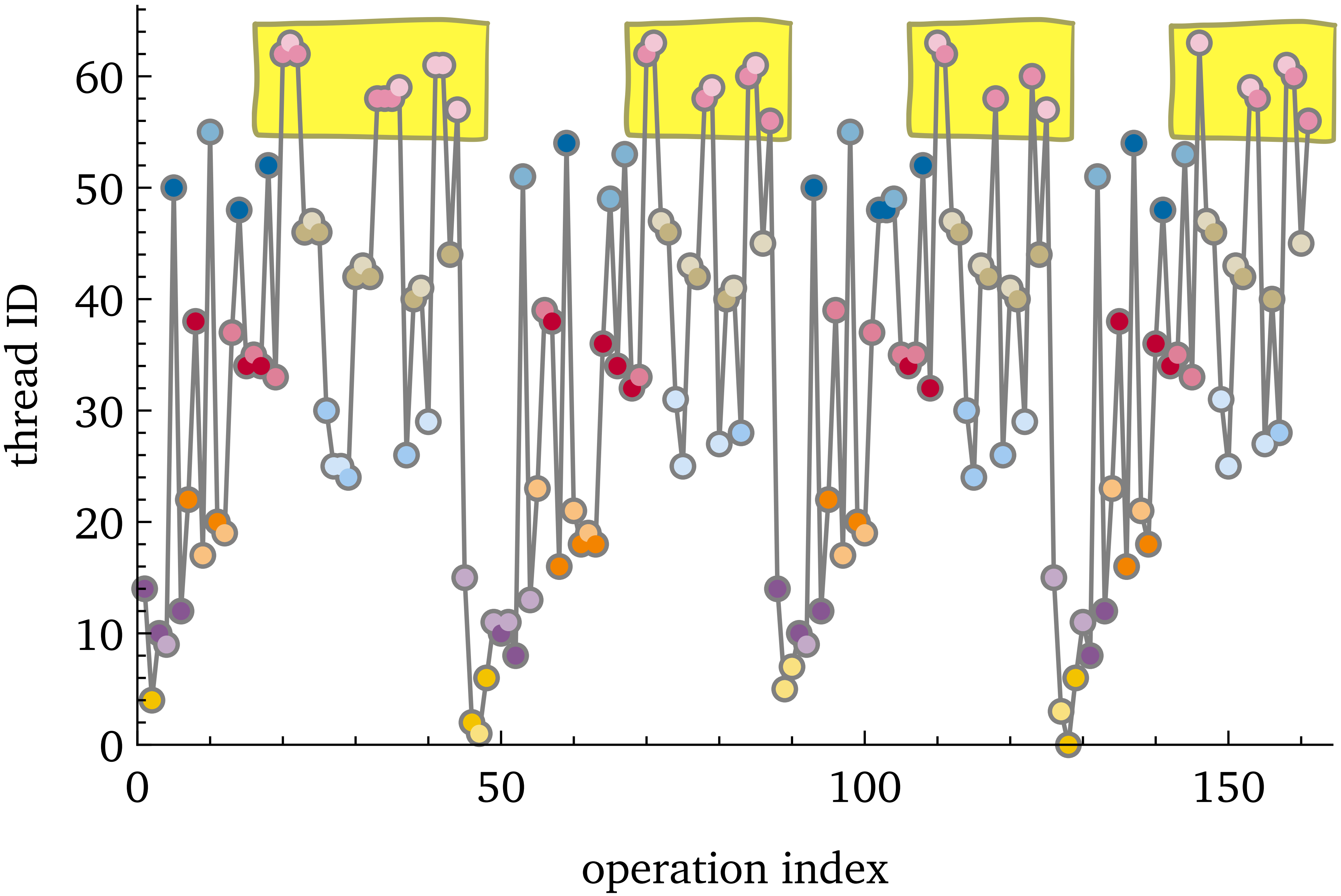
Setup: all nodes running, target on N0

Interlagos F&I Schedule



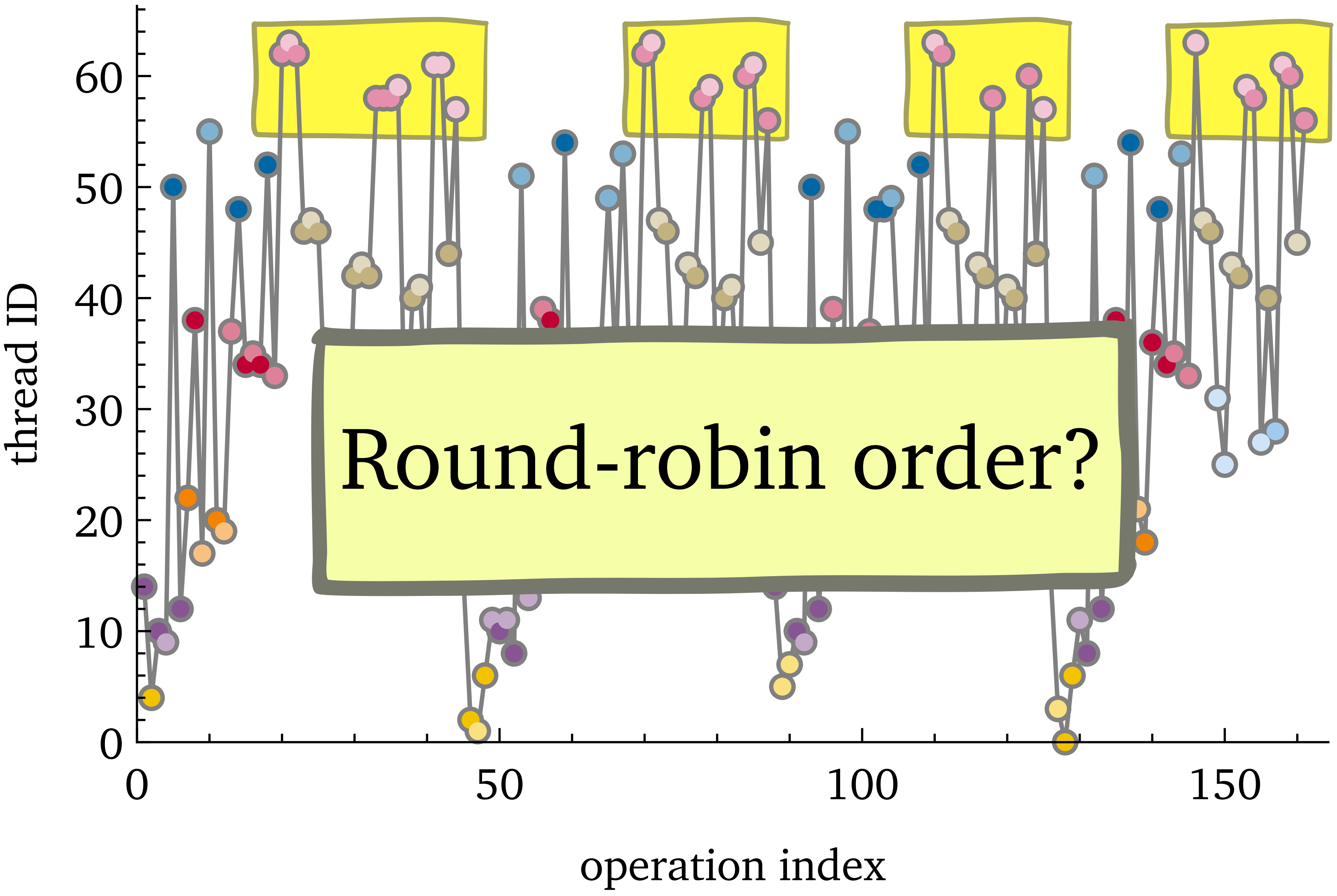
Setup: all nodes running, target on N0

Interlagos **F&I** Schedule



Setup: all nodes running, target on N0

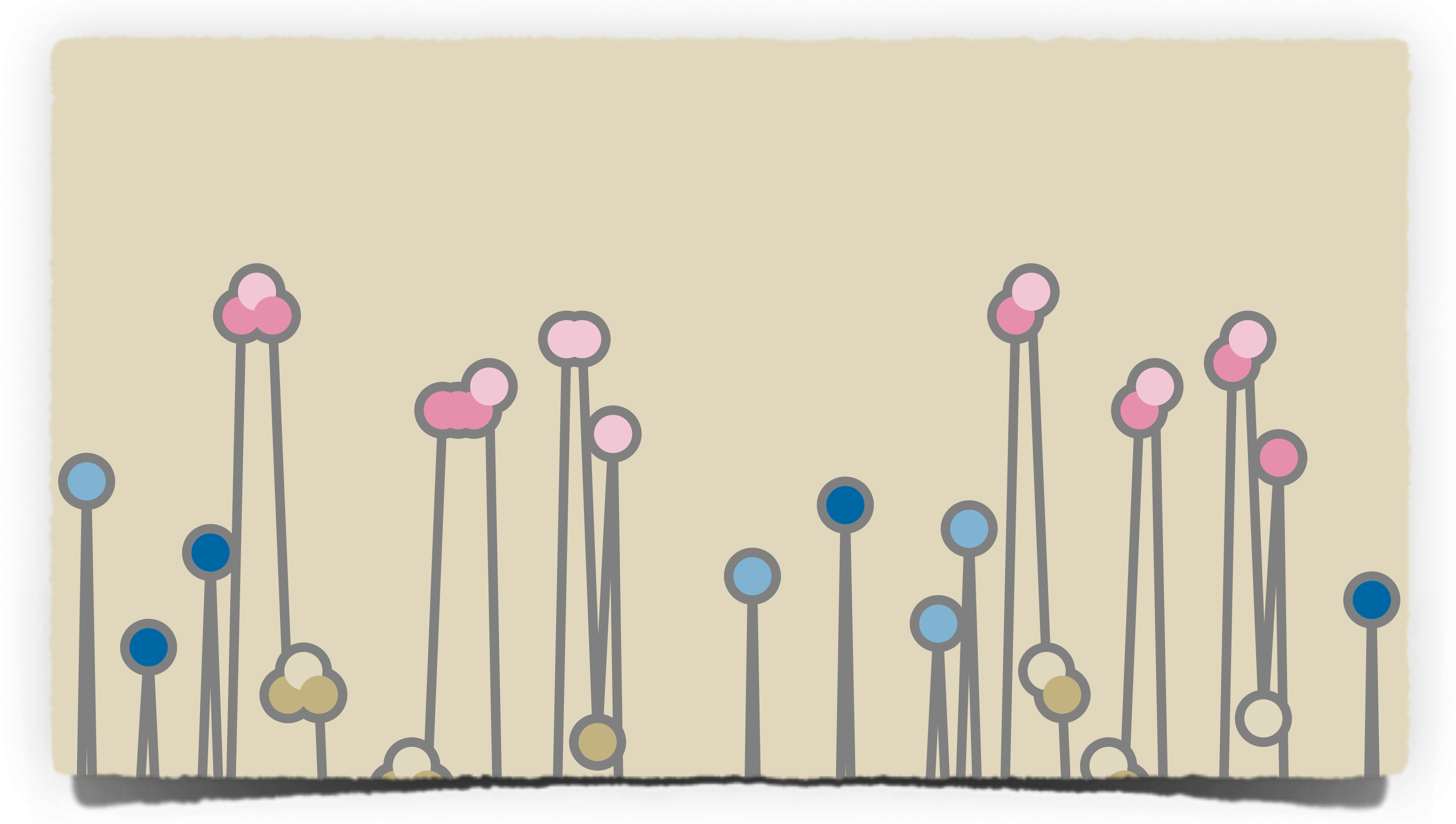
Interlagos F&I Schedule



Round-robin order?

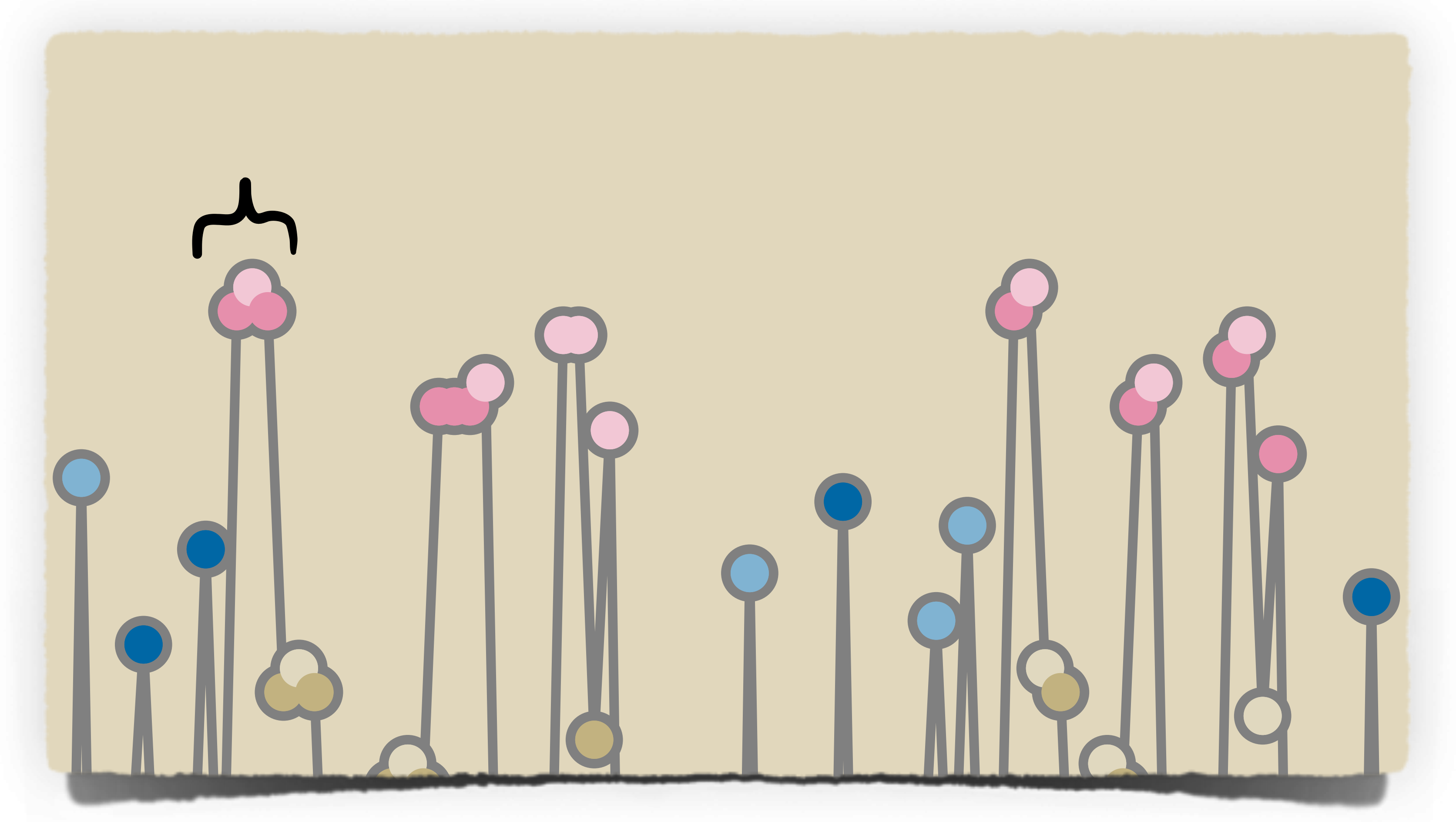
Setup: all nodes running, target on N0

Module Visits



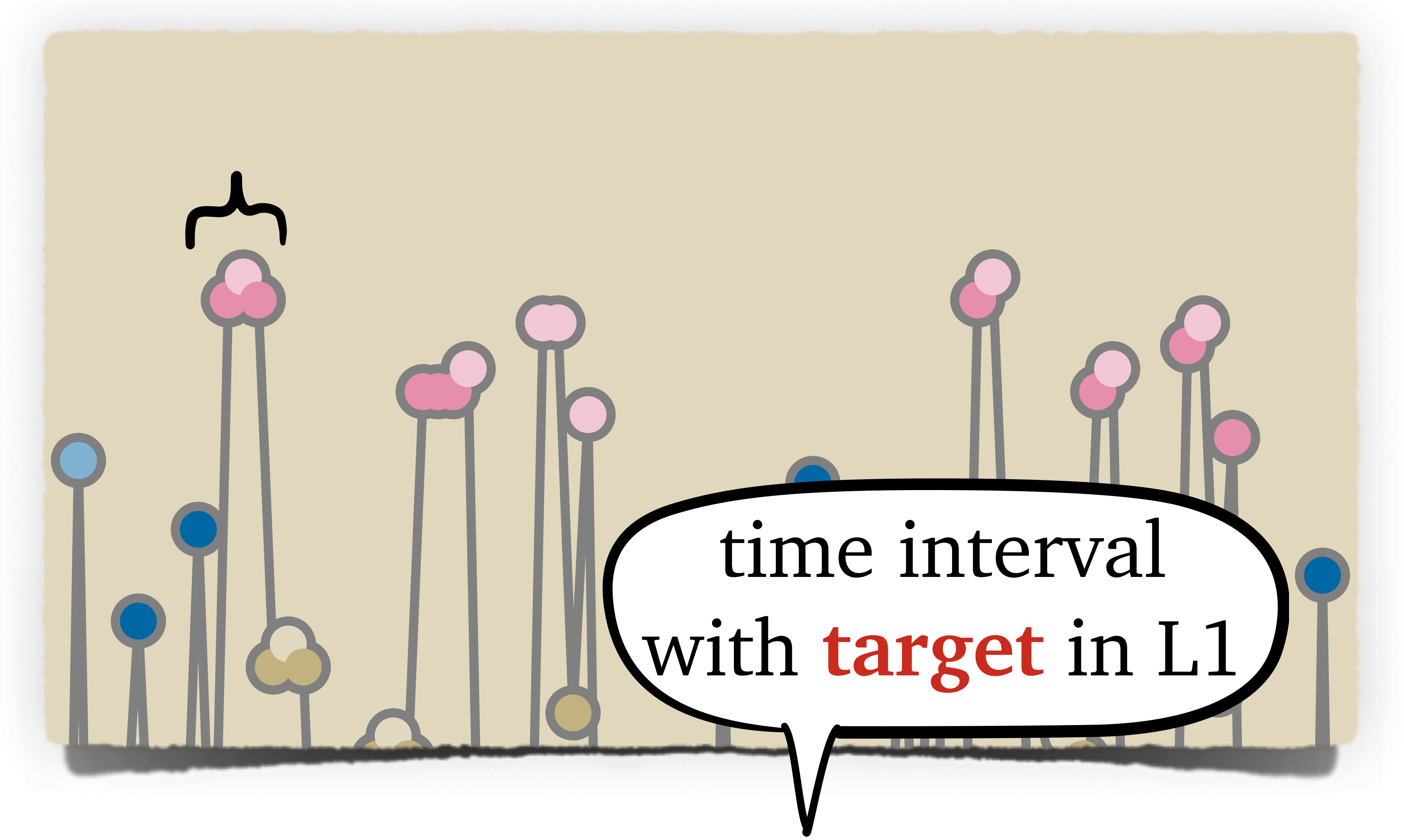
Module visit: consecutive F&Is by cores in same module

Module Visits



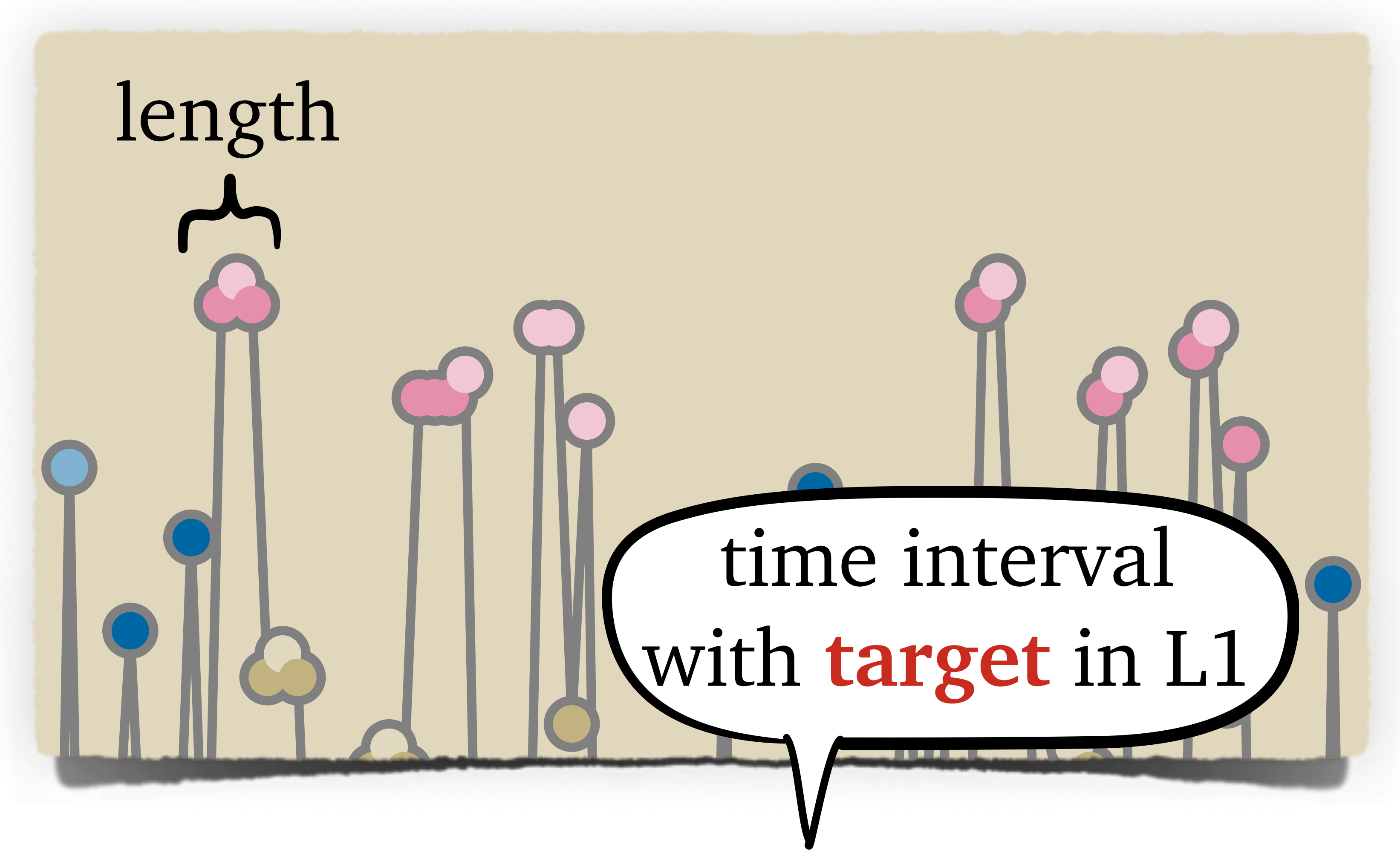
Module visit: consecutive F&Is by cores in same module

Module Visits



Module visit: consecutive F&Is by cores in same module

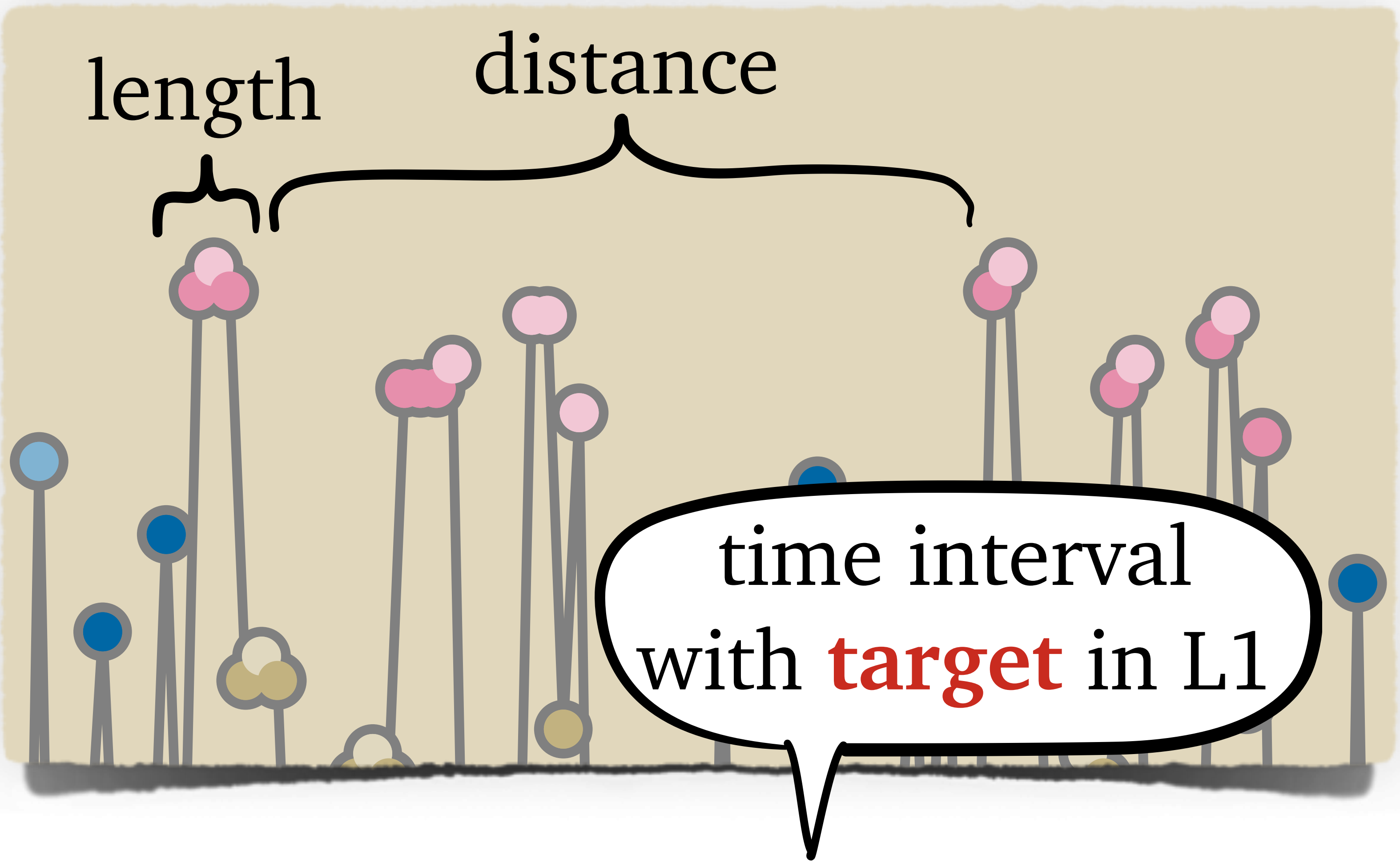
Module Visits



Module visit: consecutive F&Is by cores in same module

Visit length: number of F&Is in a visit

Module Visits



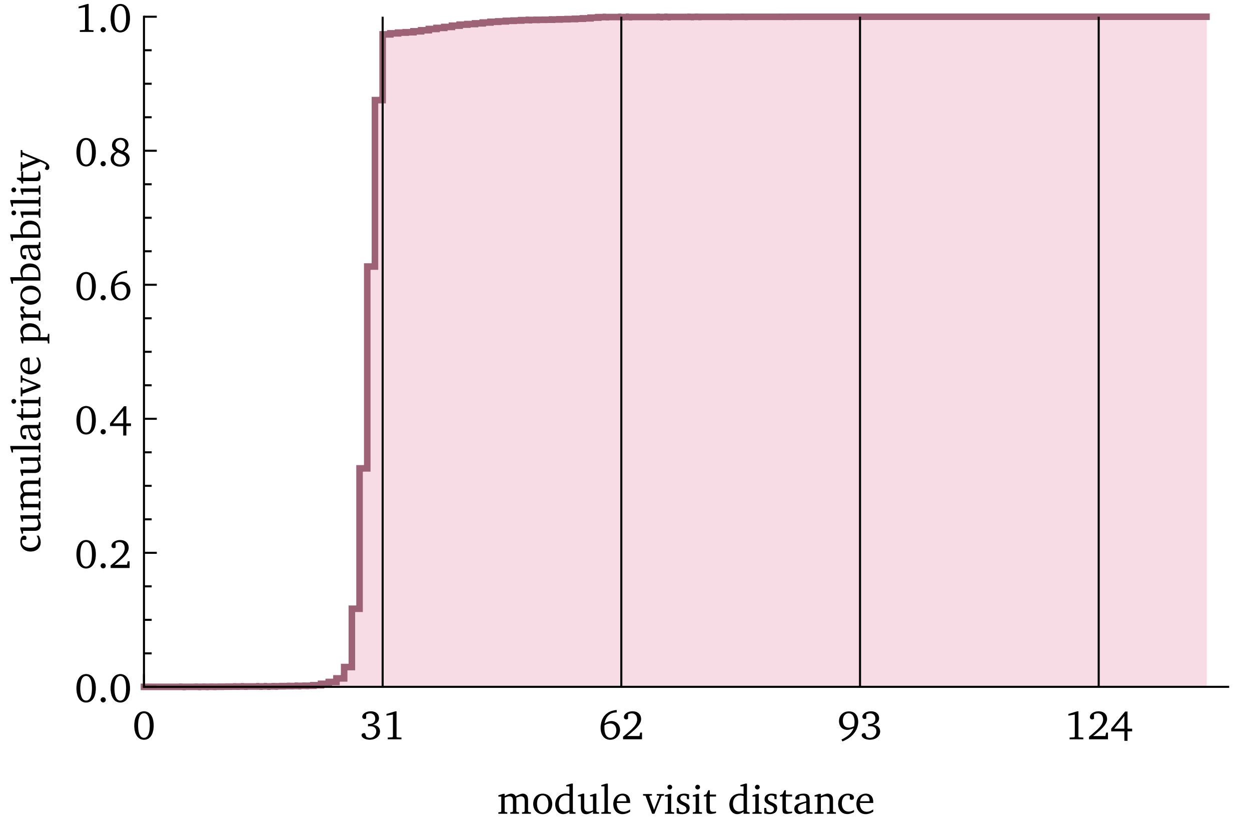
Module visit: consecutive F&Is by cores in same module

Visit length: number of F&Is in a visit

Visit distance: number of other visits between two visits to the same module

Interlagos F&I Visit Distances

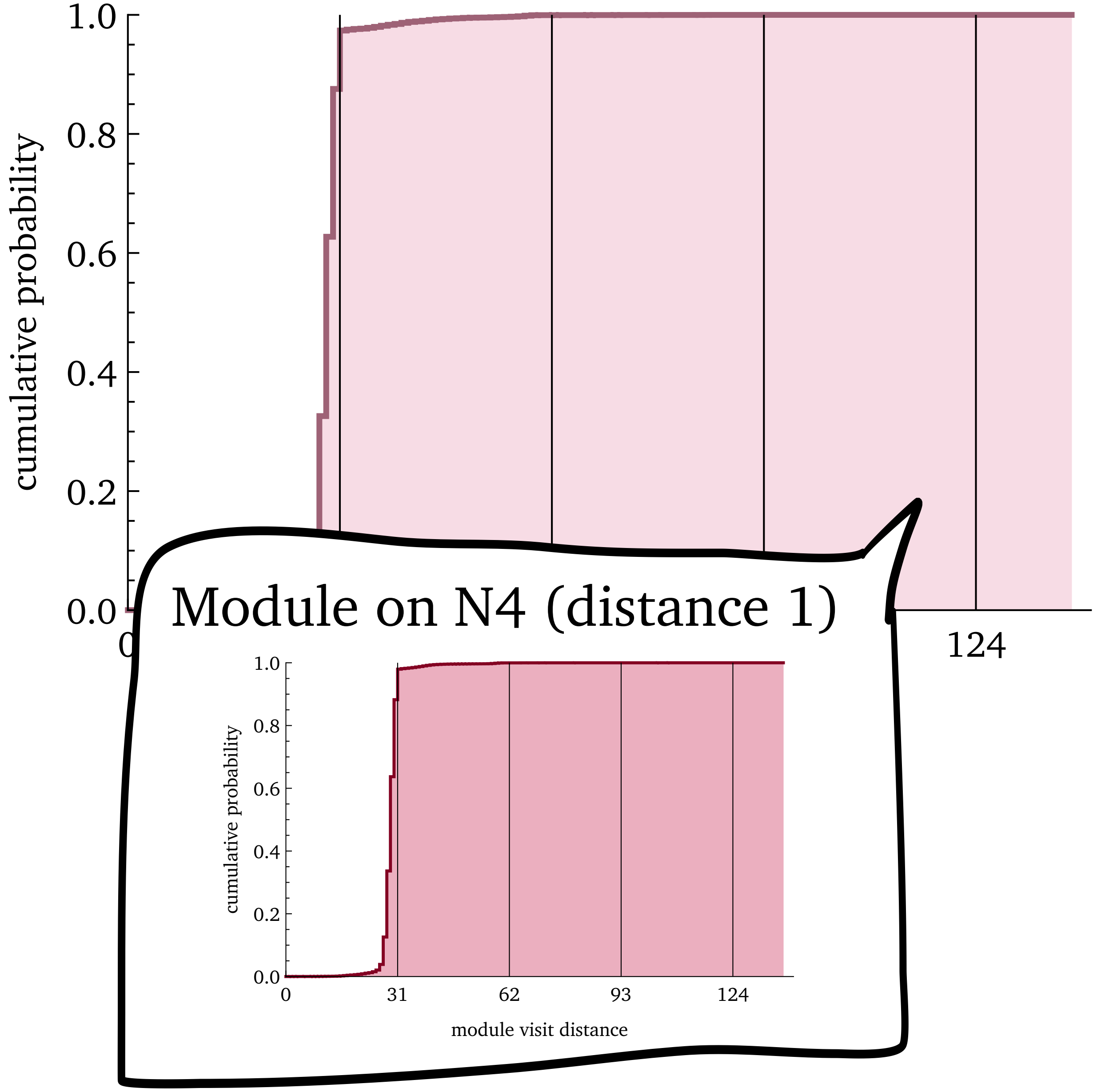
Module on N7 (distance 2)



Setup: all nodes running, target on N0

Interlagos **F&I** Visit Distances

Module on N7 (distance 2)

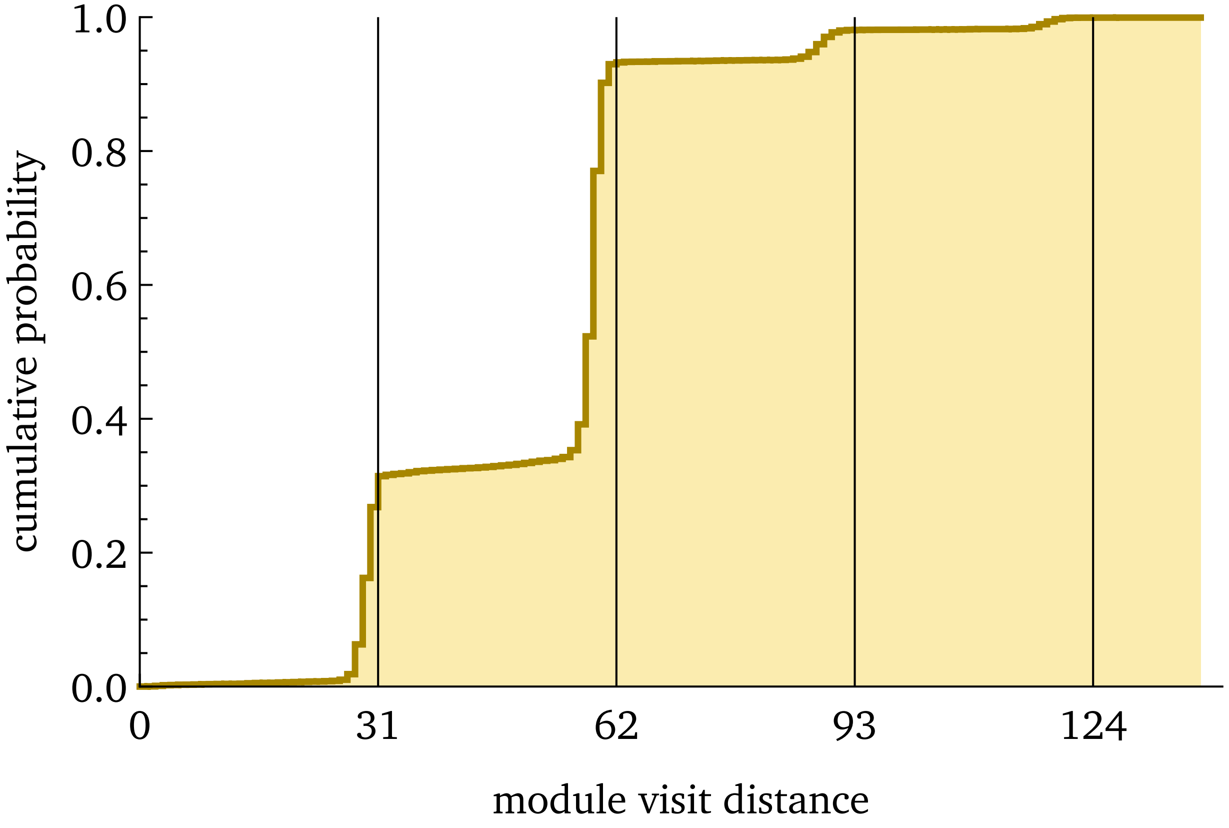


Module on N4 (distance 1)

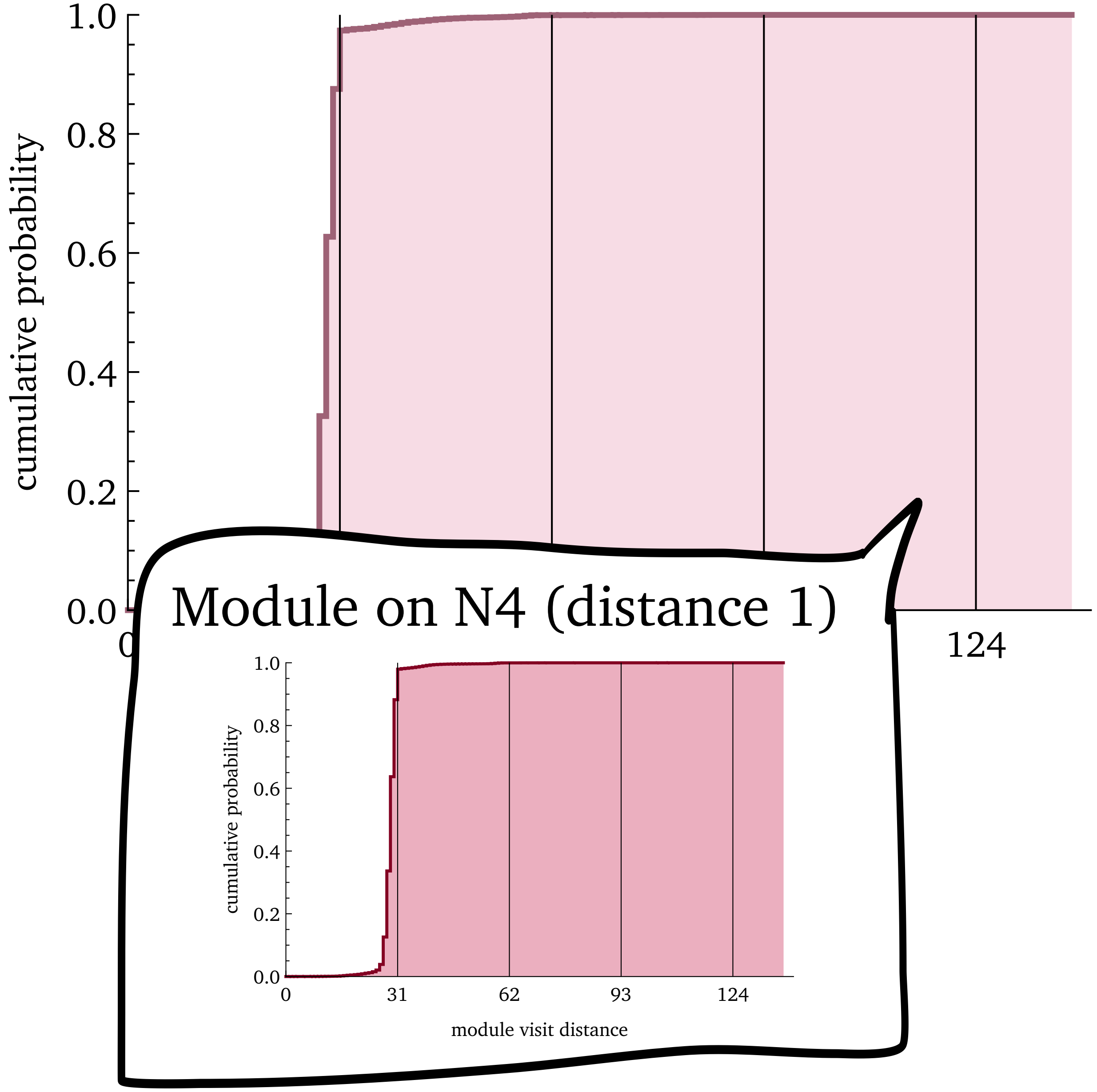
Setup: all nodes running, target on N0

Interlagos F&I Visit Distances

Module on N0 (distance 0)



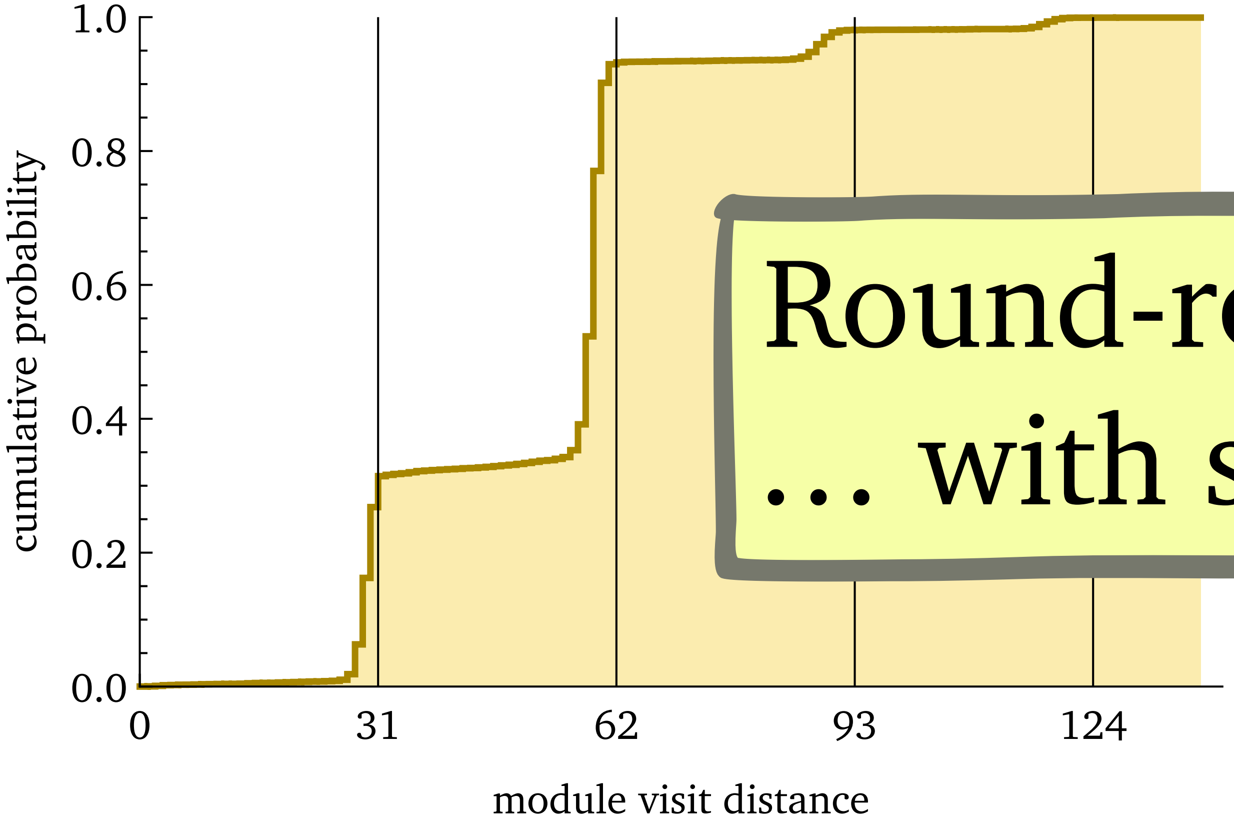
Module on N7 (distance 2)



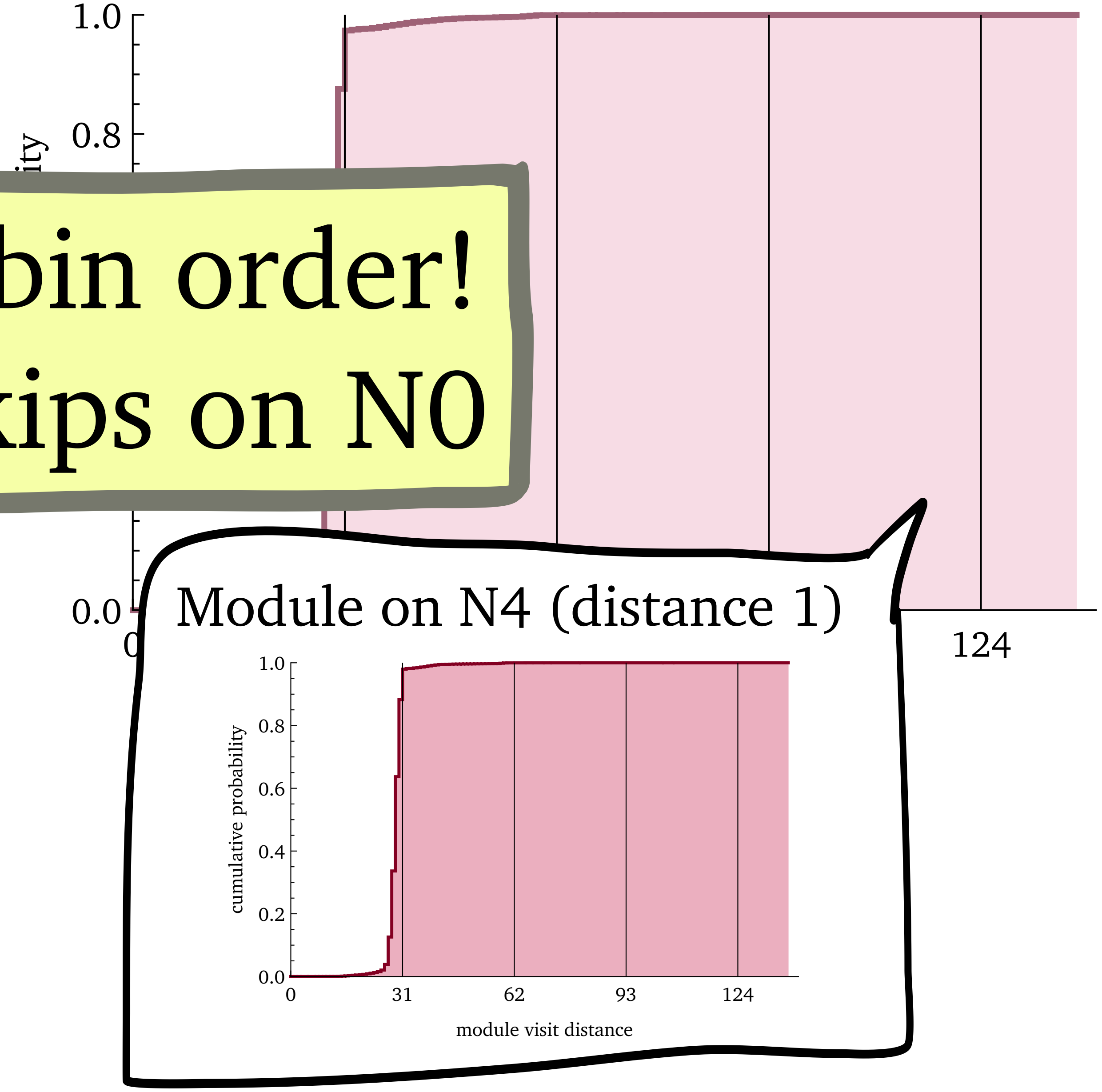
Setup: all nodes running, target on N0

Interlagos F&I Visit Distances

Module on N0 (distance 0)



Module on N7 (distance 2)



Round-robin order!
… with skips on N0

Setup: all nodes running, target on N0

Interlagos **F&I** Unfairness

distance	skip probability	mean visit length
0		
1		
2		

Interlagos **F&I** Unfairness

distance	skip probability	mean visit length
0	≈ .40 ☹	
1	≈ .03	
2	≈ .03	

Interlagos **F&I** Unfairness

distance	skip probability	mean visit length
0	≈ .40 ☹	≈ 1.1
1	≈ .03	≈ 1.1
2	≈ .03	≈ 1.6 ☺

Interlagos **F&I** Unfairness

distance	skip probability	mean visit length
0	≈ .40 ☹	≈ 1.1
1	≈ .03	≈ 1.1
2	≈ .03	≈ 1.6 ☺

💡 Think of skips as length-0 visits

Interlagos **F&I** Unfairness

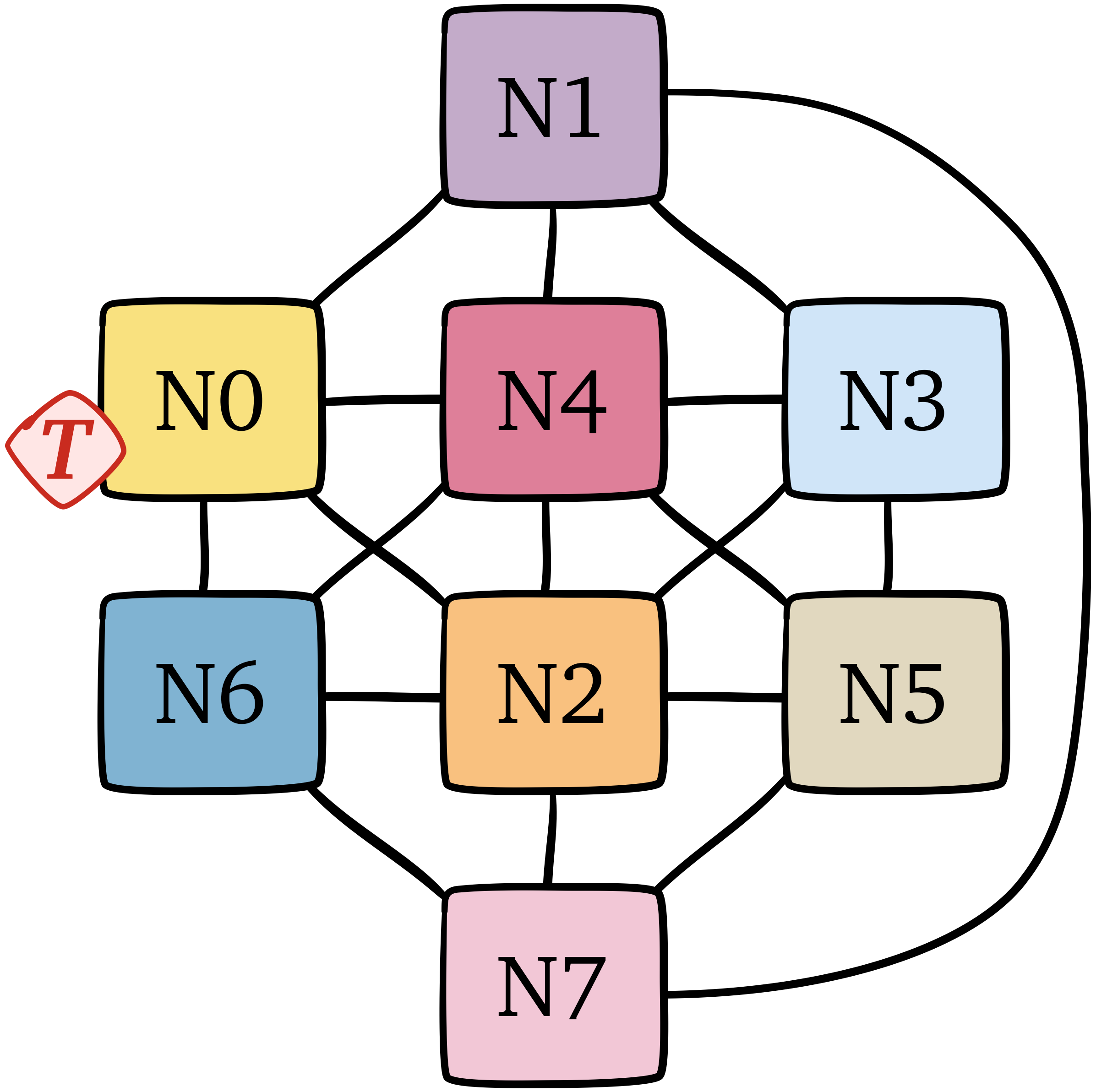
distance	skip probability	mean visit length
0	≈ .40 ☹	≈ 1.1
1	≈ .03	≈ 1.1
2	≈ .03	≈ 1.6 ☺

Think of skips as length-0 visits

Larger distance ⇒ longer module visit

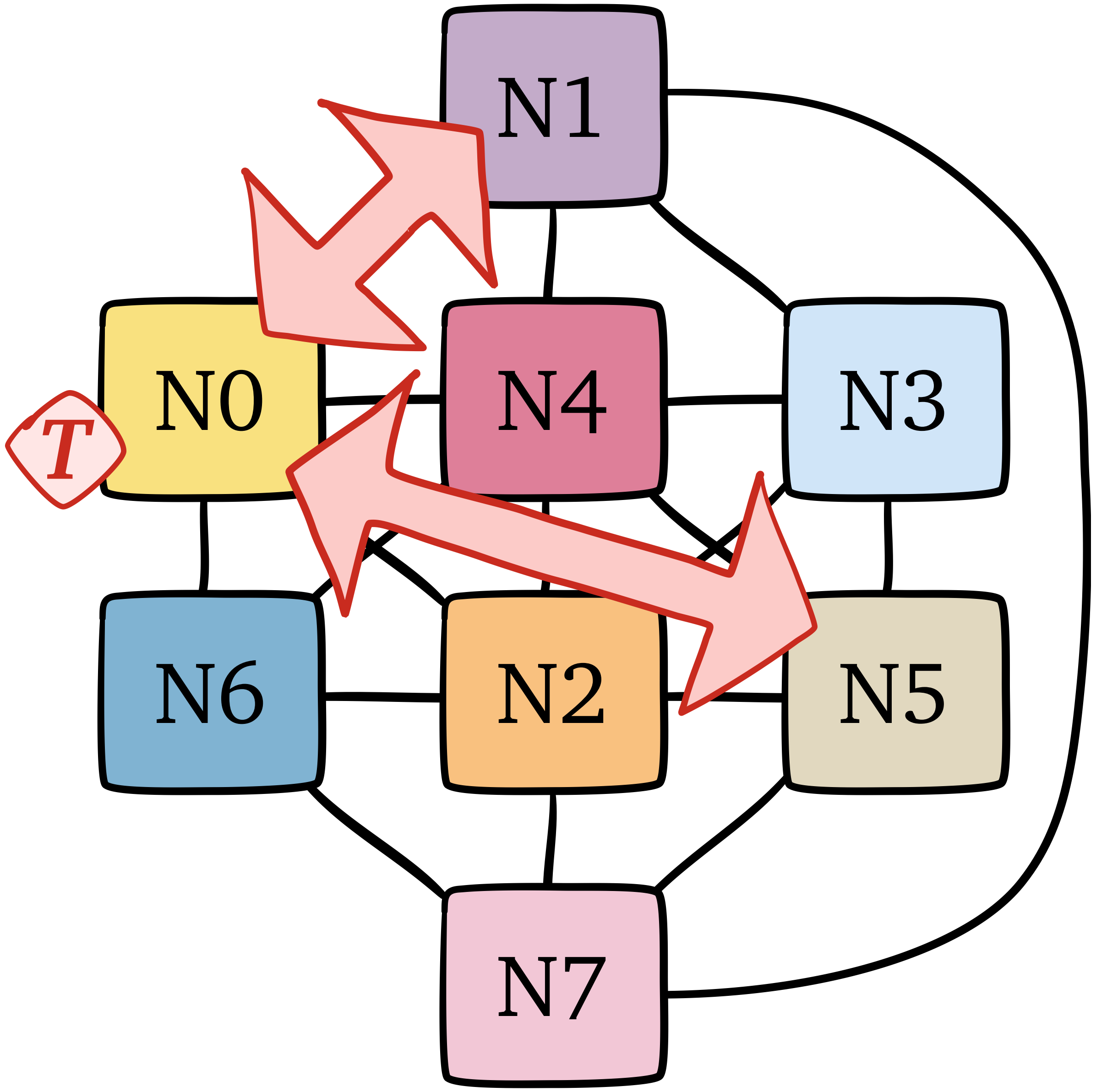
Potential Explanation

Target's cache coherence messages go to/from N0



Potential Explanation

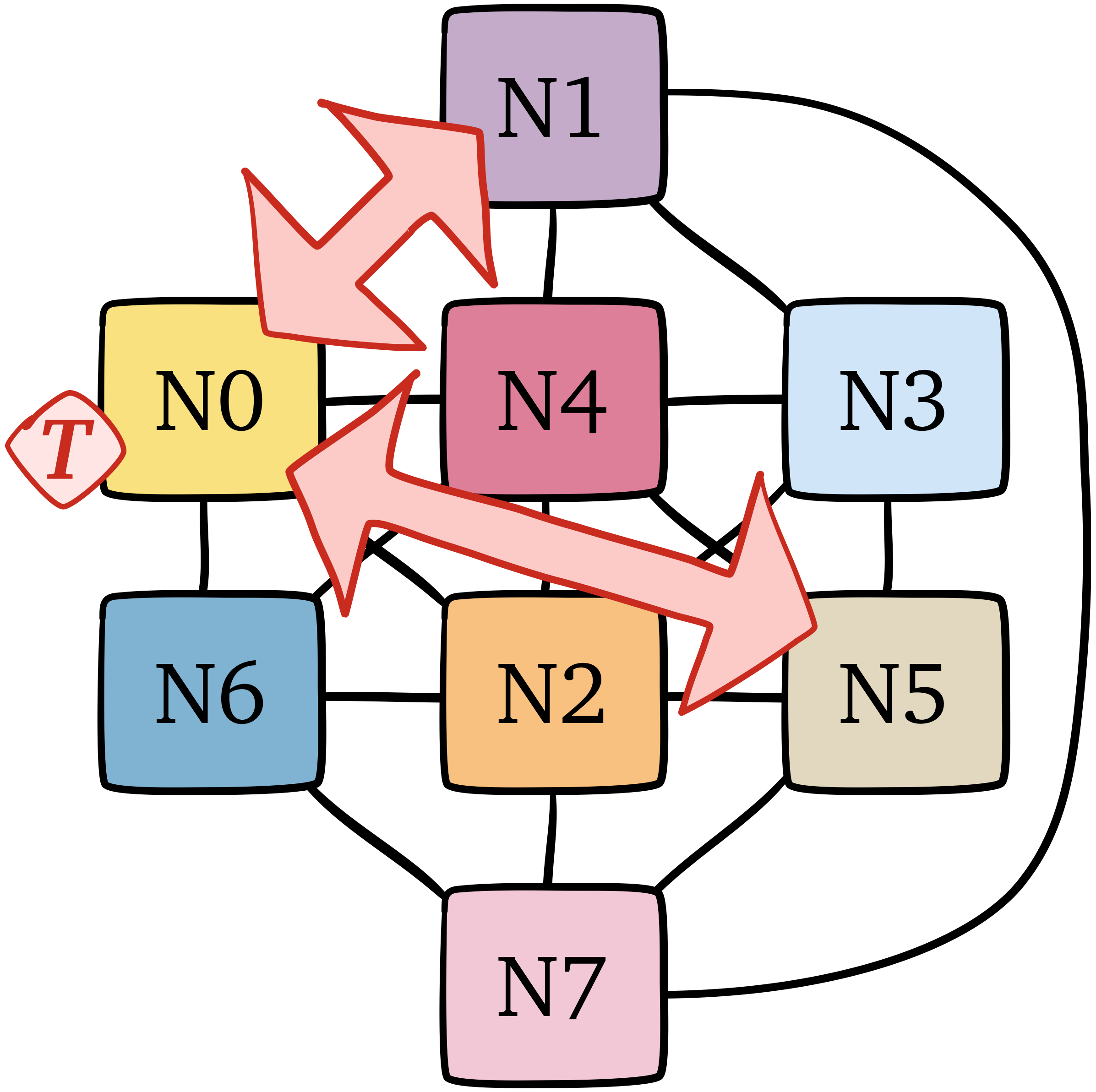
Target's cache coherence messages go to/from N0



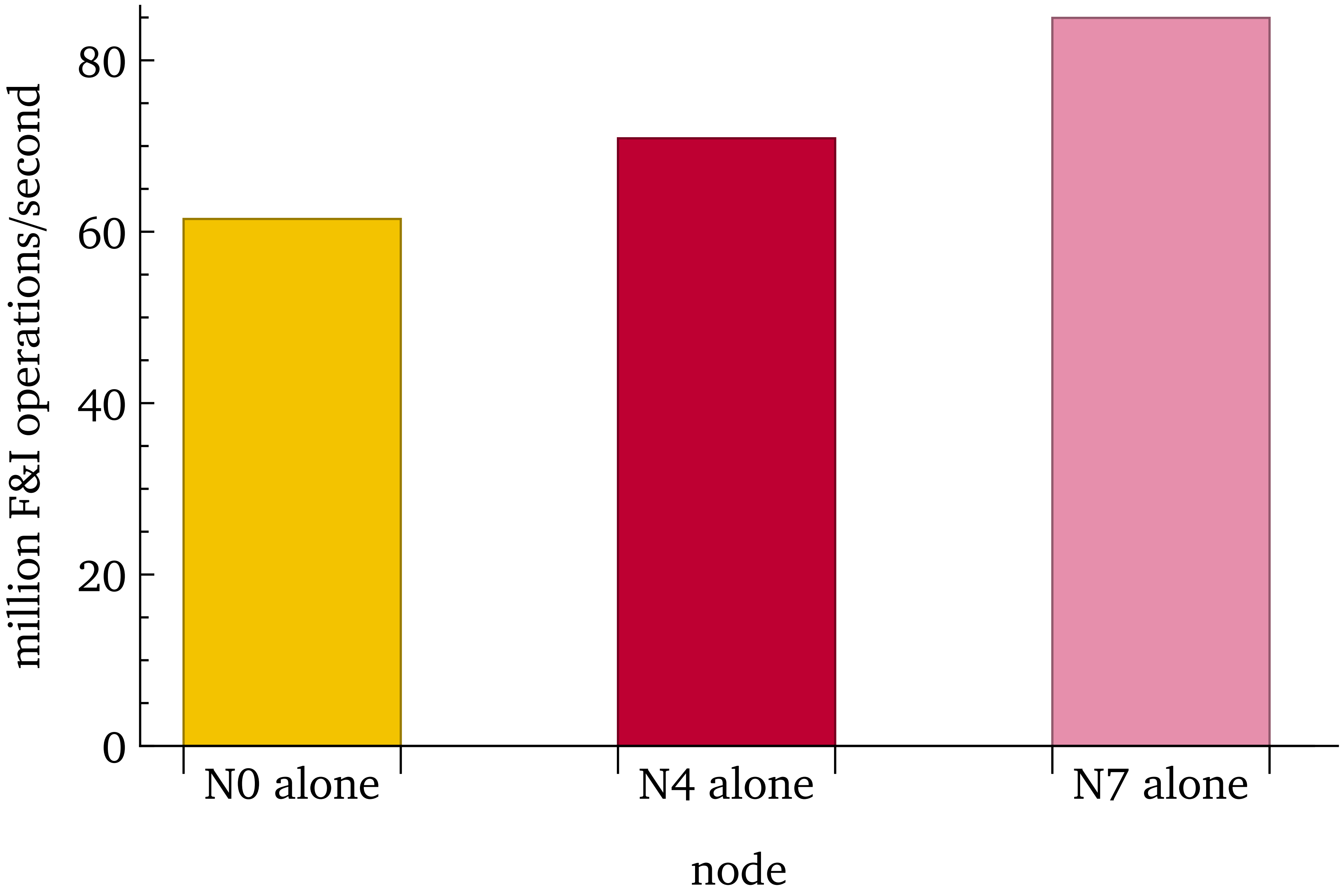
Potential Explanation

Target’s cache coherence messages go to/from N0

… even though **target** always in some module’s L1!

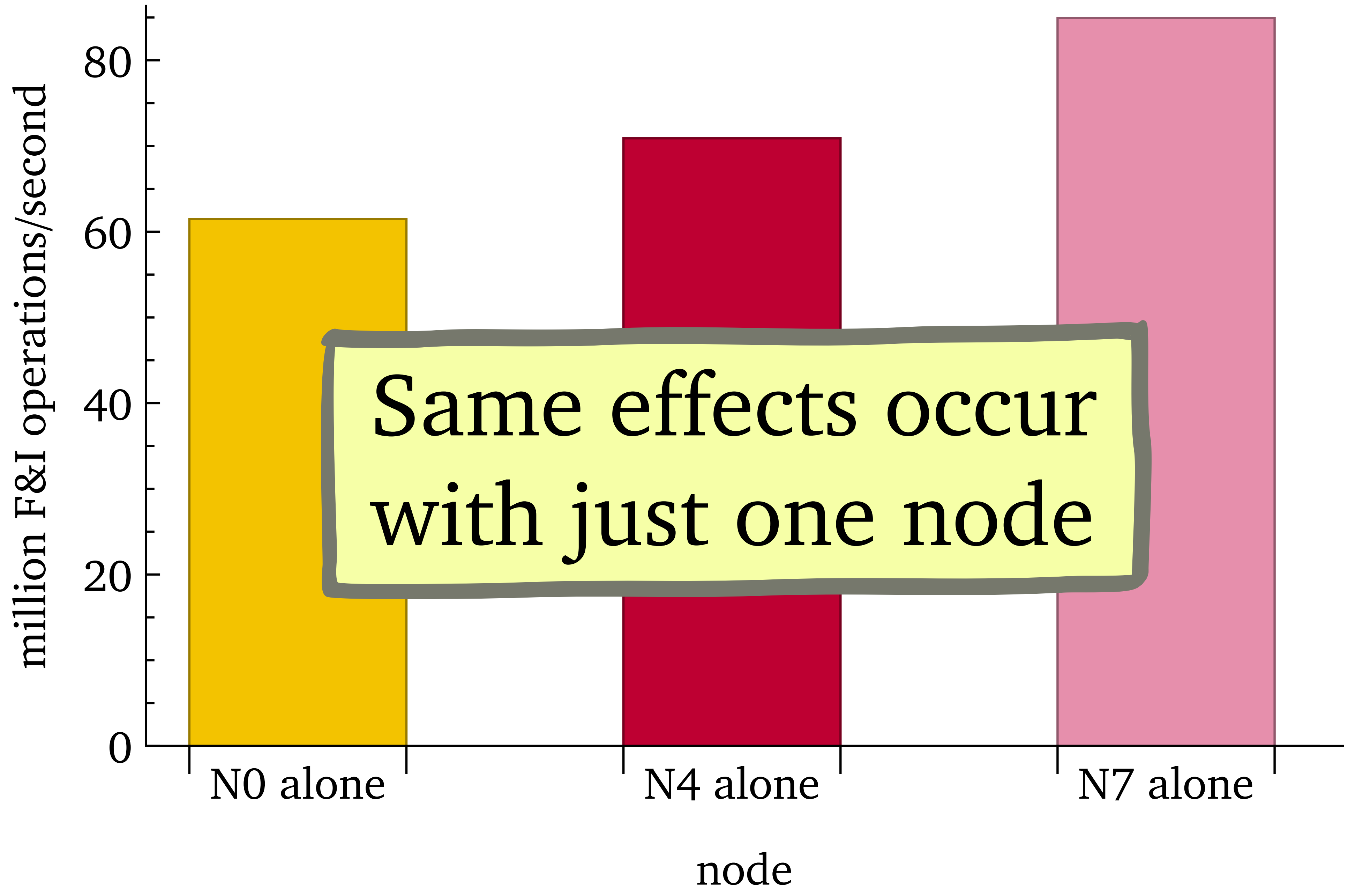


Interlagos F&I One-Node Throughput



Setup: *one node running at a time*, target on N0

Interlagos F&I One-Node Throughput



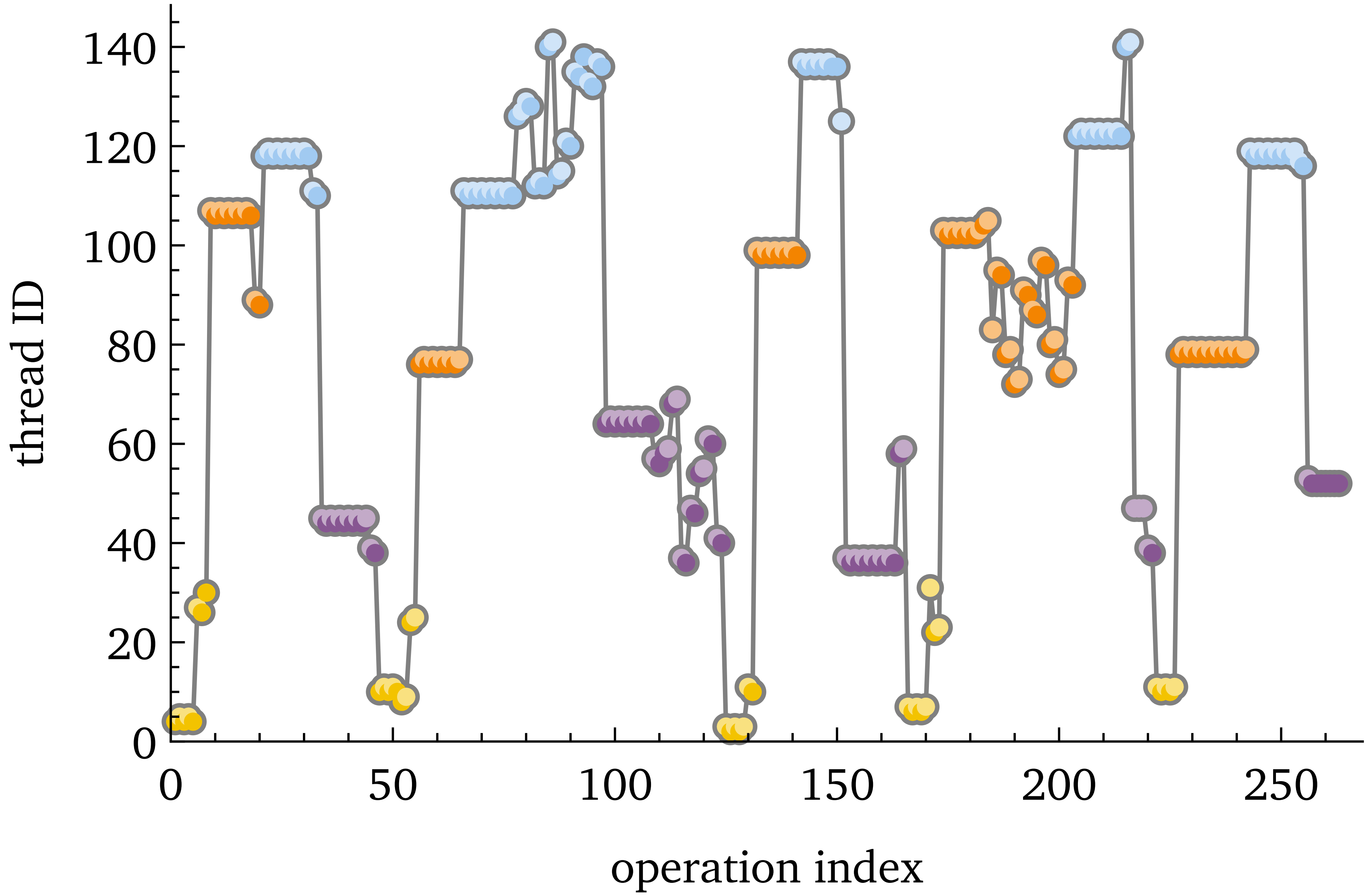
Same effects occur with just one node

Setup: *one node running at a time*, target on N0

Intel Broadwell-EX
F&I Experiments

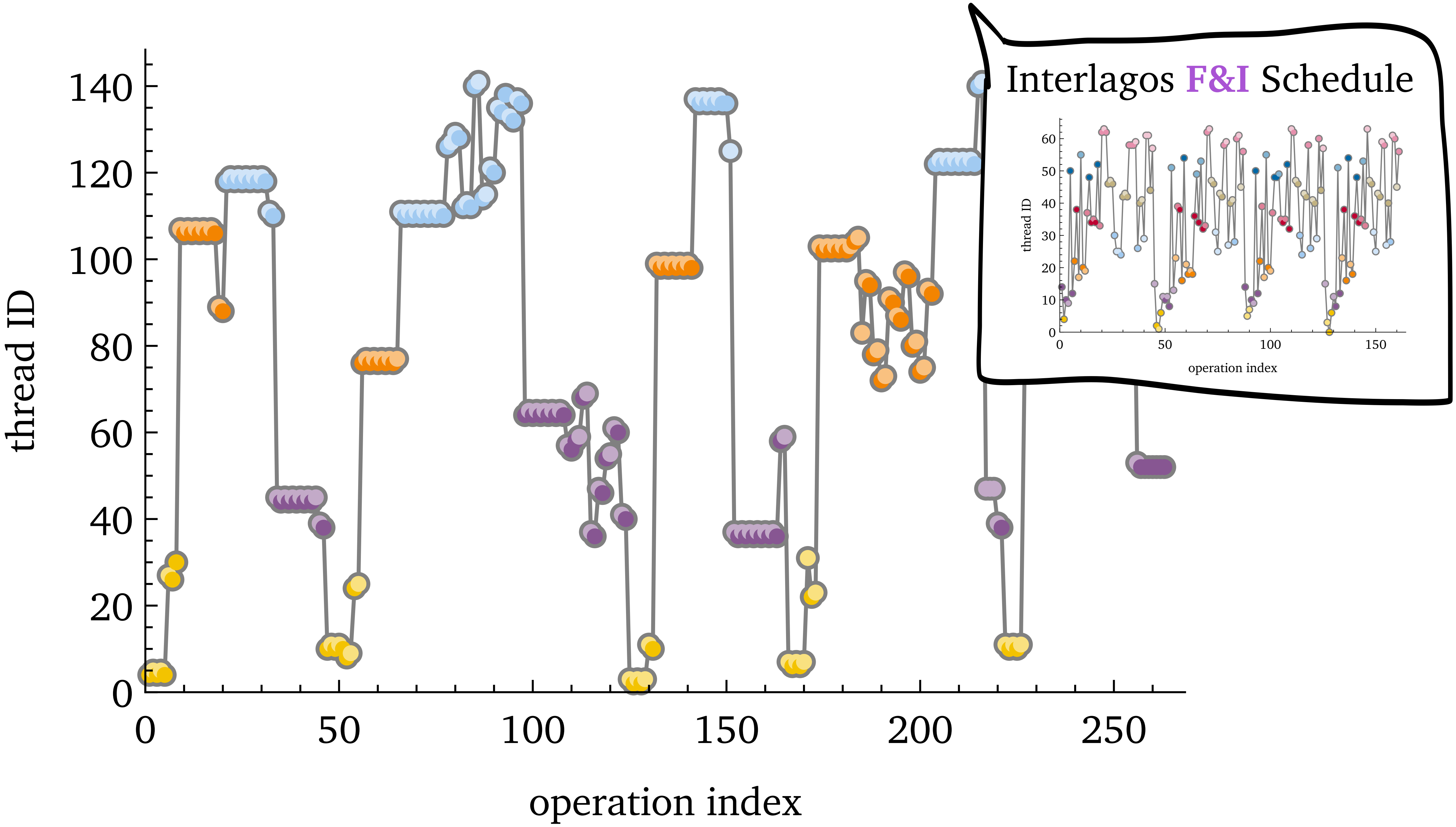
(preview)

Broadwell-EX F&I Schedule



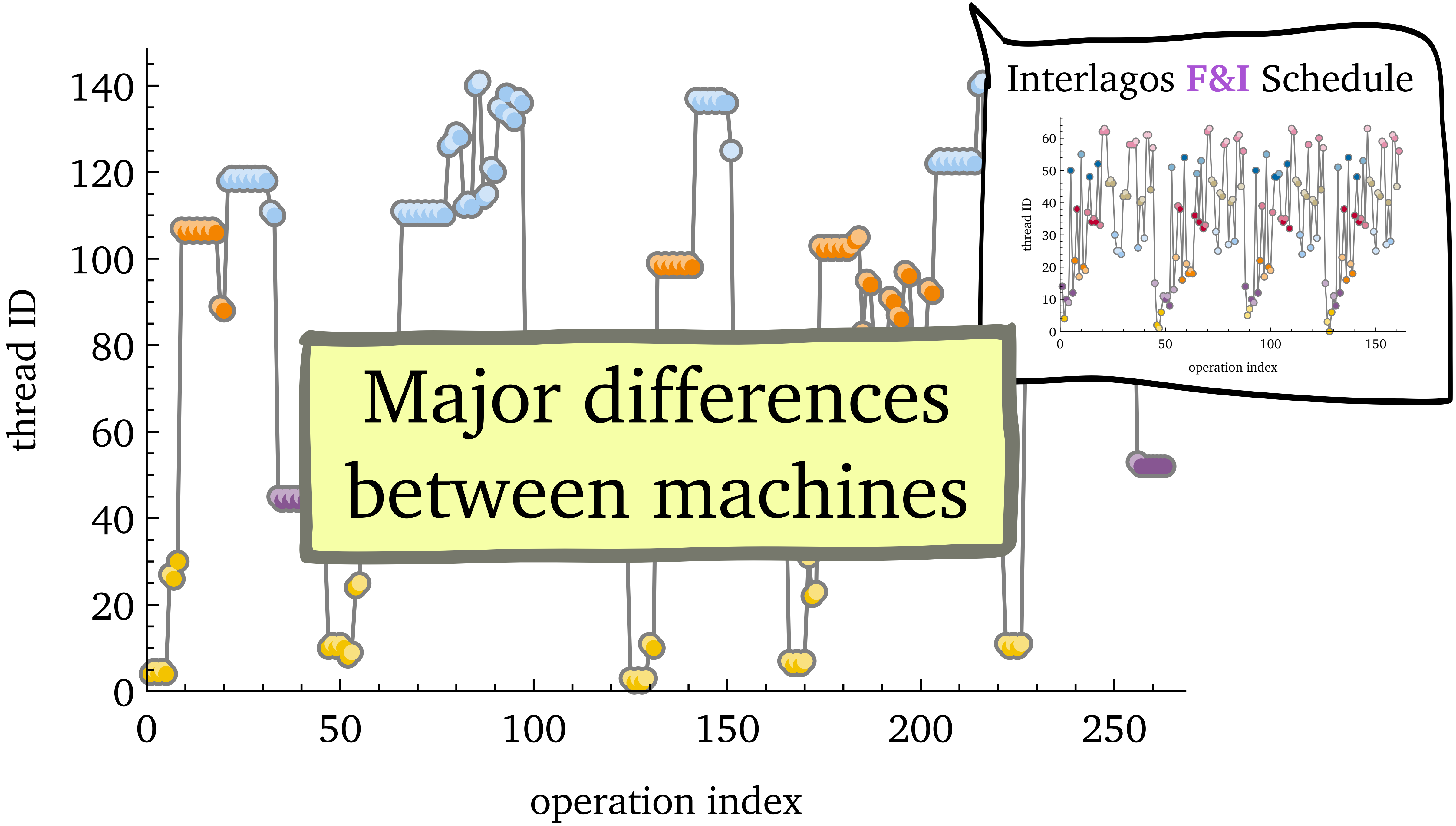
Setup: all nodes running, target on N0

Broadwell-EX **F&I** Schedule


Interlagos **F&I** Schedule

Setup: all nodes running, target on N0

Broadwell-EX F&I Schedule

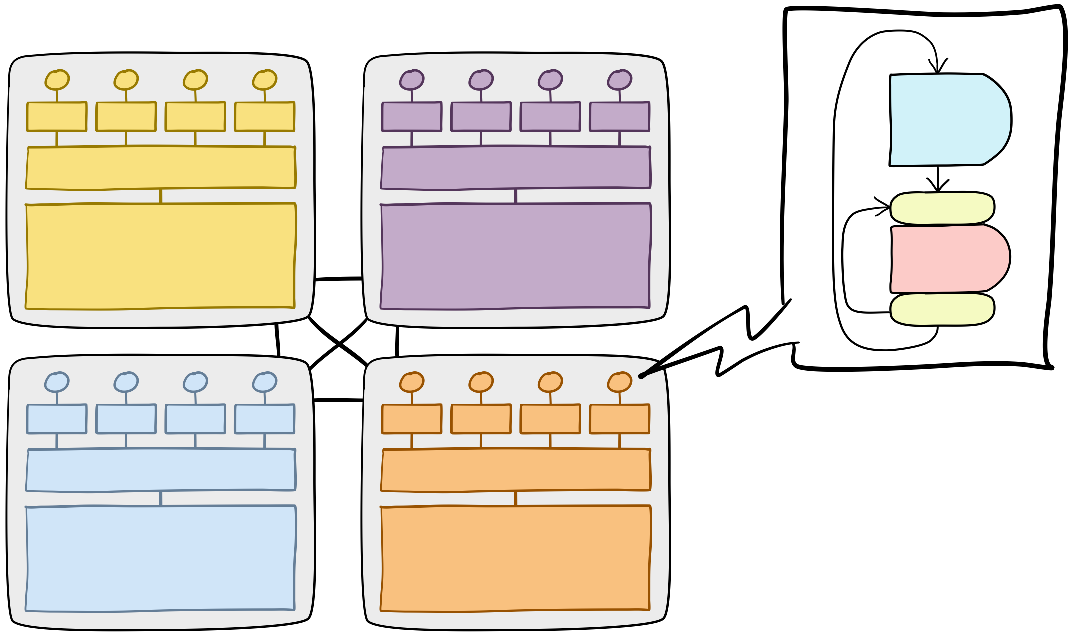


Interlagos F&I Schedule

Major differences between machines

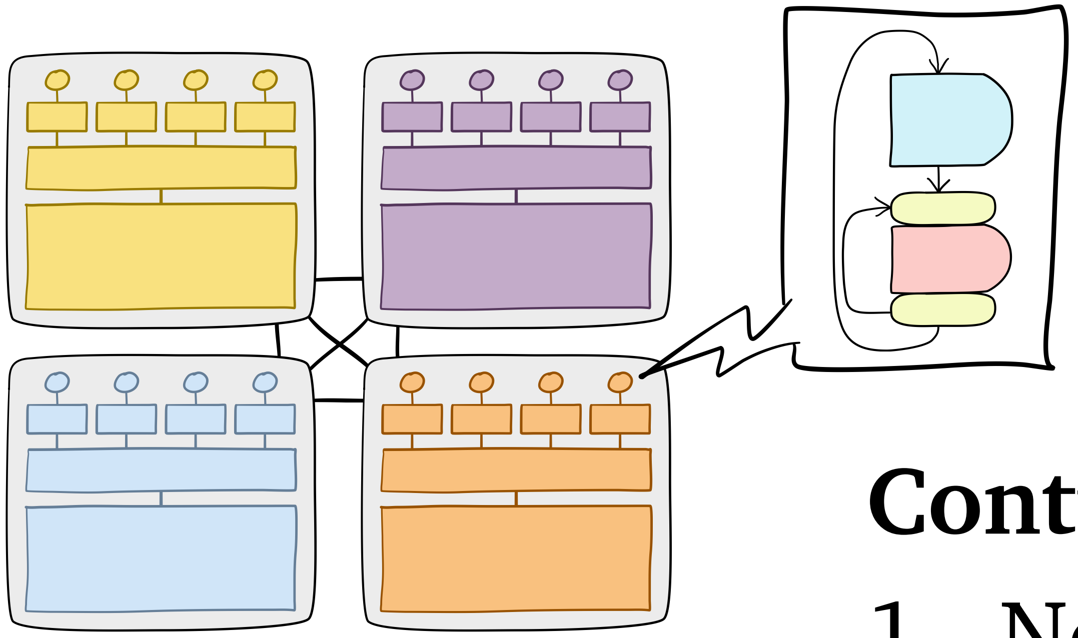
Setup: all nodes running, target on N0

Summary



Question: how does NUMA affect *memory access schedules*?

Summary

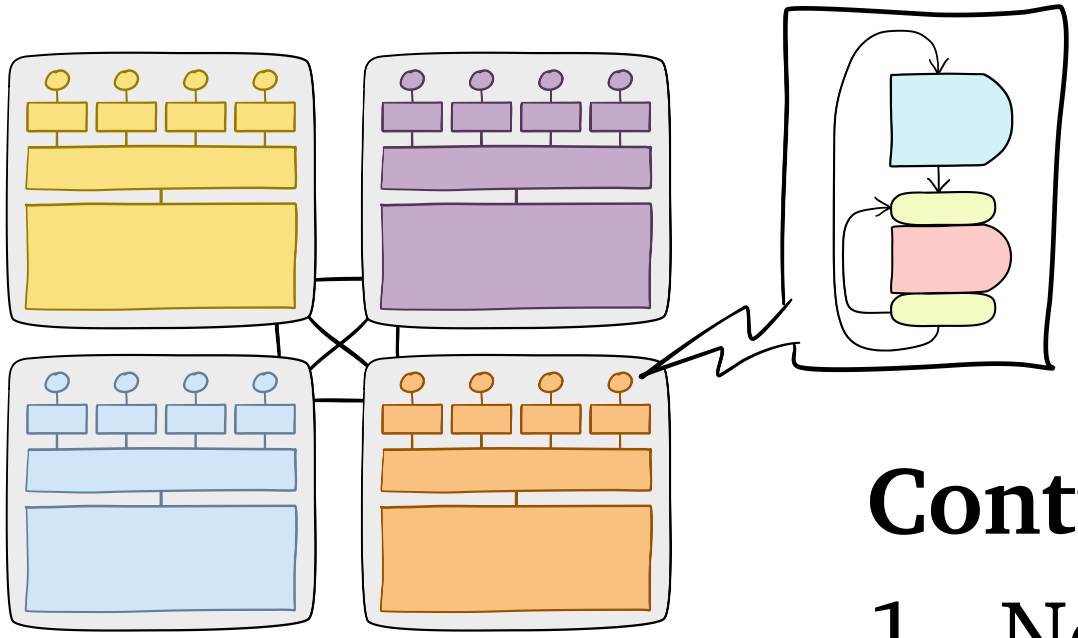


Question: how does NUMA affect *memory access schedules*?

Contributions:

1. New tool, **Severus**
2. Case studies on two machines

Summary



Question: how does NUMA affect *memory access schedules*?

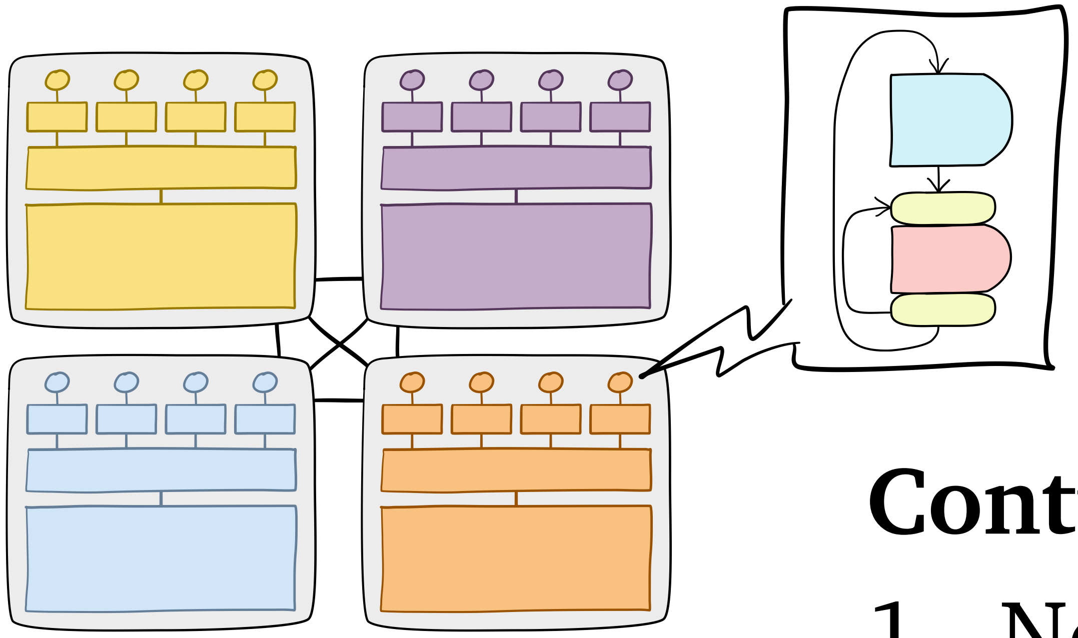
Contributions:

1. New tool, **Severus**
2. Case studies on two machines

Findings:

- NUMA can be unfair to *local* cores
- Schedule is decipherable!

Summary



Question: how does NUMA affect *memory access schedules*?

Contributions:

1. New tool, **Severus**
2. Case studies on two machines

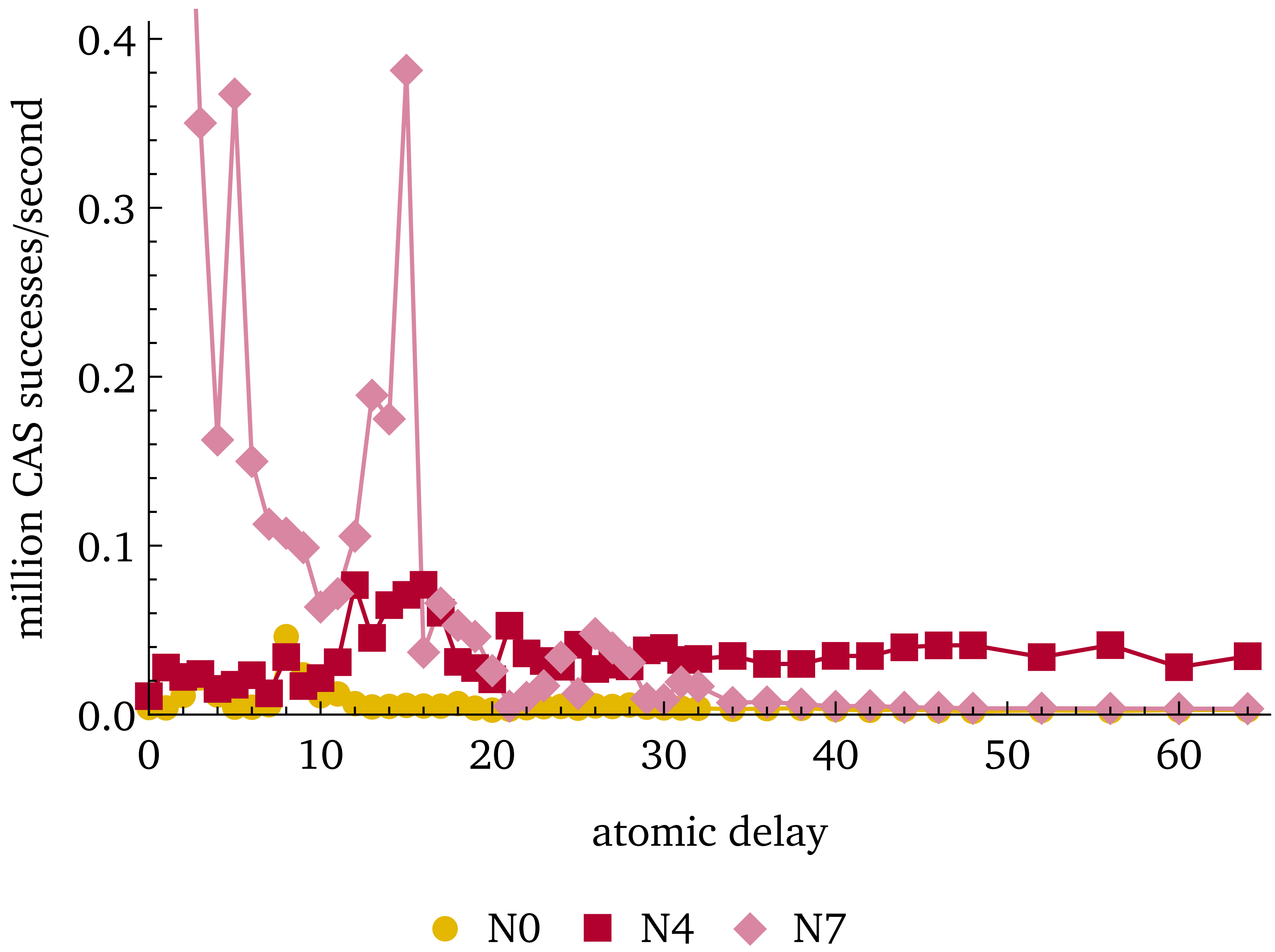
Findings:

- NUMA can be unfair to *local* cores
- Schedule is decipherable!

`https://github.com/cmuparlay/severus`

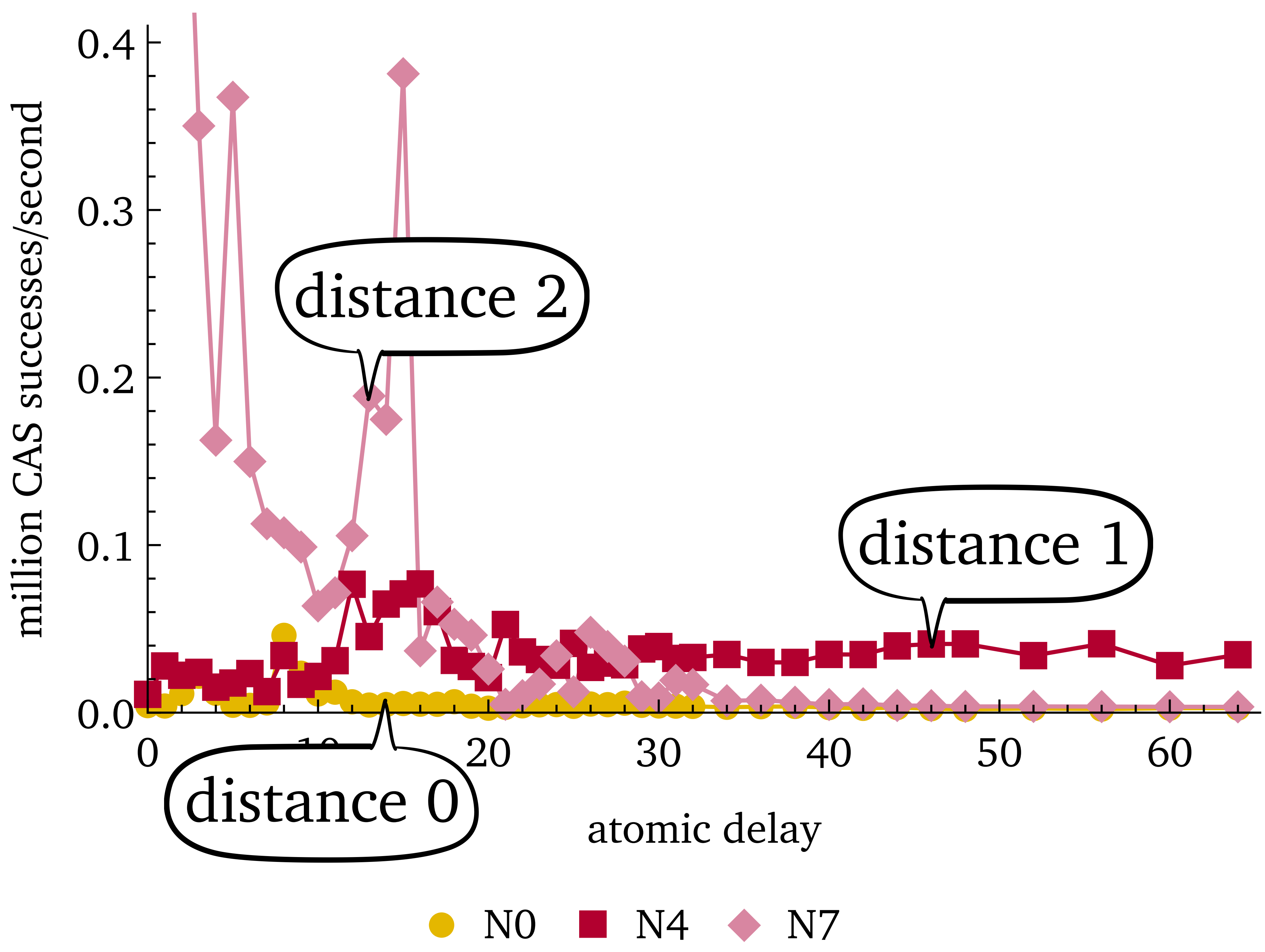
AMD Interlagos **Read-CAS** Experiments

Interlagos **Read-CAS** Throughput



Setup: all nodes running, target on N0

Interlagos Read-CAS Throughput



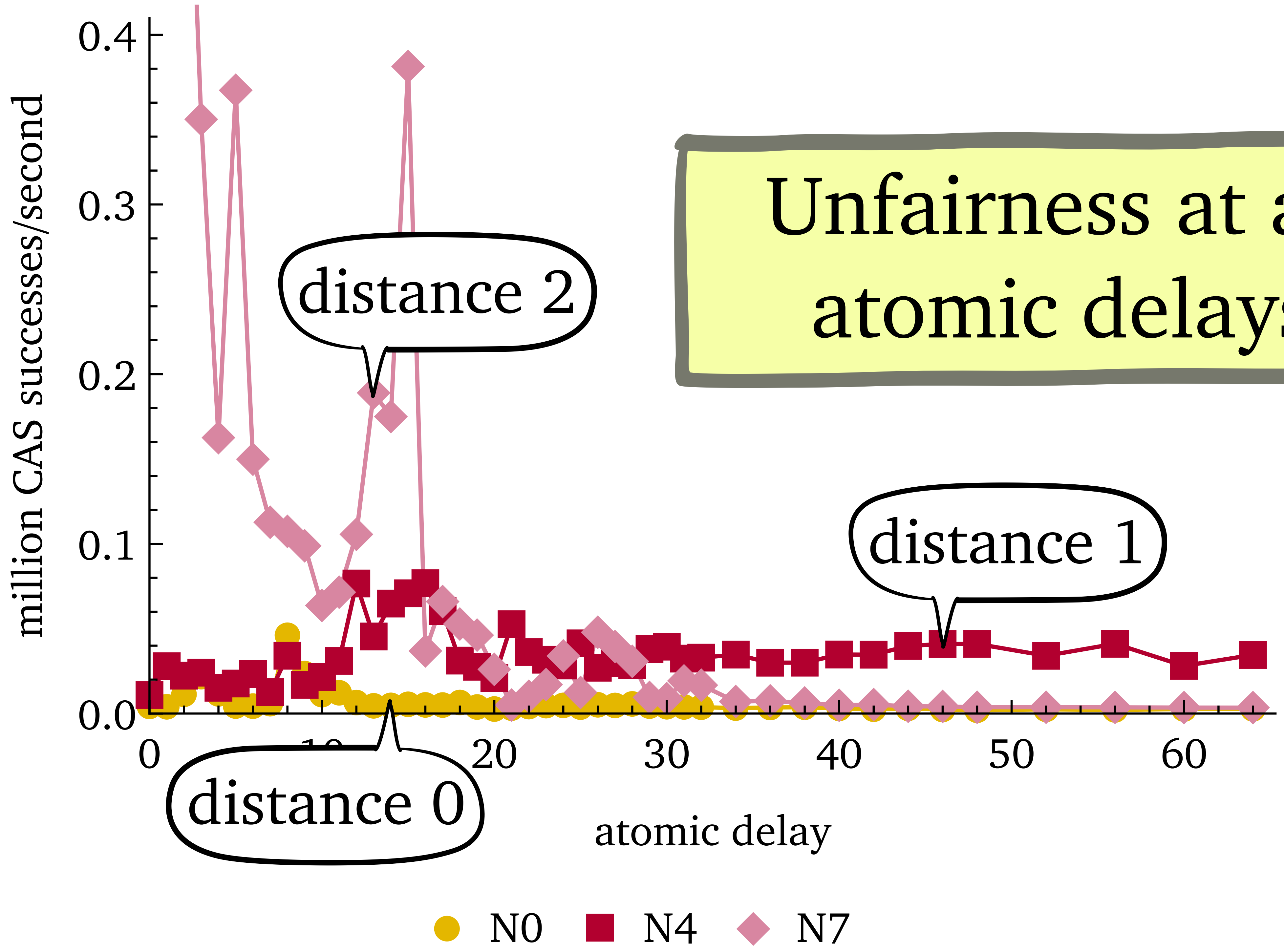
distance 2

distance 1

distance 0

Setup: all nodes running, target on N0

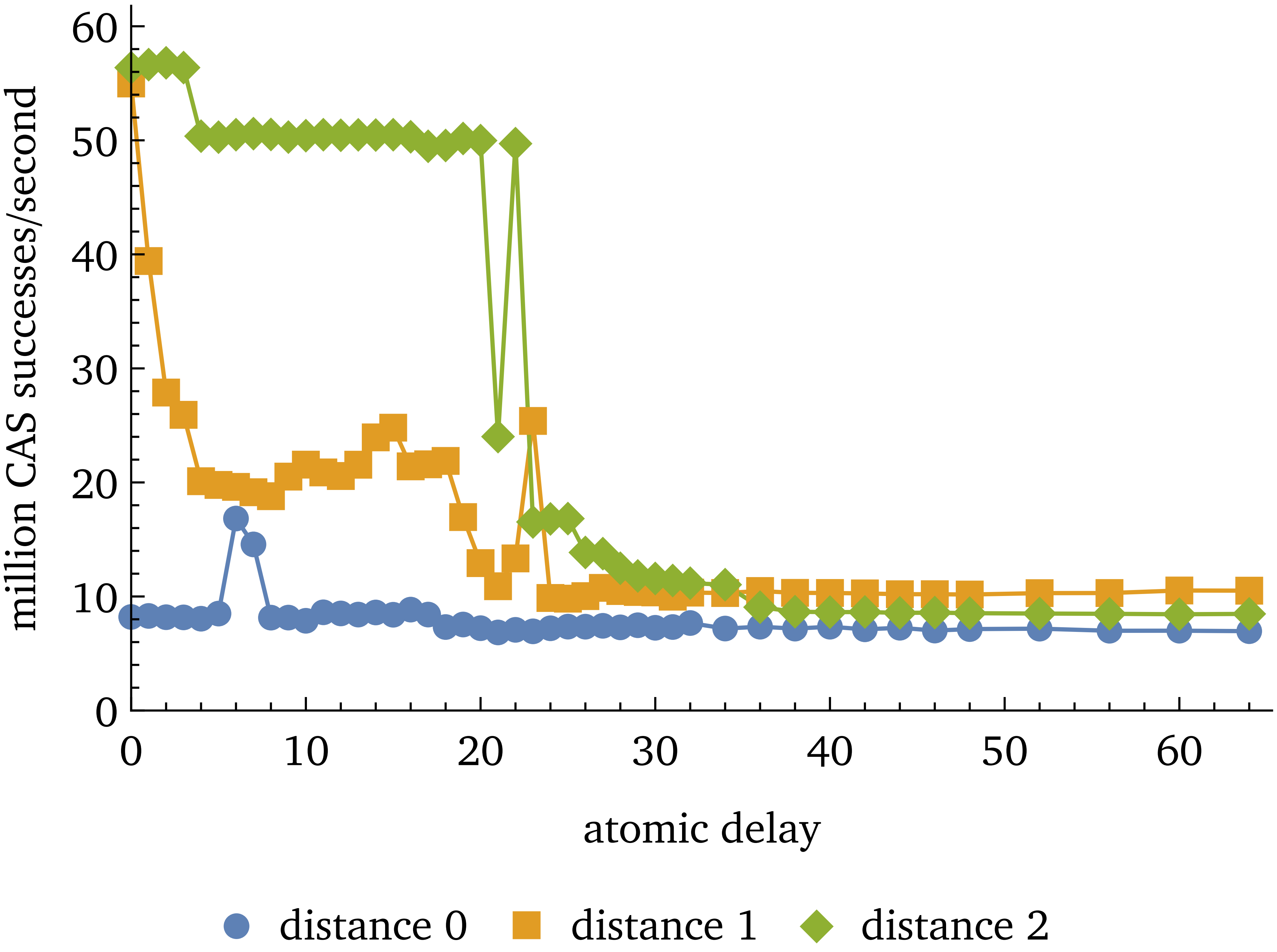
Interlagos Read-CAS Throughput



Unfairness at all atomic delays

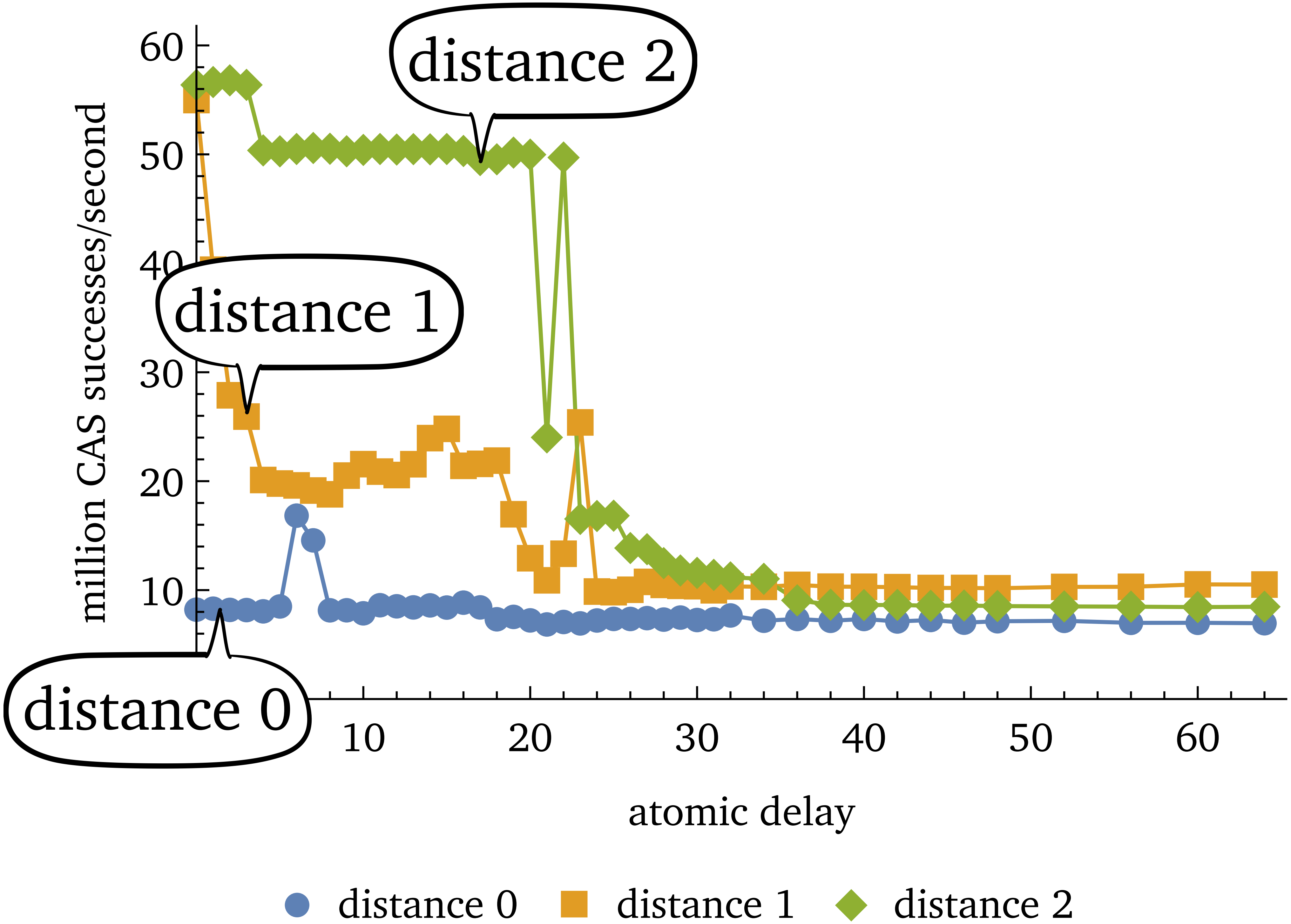
Setup: all nodes running, target on N0

Interlagos Read-CAS Many-Target Throughput



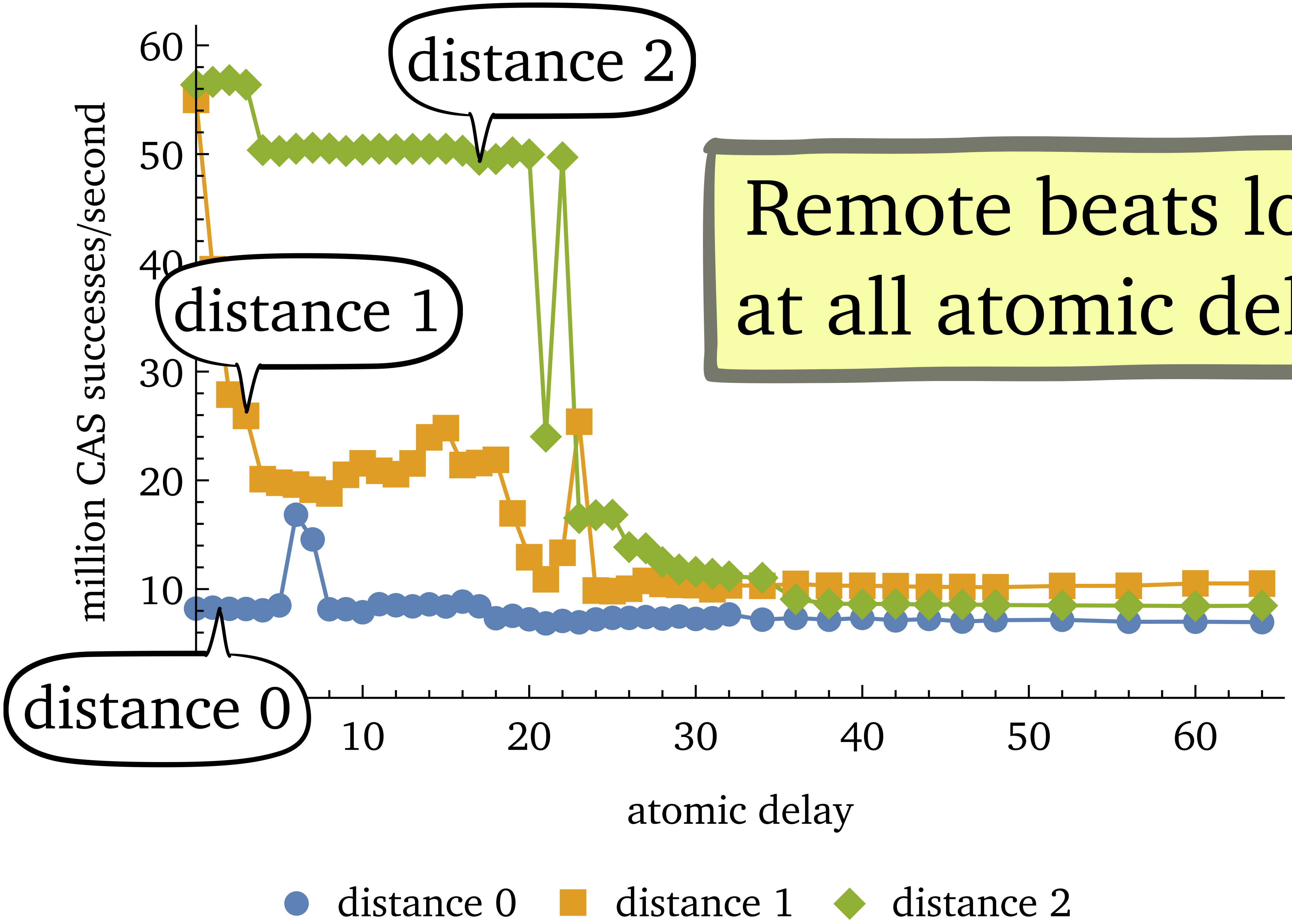
Setup: all nodes running, *targets on each node*

Interlagos Read-CAS Many-Target Throughput



Setup: all nodes running, *targets on each node*

Interlagos Read-CAS Many-Target Throughput



Remote beats local at all atomic delays

Setup: all nodes running, *targets on each node*