SNUG: Architectural Support for Relaxed Concurrent Priority Queueing in Chip Multiprocessors

Azin Heidarshenas, Tanmay Gangwani*, Serif Yesil, Adam Morrison, and Josep Torrellas*

* Co-first authorship

International Conference on Supercomputing 2020

Priority-based Task Scheduling

Tasks are executed based on their *priority order*

If T_1 has higher priority than T_2 it gets executed before T_2

Priority-based Task Scheduling

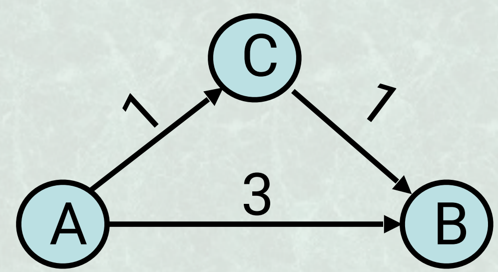
Tasks are executed based on their *priority order*

If T_1 has higher priority than T_2 it gets executed before T_2

Example: Dijkstra's Single-Source Shortest Path (SSSP)

- Source vertex A

Vertex	Distance
A	0
B	2
C	1



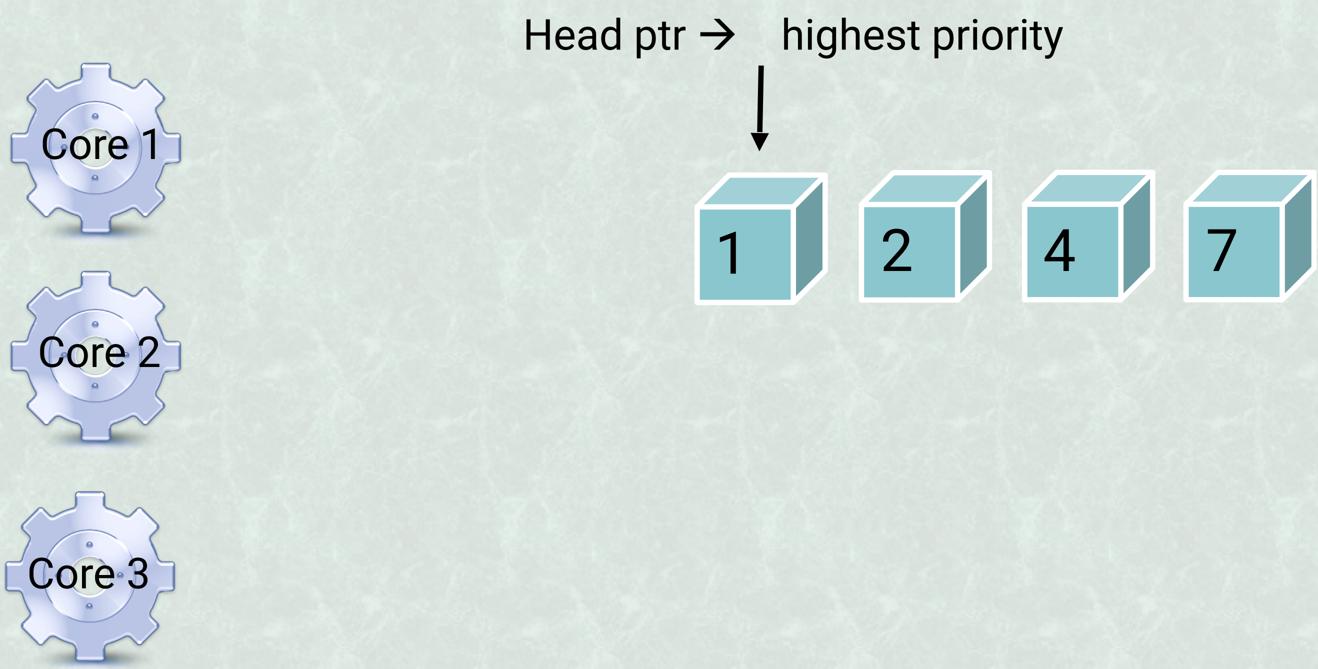
vertex

priority

A, 0 C, 1 B, 2 B, 3

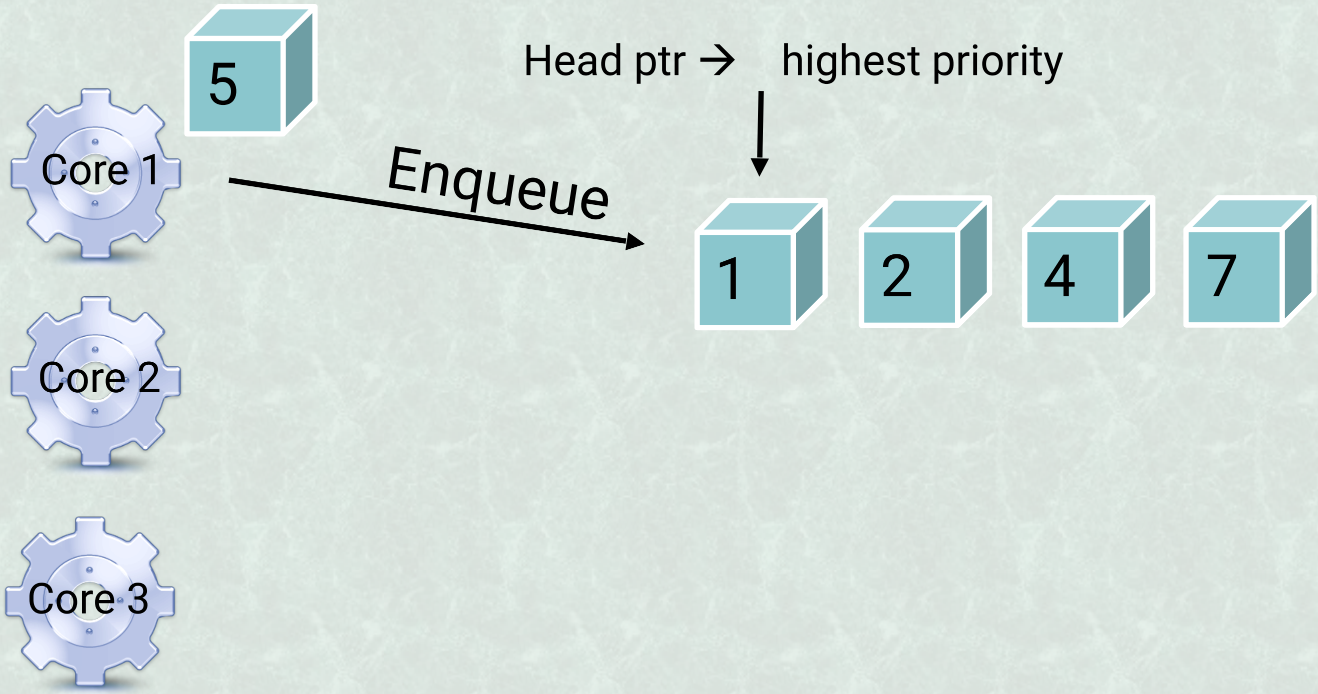
Concurrent Priority Queue (PQ)

Parallel threads ***dequeue*** (pop) tasks and ***enqueue*** (push) new ones



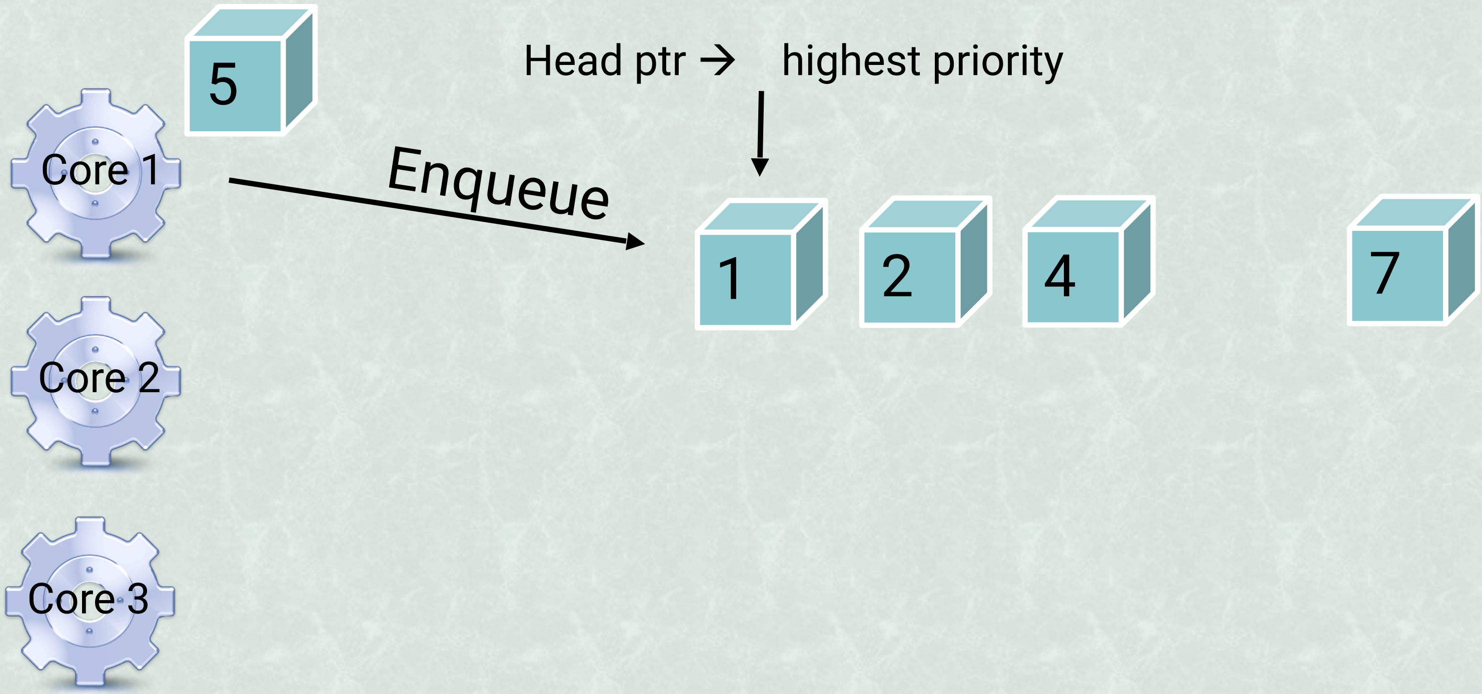
Concurrent Priority Queue (PQ)

Parallel threads ***dequeue*** (pop) tasks and ***enqueue*** (push) new ones



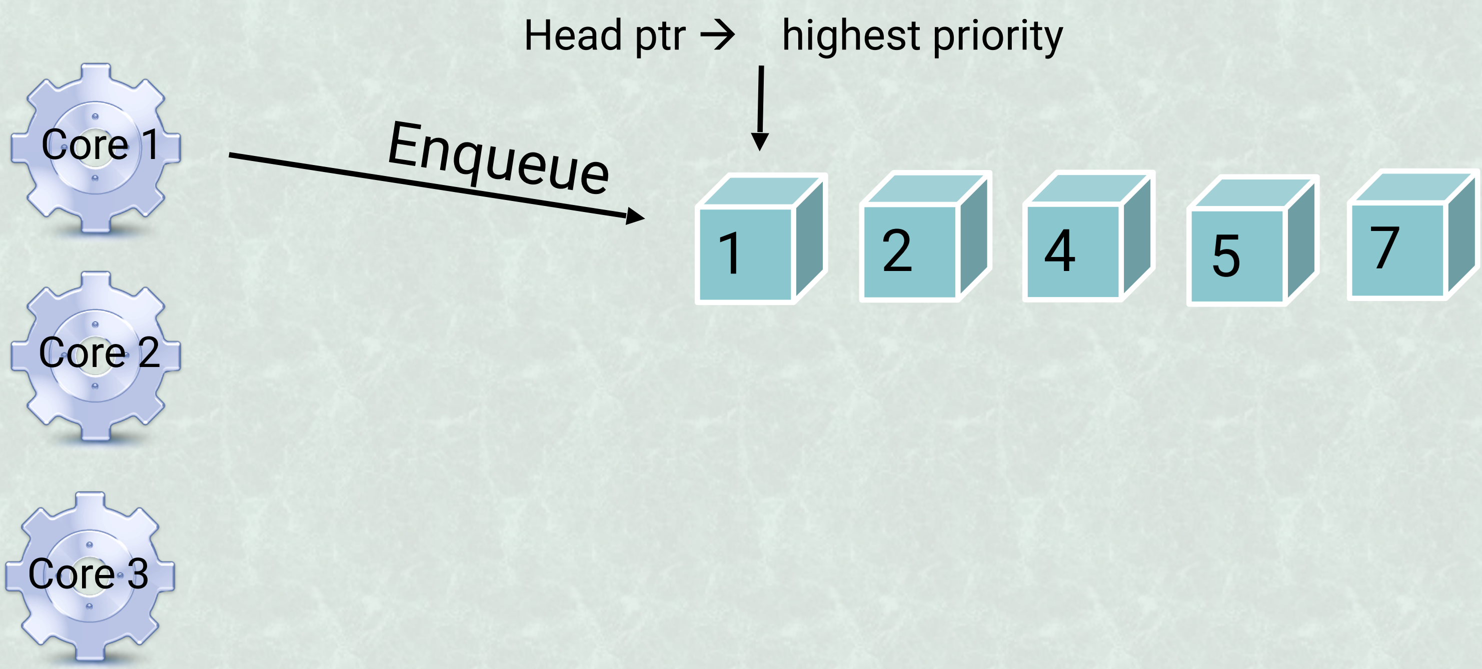
Concurrent Priority Queue (PQ)

Parallel threads ***dequeue*** (pop) tasks and ***enqueue*** (push) new ones



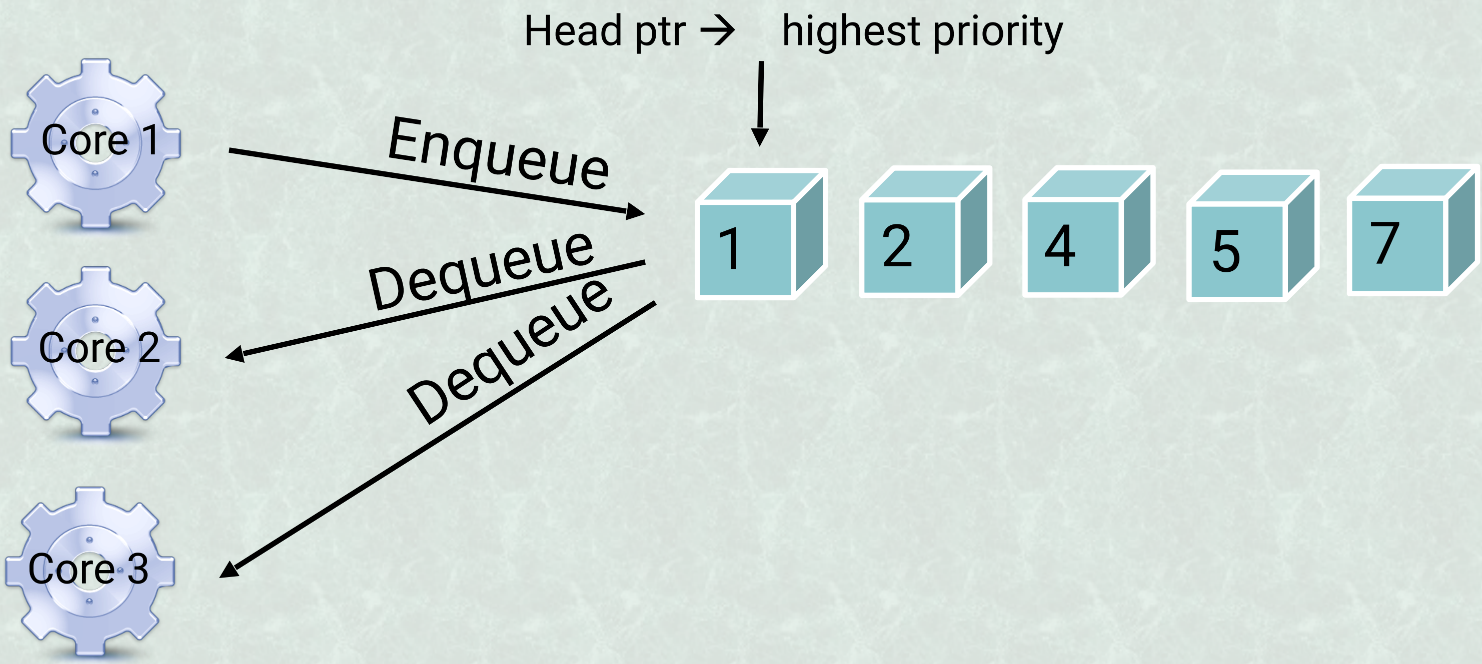
Concurrent Priority Queue (PQ)

Parallel threads ***dequeue*** (pop) tasks and ***enqueue*** (push) new ones



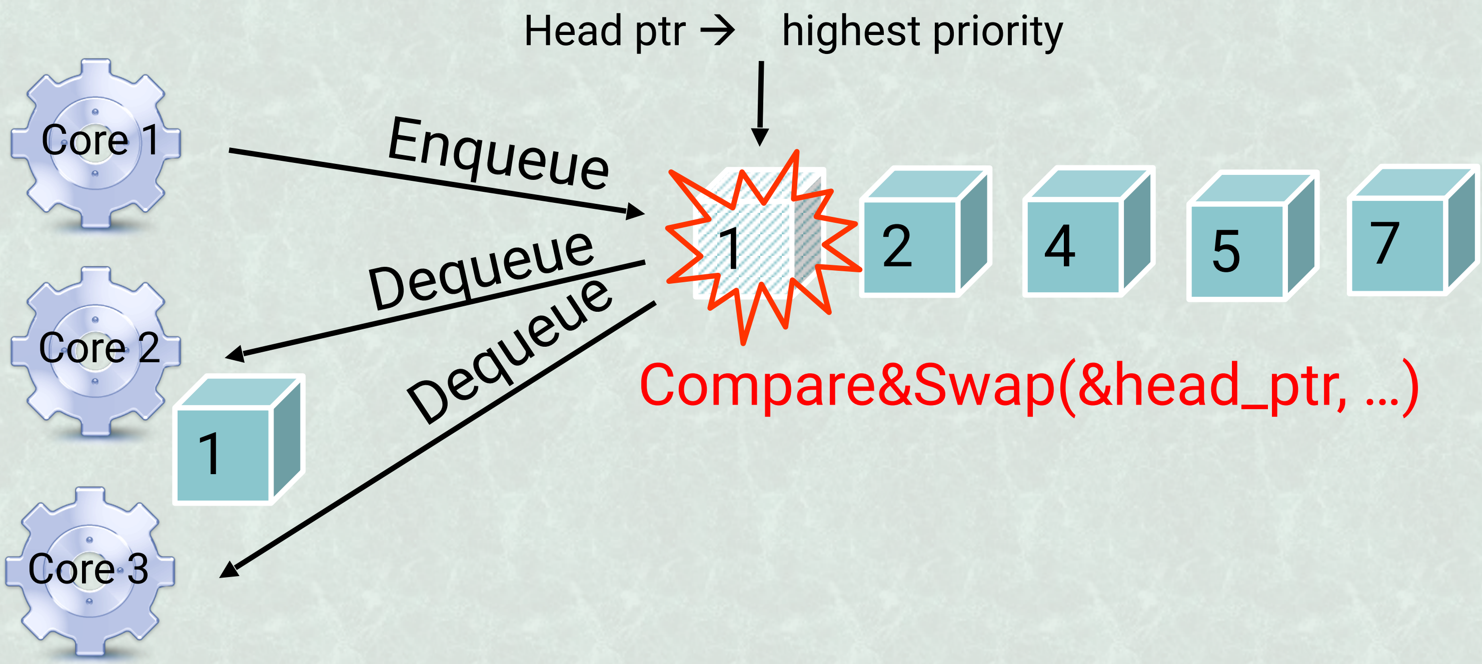
Concurrent Priority Queue (PQ)

Parallel threads ***dequeue*** (pop) tasks and ***enqueue*** (push) new ones



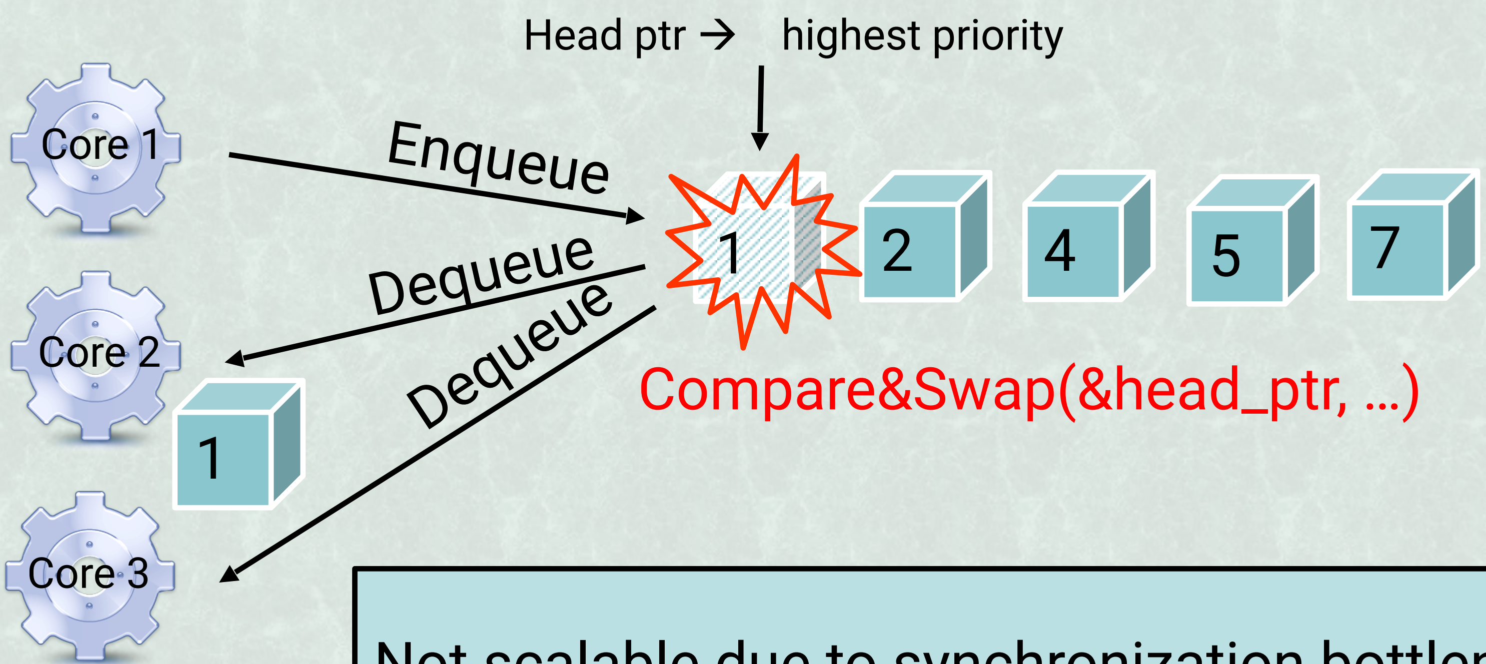
Concurrent Priority Queue (PQ)

Parallel threads ***dequeue*** (pop) tasks and ***enqueue*** (push) new ones



Concurrent Priority Queue (PQ)

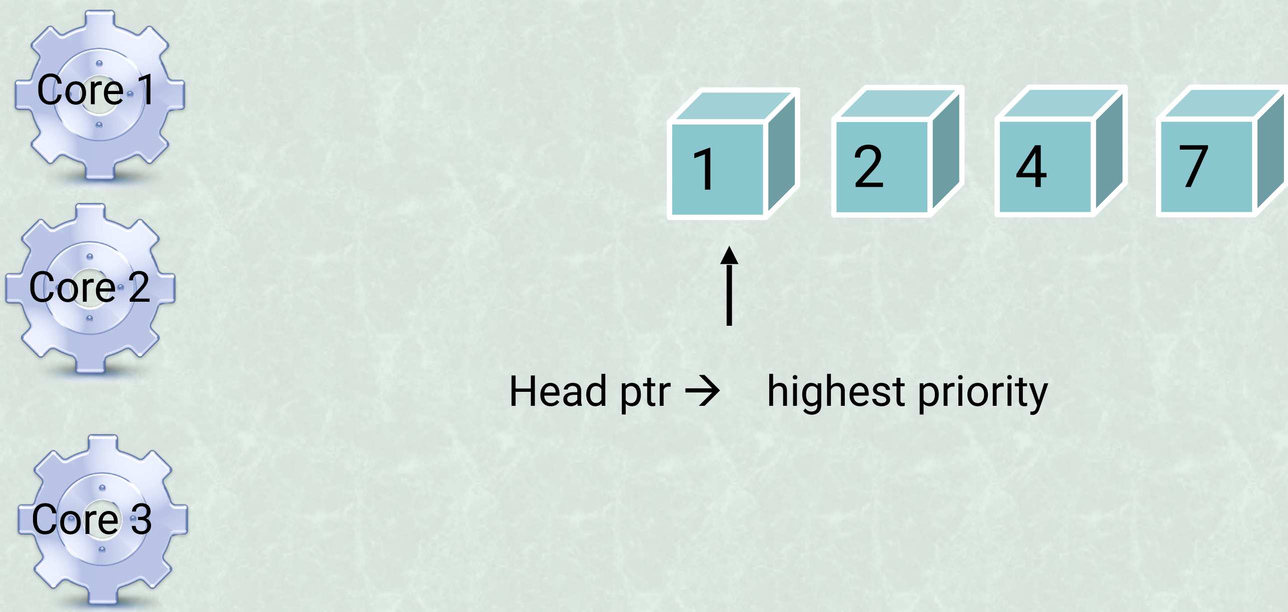
Parallel threads ***dequeue*** (pop) tasks and ***enqueue*** (push) new ones



Not scalable due to synchronization bottleneck

Relaxed PQ

Relaxed Priority → alleviate synchronization at the dequeues

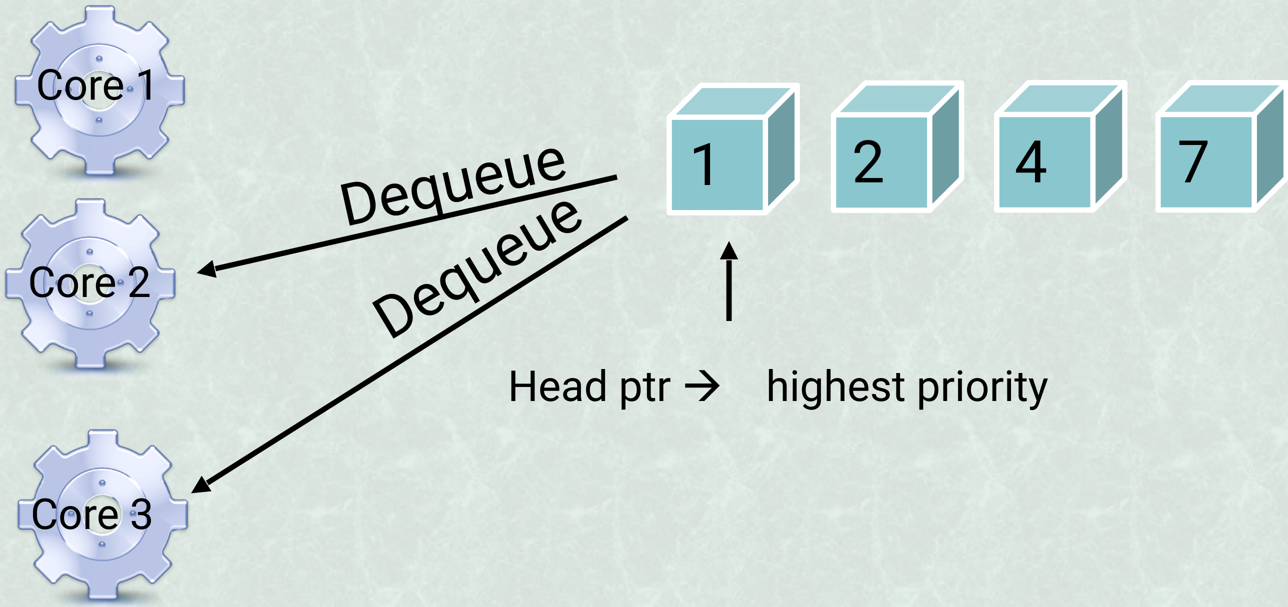


In many applications, relaxed priorities don't harm correctness

Graph applications, discrete event simulation, ...

Relaxed PQ

Relaxed Priority → alleviate synchronization at the dequeues

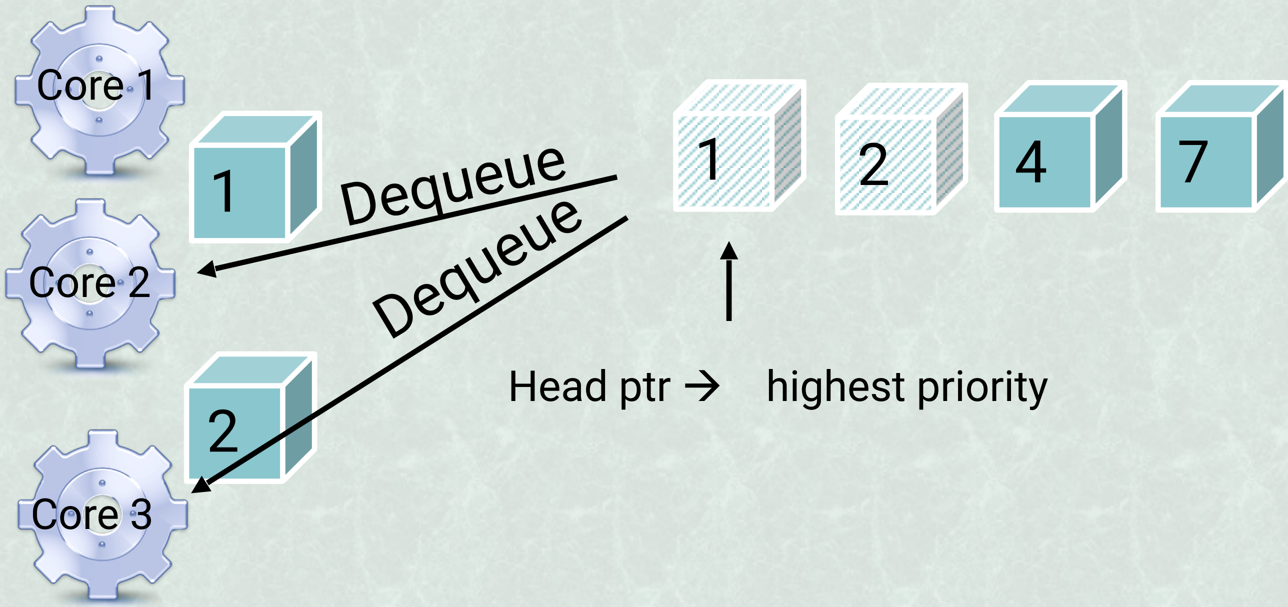


In many applications, relaxed priorities don't harm correctness

Graph applications, discrete event simulation, ...

Relaxed PQ

Relaxed Priority → alleviate synchronization at the dequeues



In many applications, relaxed priorities don't harm correctness

Graph applications, discrete event simulation, ...

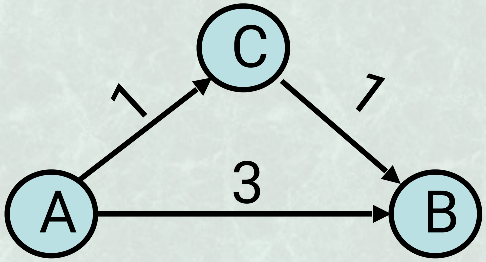
Relaxed PQ: Synch. vs Wasted Work

Relaxed PQ doesn't enforce ordered execution of tasks

Lower priority task can be overwritten by a higher priority one

Example: Single-Source Shortest Path (SSSP)

Vertex	Distance
A	∞
B	∞
C	∞



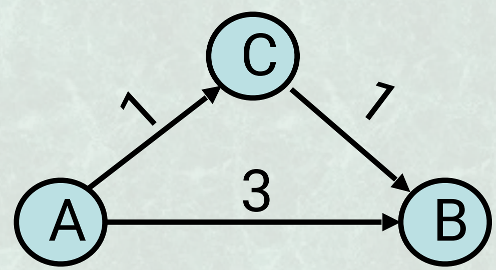
Relaxed PQ: Synch. vs Wasted Work

Relaxed PQ doesn't enforce ordered execution of tasks

Lower priority task can be overwritten by a higher priority one

Example: Single-Source Shortest Path (SSSP)

Vertex	Distance
A	∞
B	∞
C	∞



A, 0

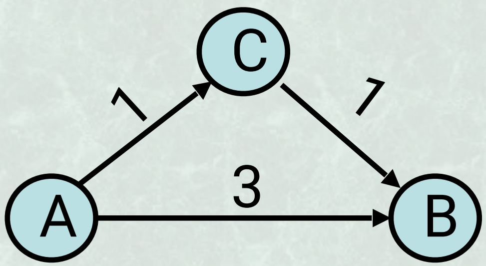
Relaxed PQ: Synch. vs Wasted Work

Relaxed PQ doesn't enforce ordered execution of tasks

Lower priority task can be overwritten by a higher priority one

Example: Single-Source Shortest Path (SSSP)

Vertex	Distance
A	0
B	∞
C	∞



~~A, 0~~ C, 1 B, 3

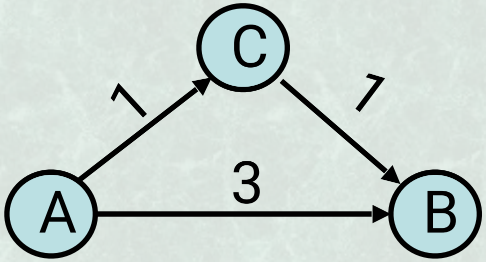
Relaxed PQ: Synch. vs Wasted Work

Relaxed PQ doesn’t enforce ordered execution of tasks

Lower priority task can be overwritten by a higher priority one

Example: Single-Source Shortest Path (SSSP)

Vertex	Distance
A	0
B	∞
C	1



~~A, 0~~ ~~C, 1~~ B, 2 B, 3

ILLINOIS

TEL AVIV UNIVERSITY אוניברסיטת תל אביב

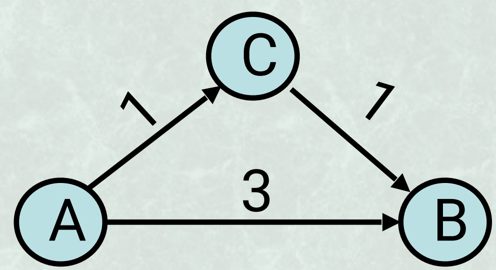
Relaxed PQ: Synch. vs Wasted Work

Relaxed PQ doesn't enforce ordered execution of tasks

Lower priority task can be overwritten by a higher priority one

Example: Single-Source Shortest Path (SSSP)

Vertex	Distance
A	0
B	3
C	1



~~A, 0~~ ~~C, 1~~ B, 2 ~~B, 3~~

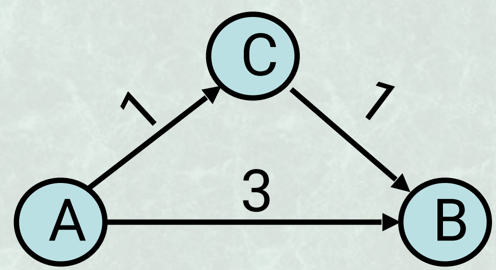
Relaxed PQ: Synch. vs Wasted Work

Relaxed PQ doesn't enforce ordered execution of tasks

Lower priority task can be overwritten by a higher priority one

Example: Single-Source Shortest Path (SSSP)

Vertex	Distance
A	0
B	~~3~~ 2
C	1

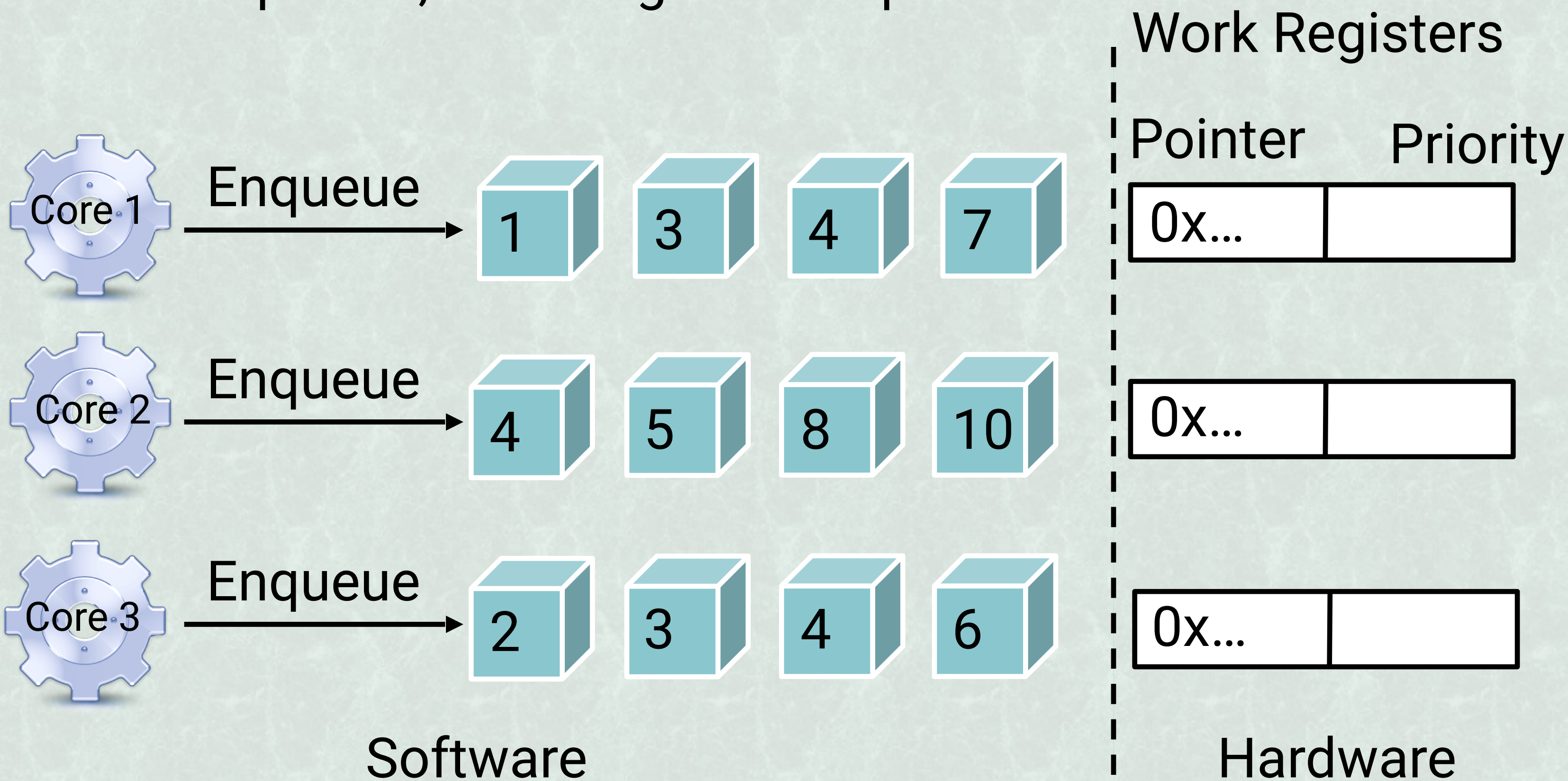


~~A, 0~~ ~~C, 1~~ ~~B, 2~~ ~~B, 3~~

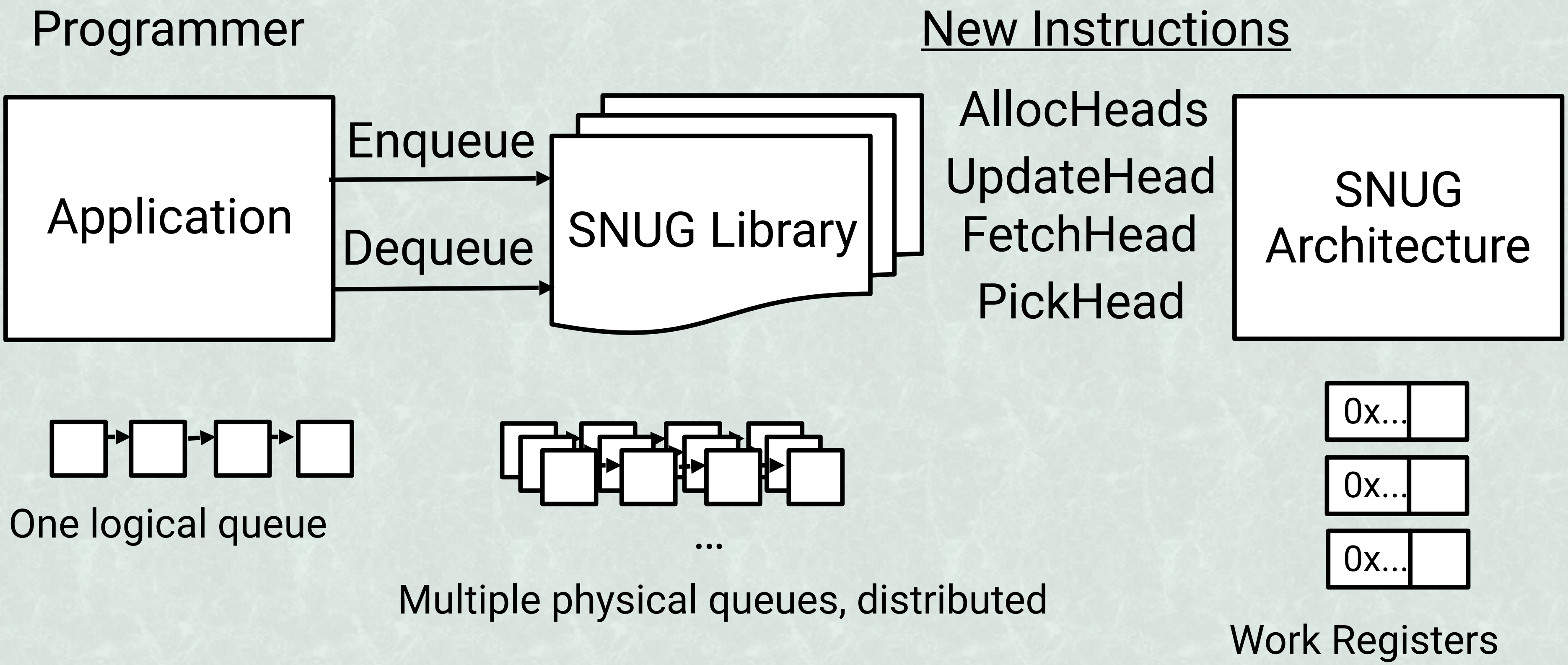
Overview of SNUG

Wasted work vs. synchronization trade-off

Local enqueues, relaxed global dequeues

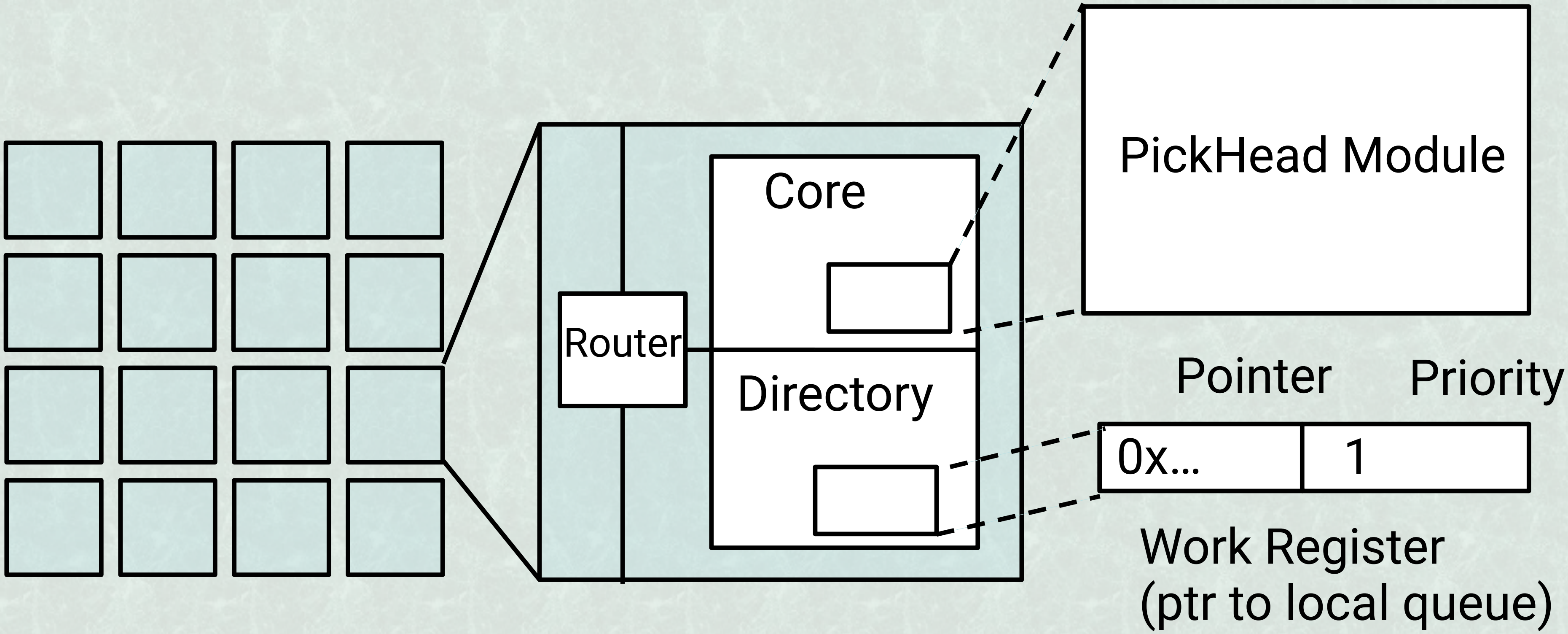


SNUG Programming Model

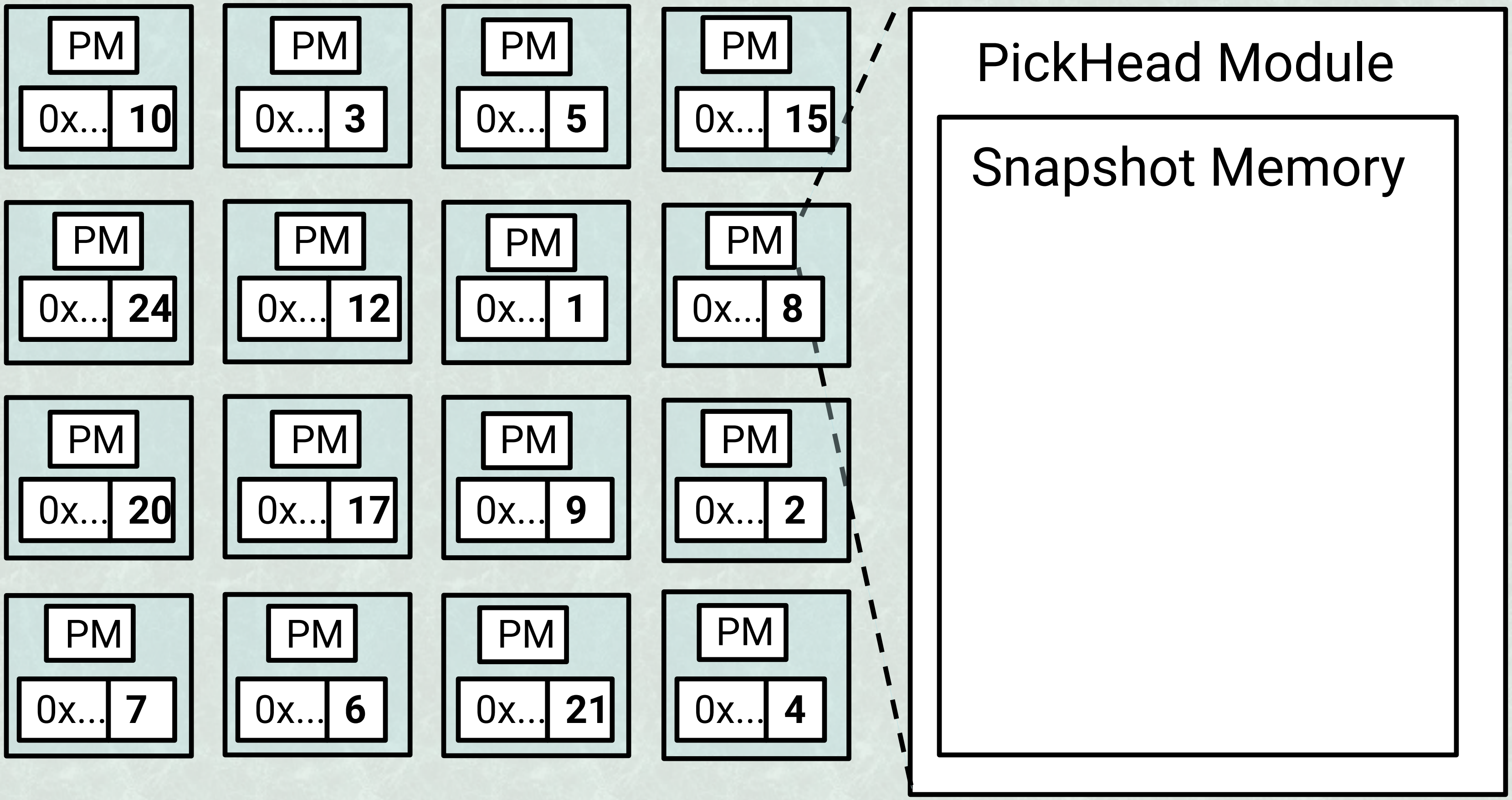


SNUG Architecture

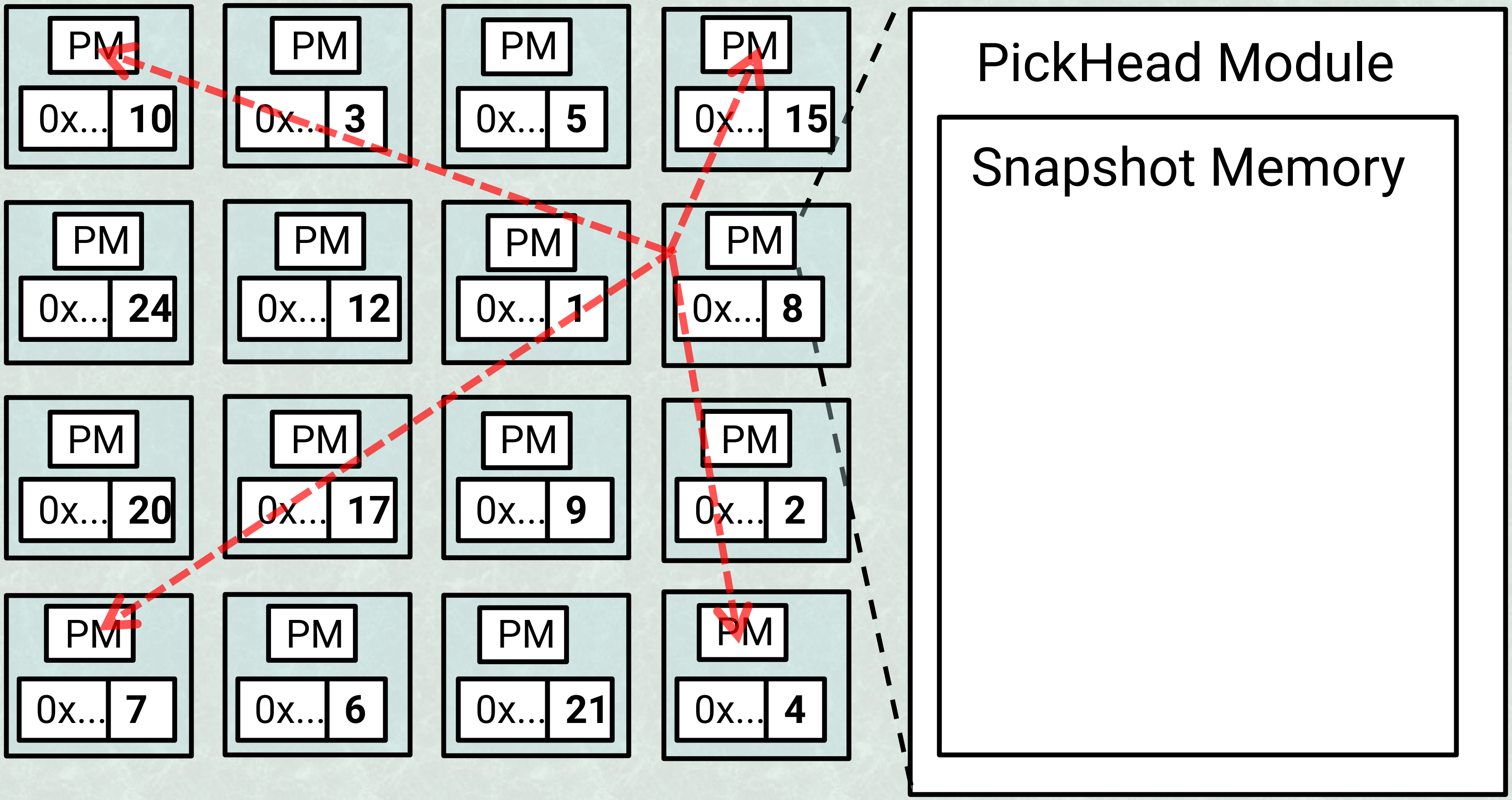
Each core has access to a set of work registers and a PickHead module



PickHead Instruction

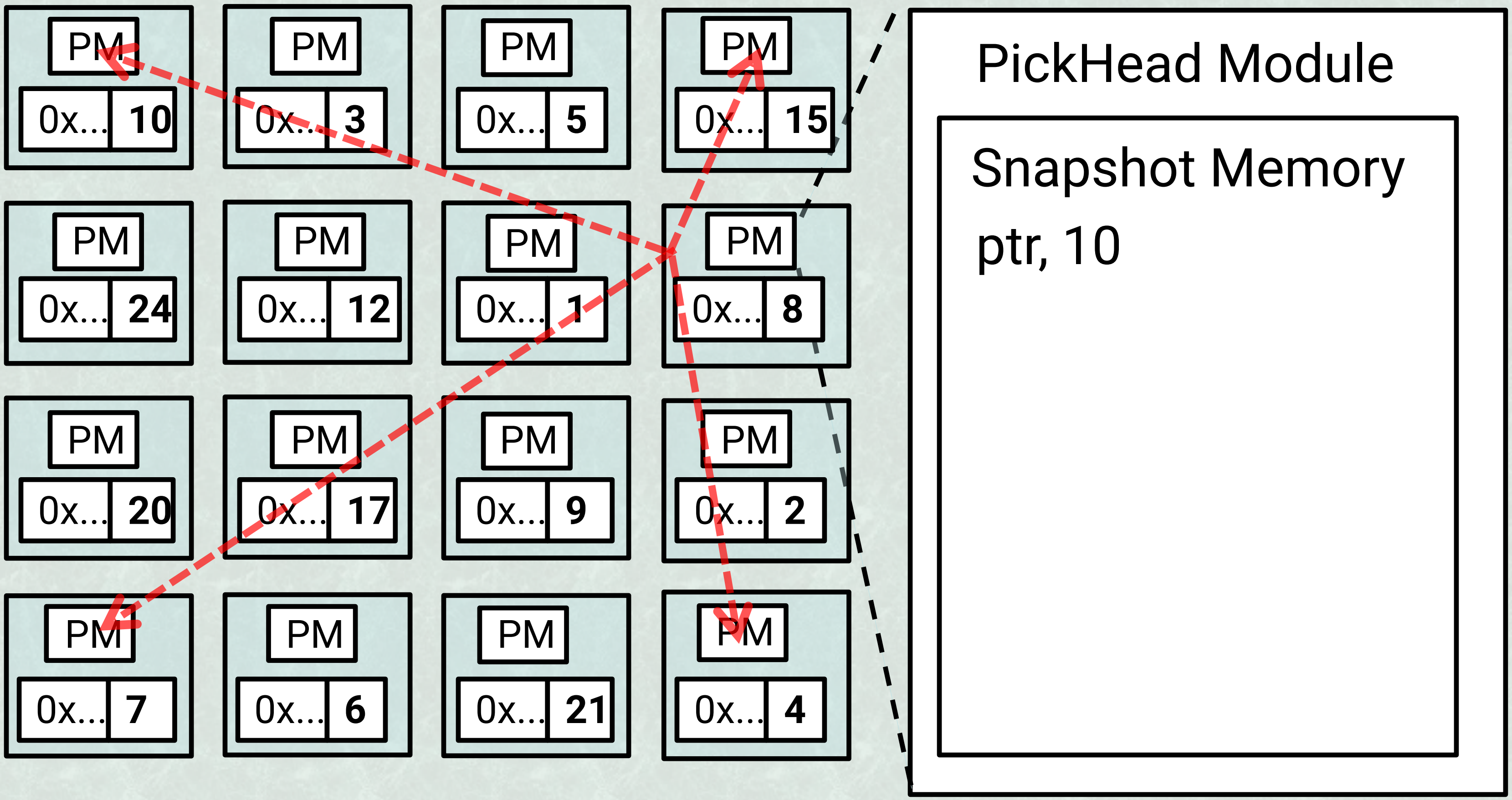


PickHead Instruction



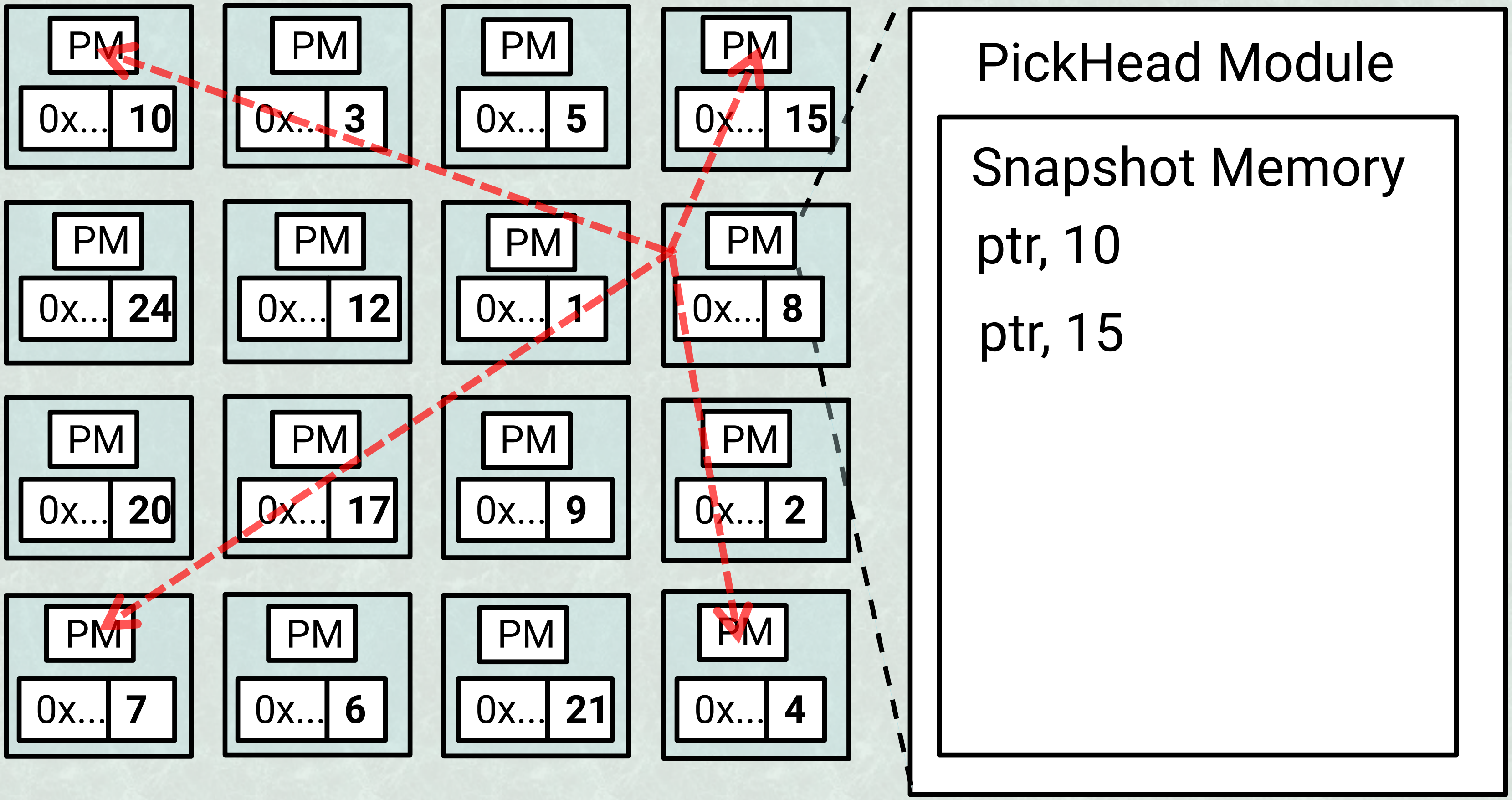
Gather ptrs to queue heads + priorities

PickHead Instruction



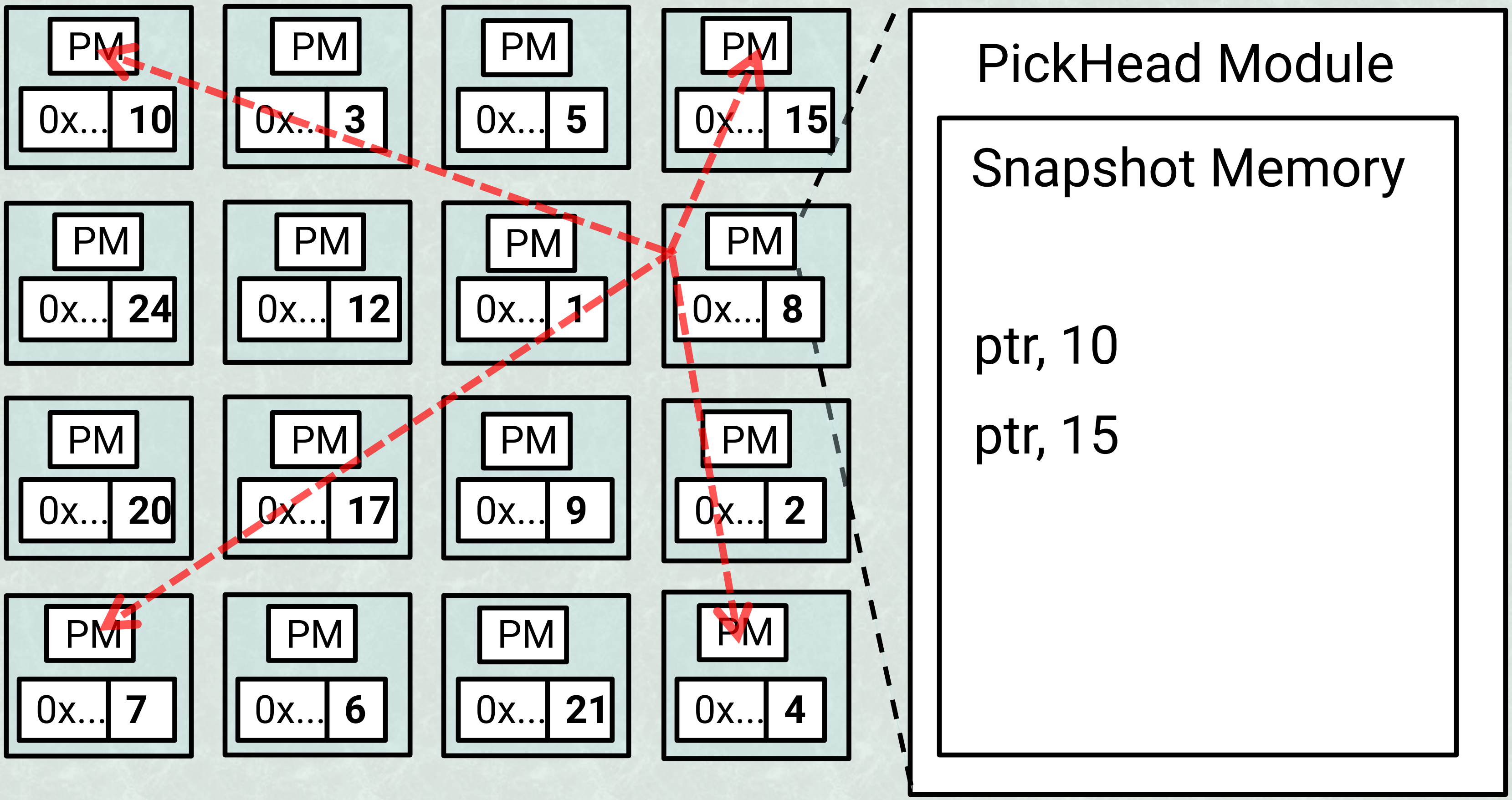
Gather ptrs to queue heads + priorities

PickHead Instruction



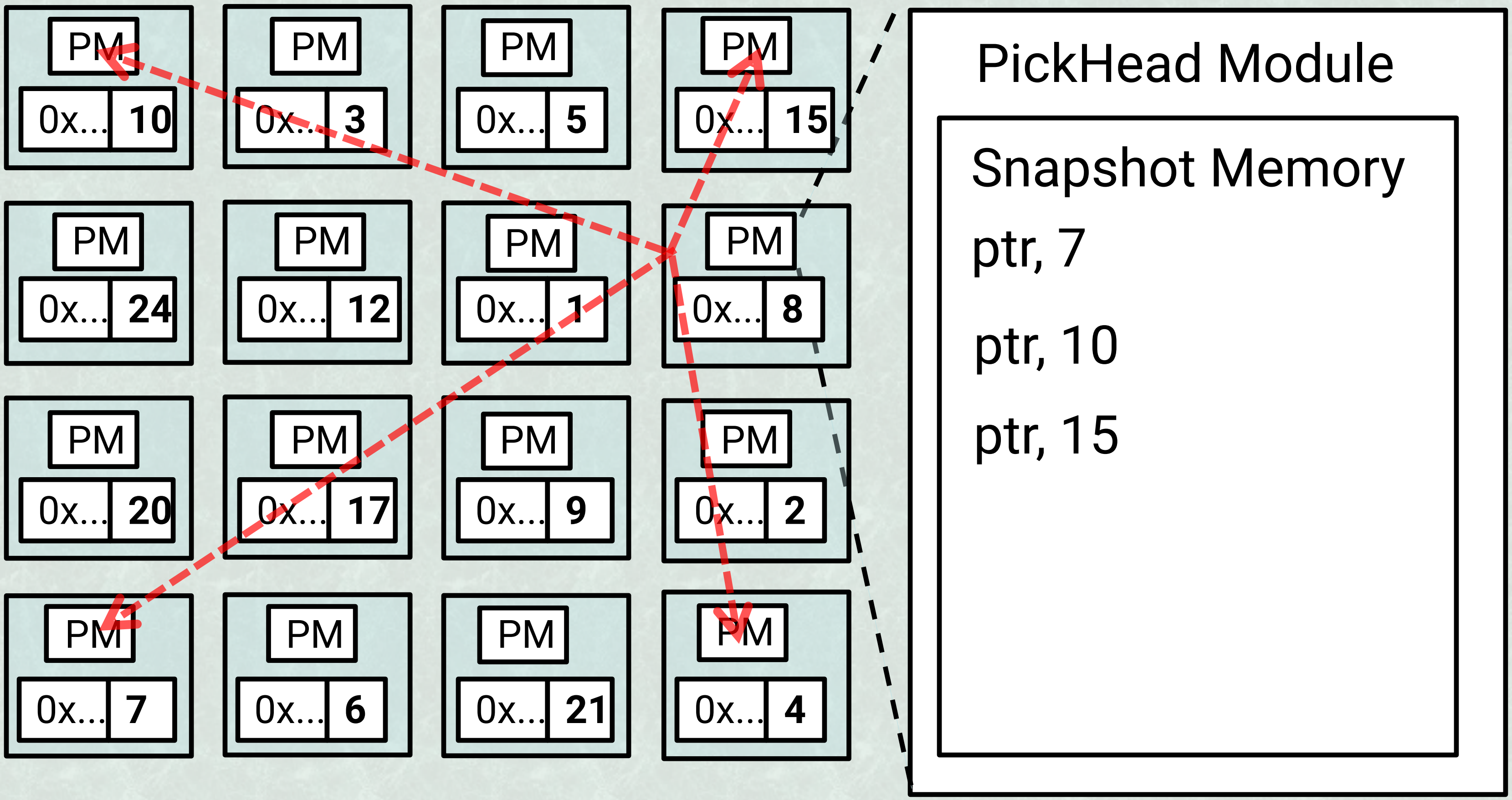
Gather ptrs to queue heads + priorities

PickHead Instruction



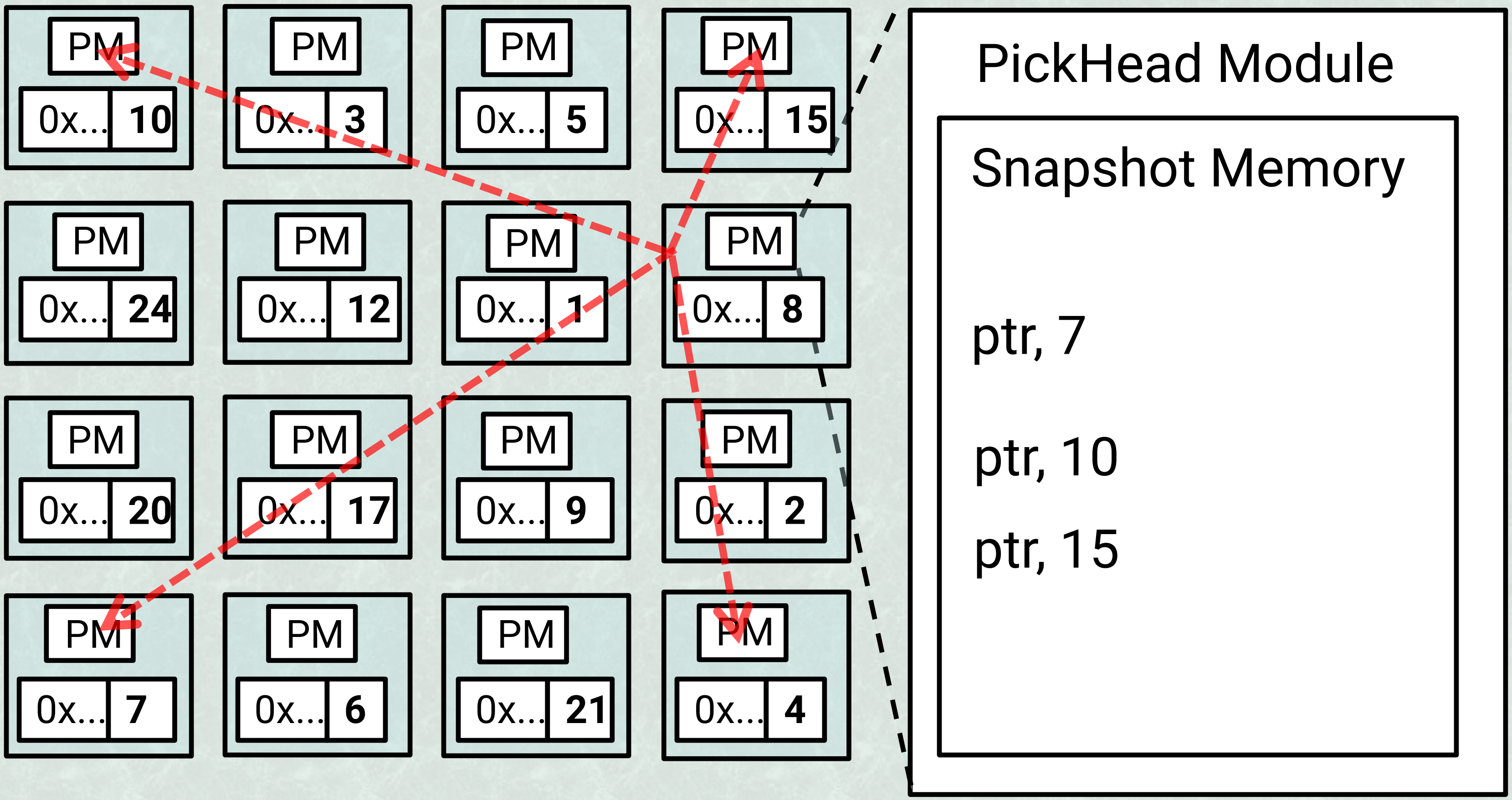
Gather ptrs to queue heads + priorities

PickHead Instruction



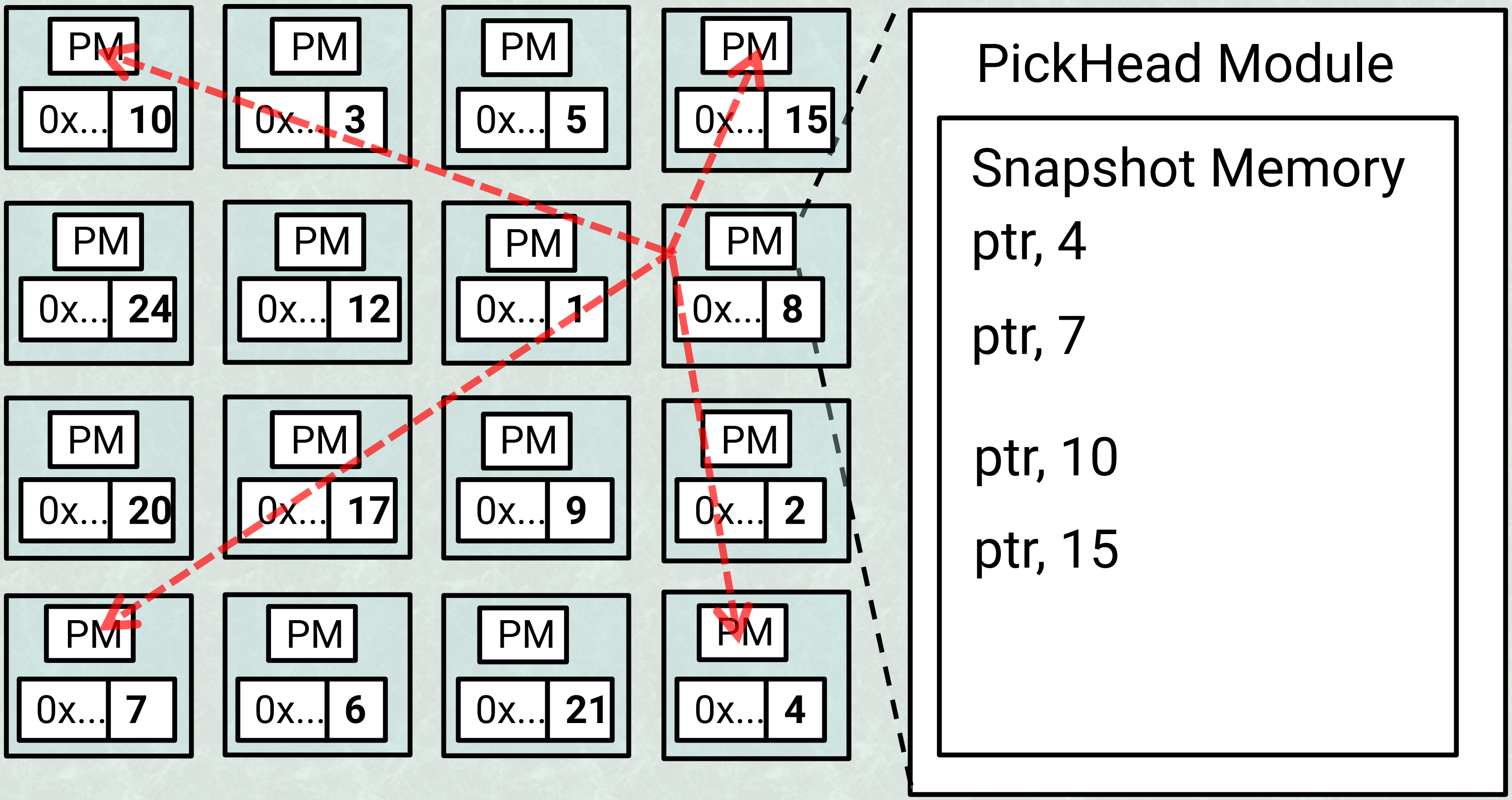
Gather ptrs to queue heads + priorities

PickHead Instruction



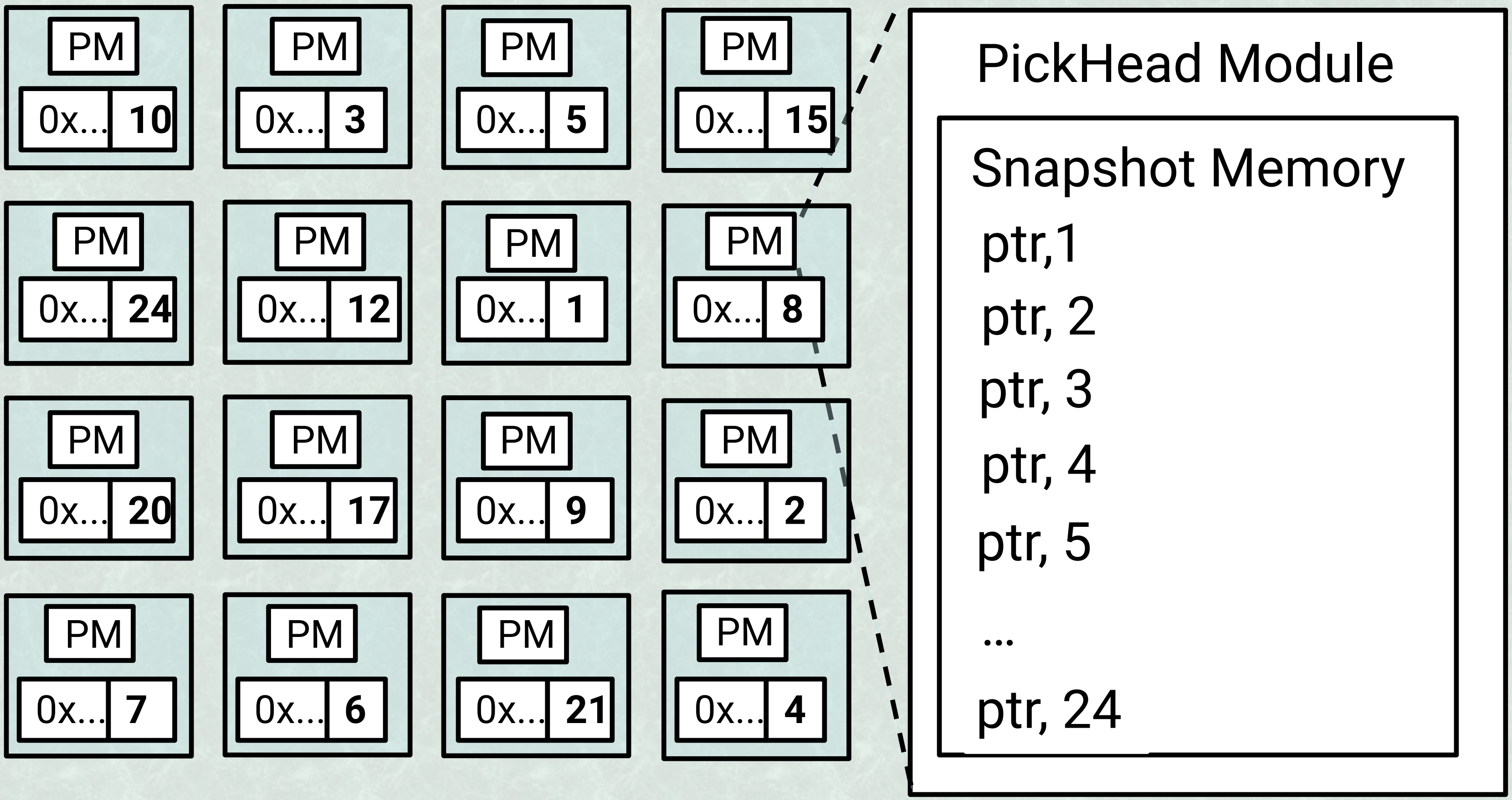
Gather ptrs to queue heads + priorities

PickHead Instruction



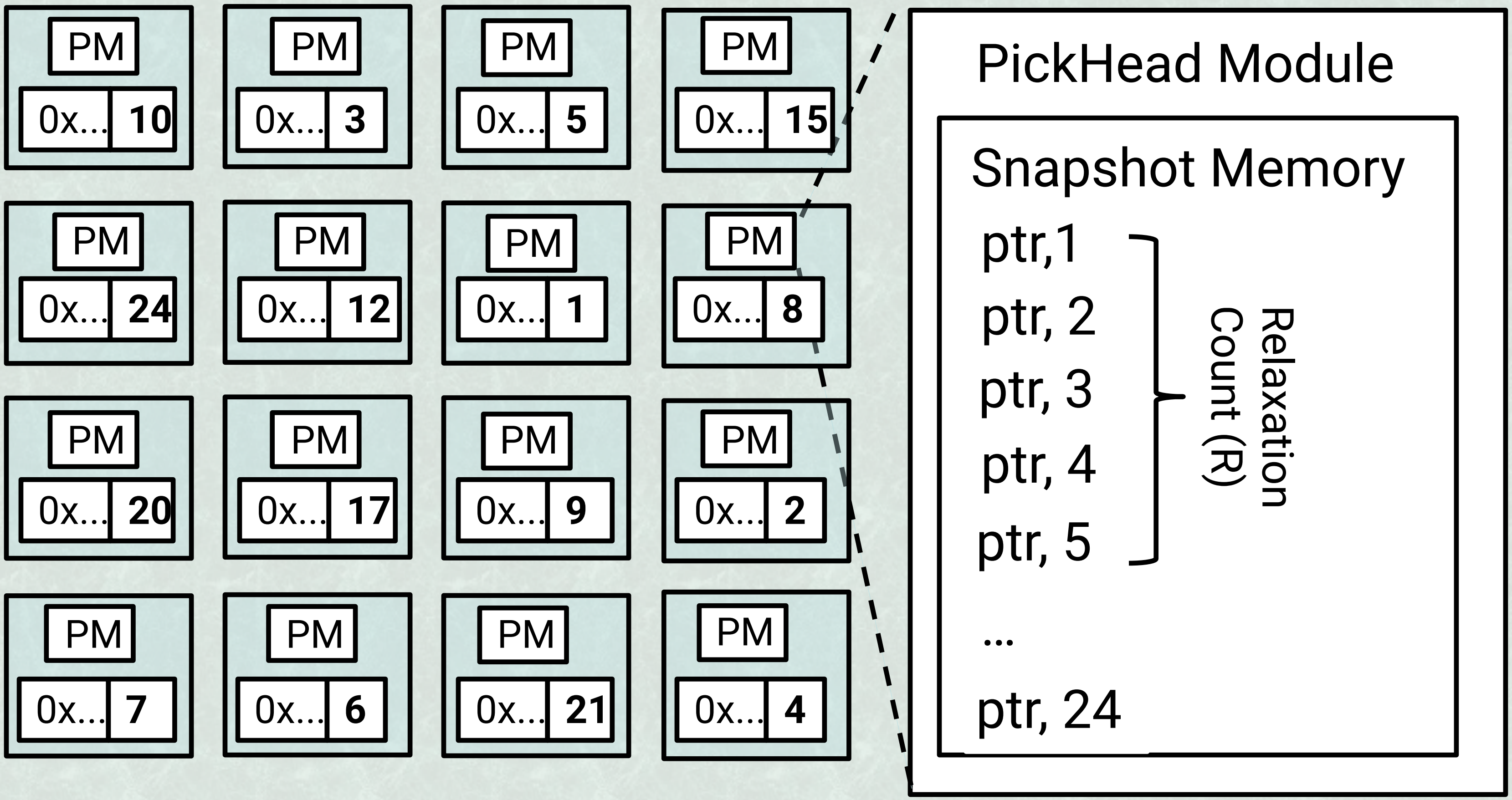
Gather ptrs to queue heads + priorities

PickHead Instruction



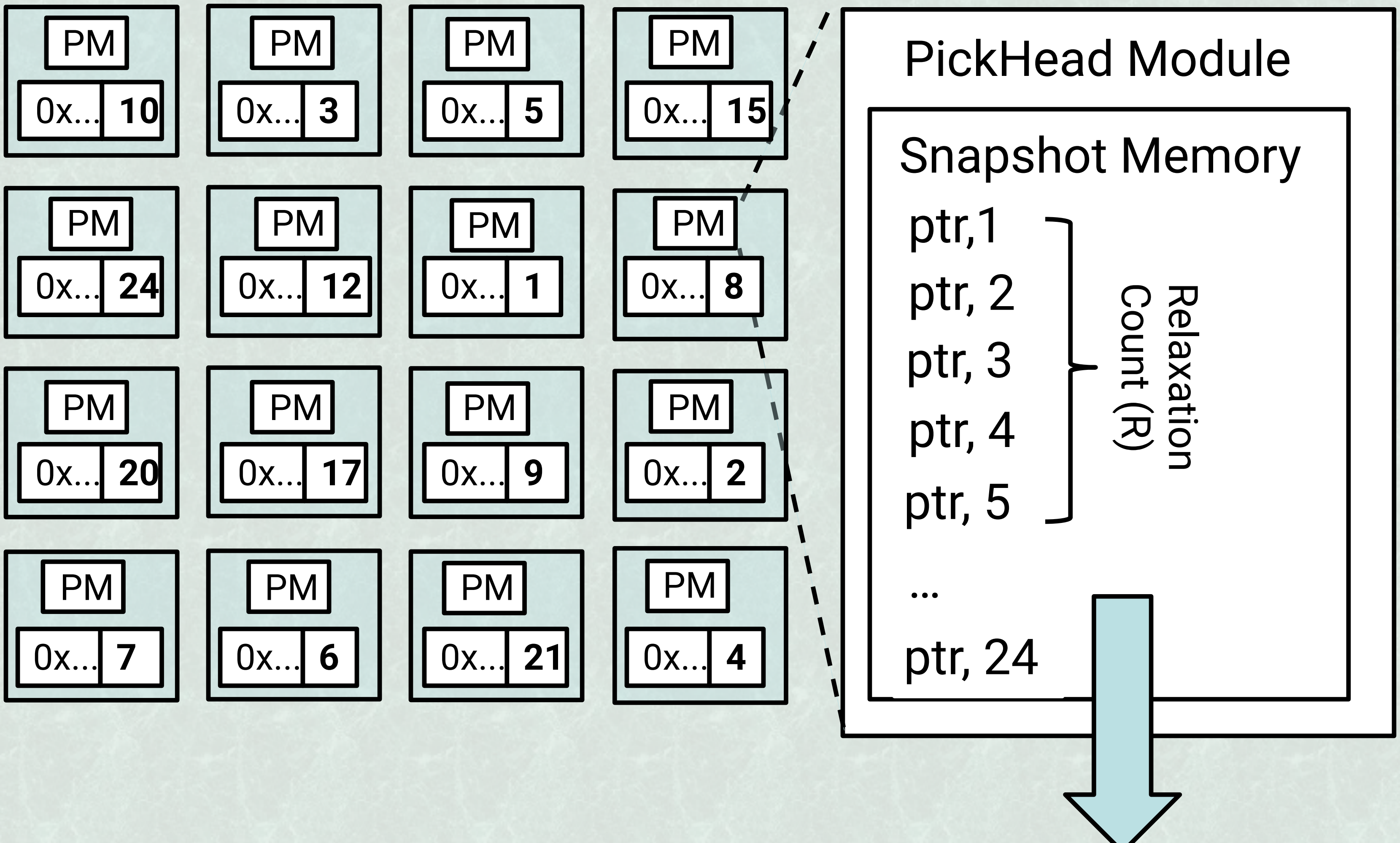
Sorting heads of physical queues

PickHead Instruction



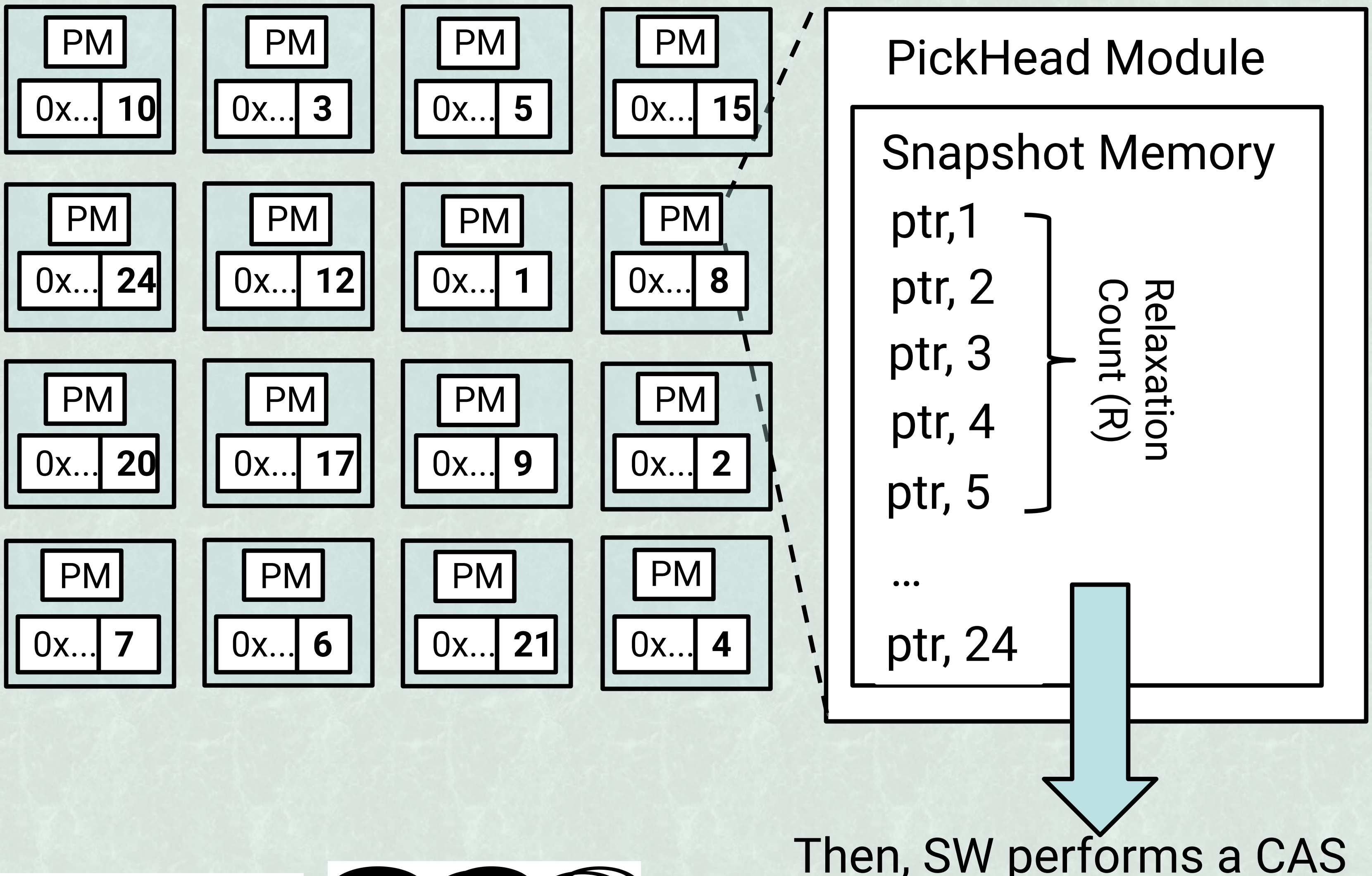
Sorting heads of physical queues

PickHead Instruction



Return one priority at random

PickHead Instruction



ILLINOIS

TEL AVIV UNIVERSITY אוניברסיטת תל אביב

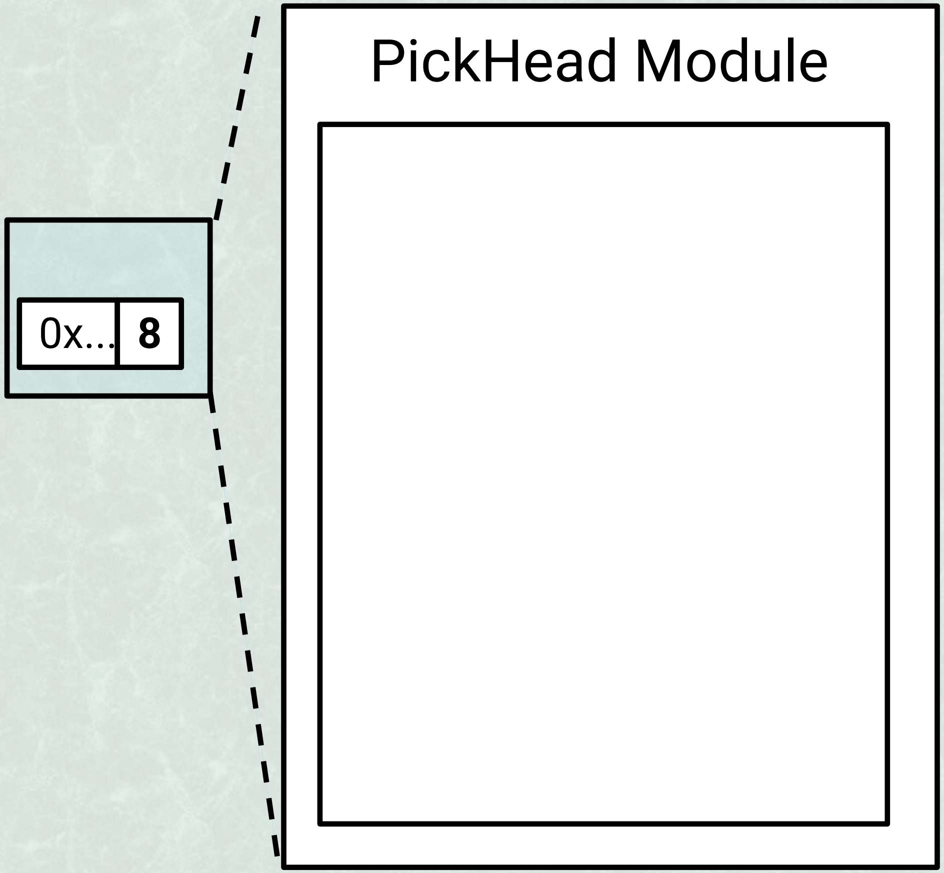
Modes of PickHead Instruction

Global Access: gathers the heads of all queues

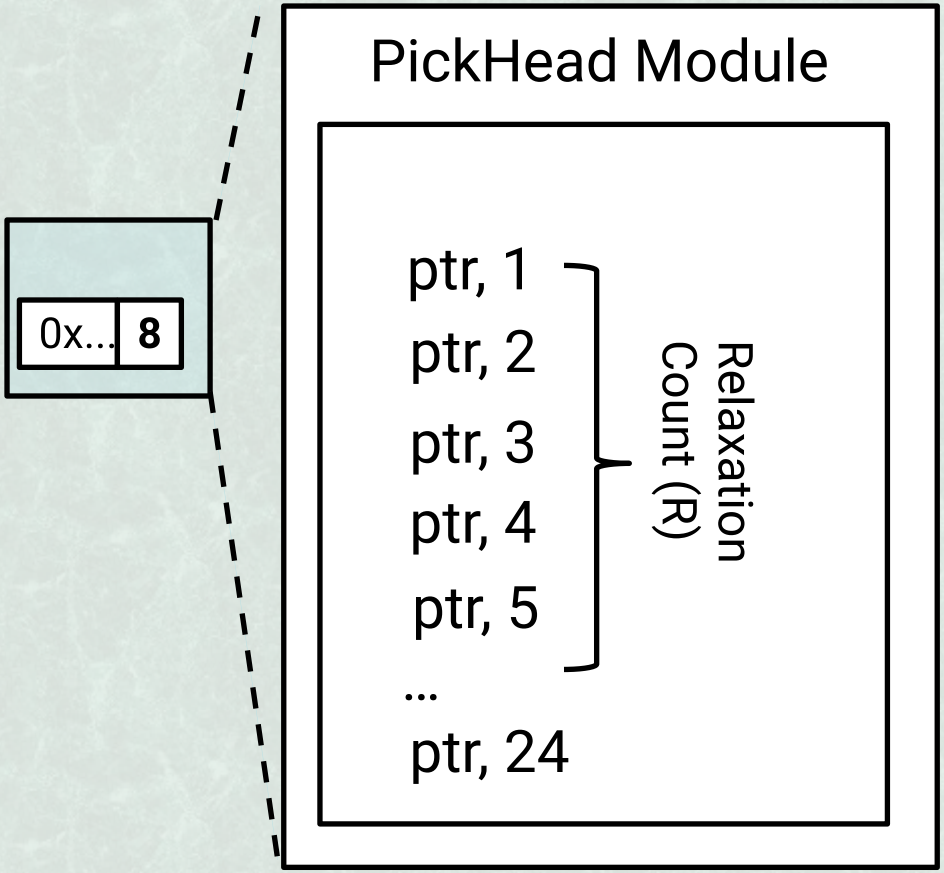
No Network Access: reuses the snapshot before it gets stale

Local Access: uses the head of the local queue

Modes of PickHead Instruction

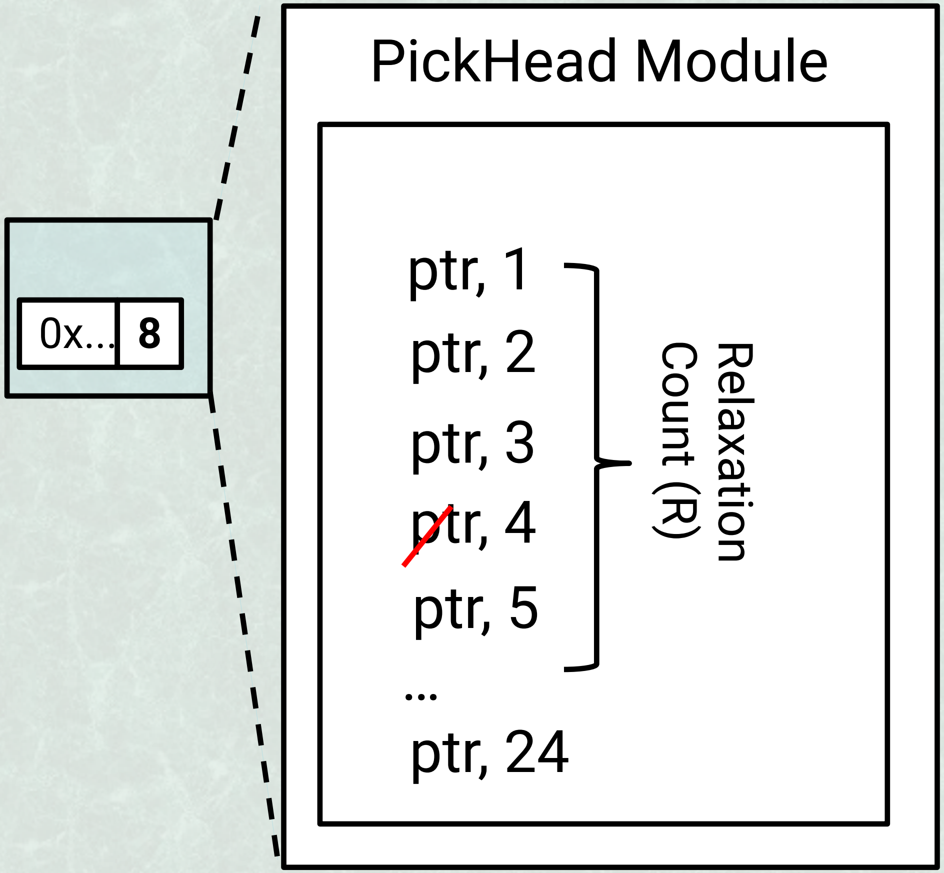


Modes of PickHead Instruction



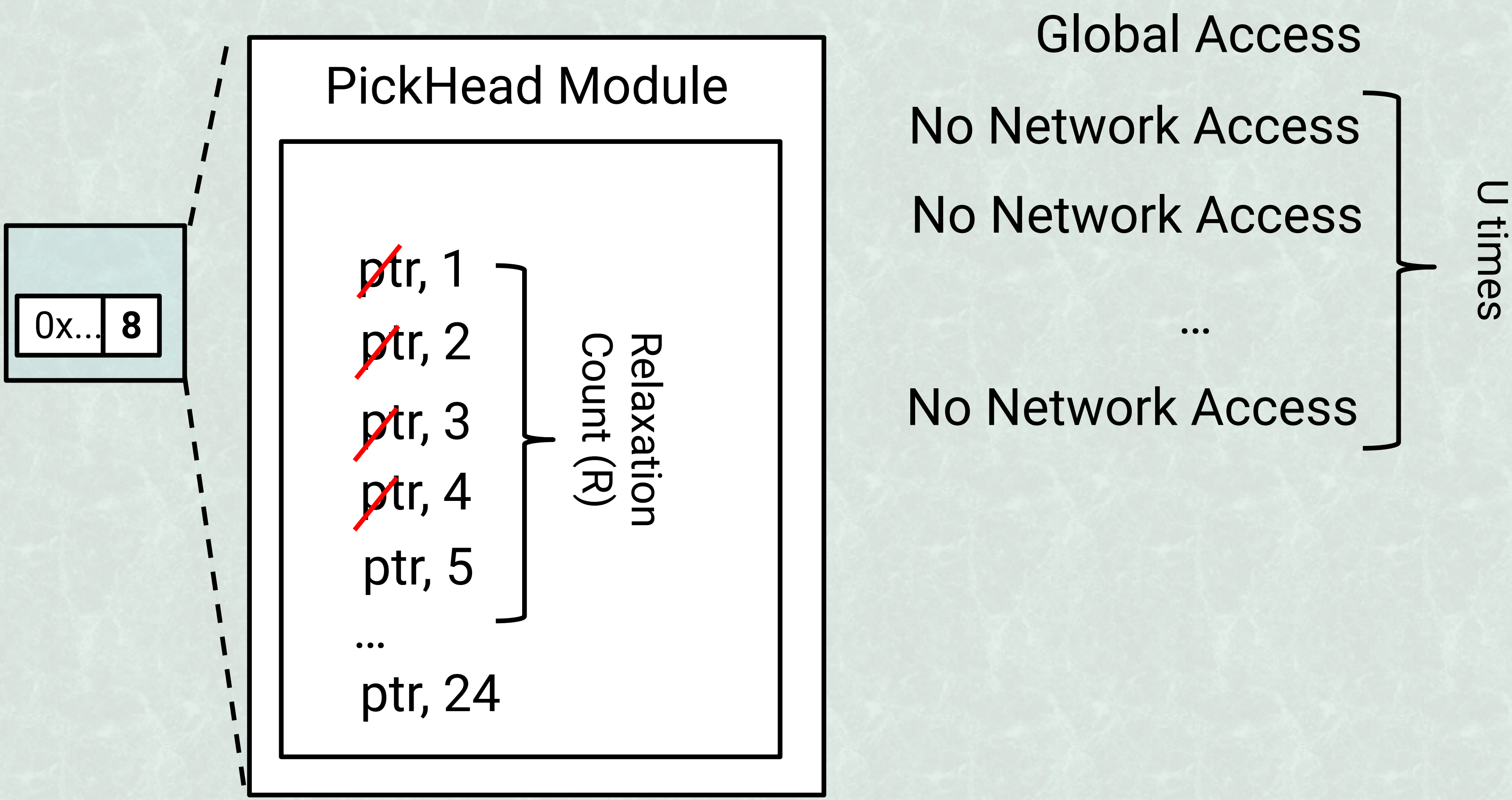
Global Access

Modes of PickHead Instruction

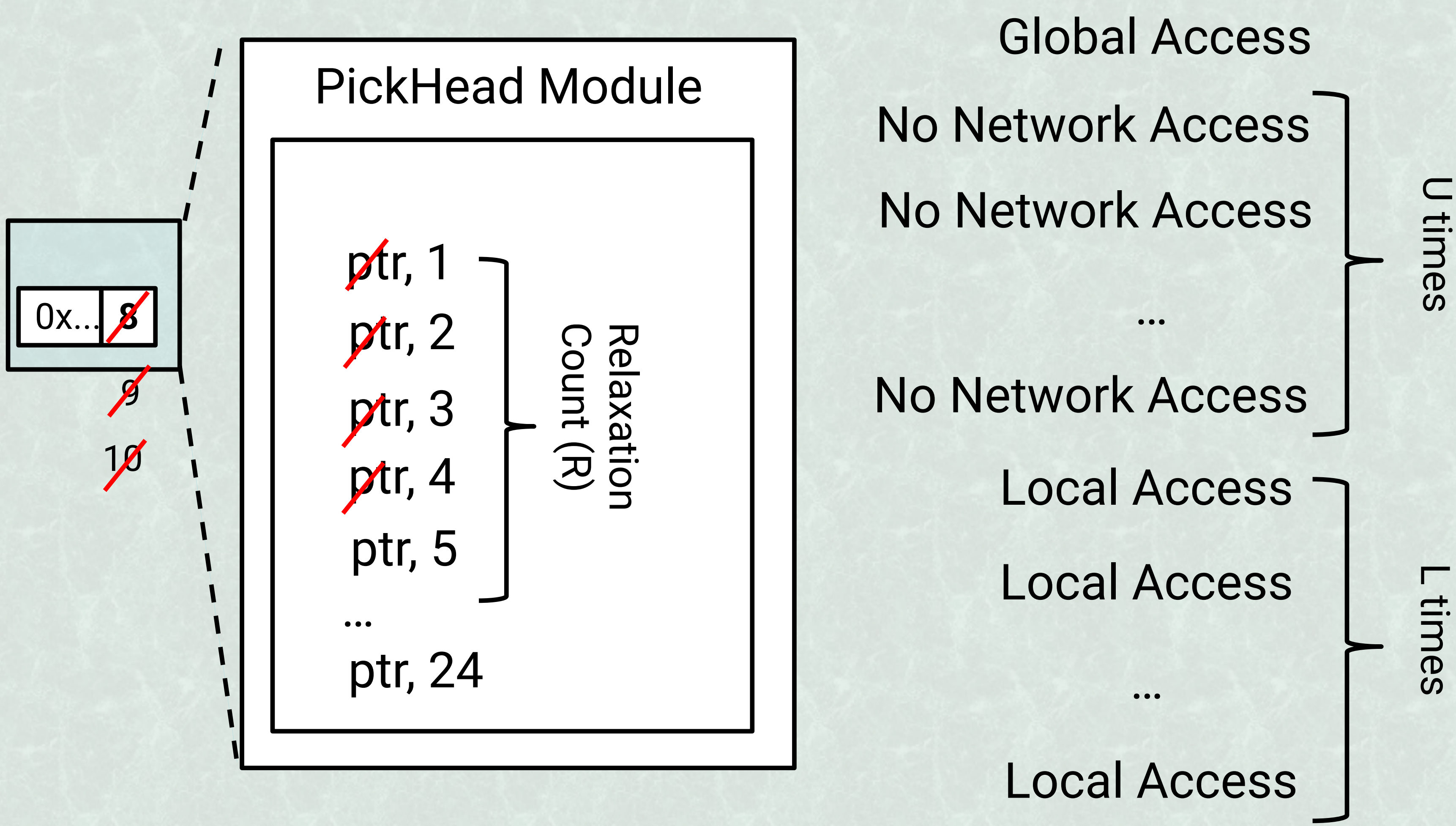


Global Access

Modes of PickHead Instruction



Modes of PickHead Instruction



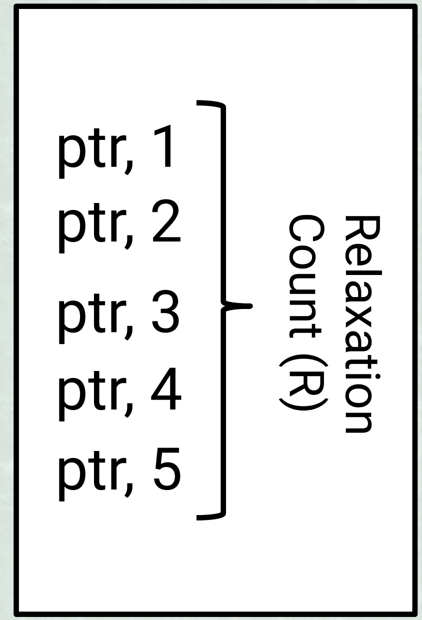
Adaptivity of SNUG to CAS Failures

SW performs a CAS on the chosen queue. It can fail.

If failure: retry the same queue

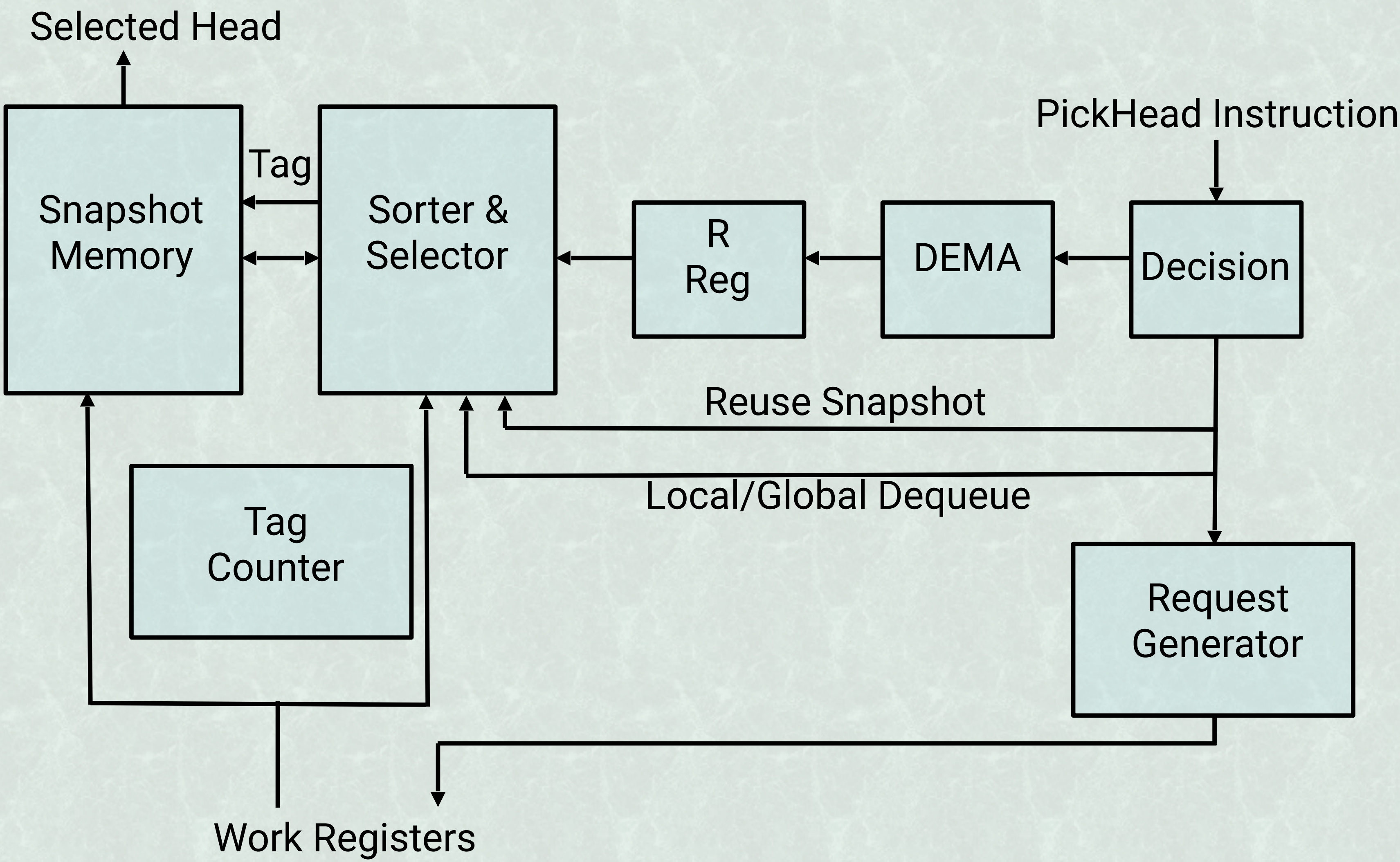
SNUG adapts to contention

- ➢If frequent CAS failures → increase R↑
- ➢If rare CAS failures → decrease R↓



TEL AVIV UNIVERSITY אוניברסיטת תל אביב

PickHead Module



Evaluation Setup

64-core simulations in Gem5, U = 4 and L = 4
Applications: SSSP, BFS, SIMUL, A*

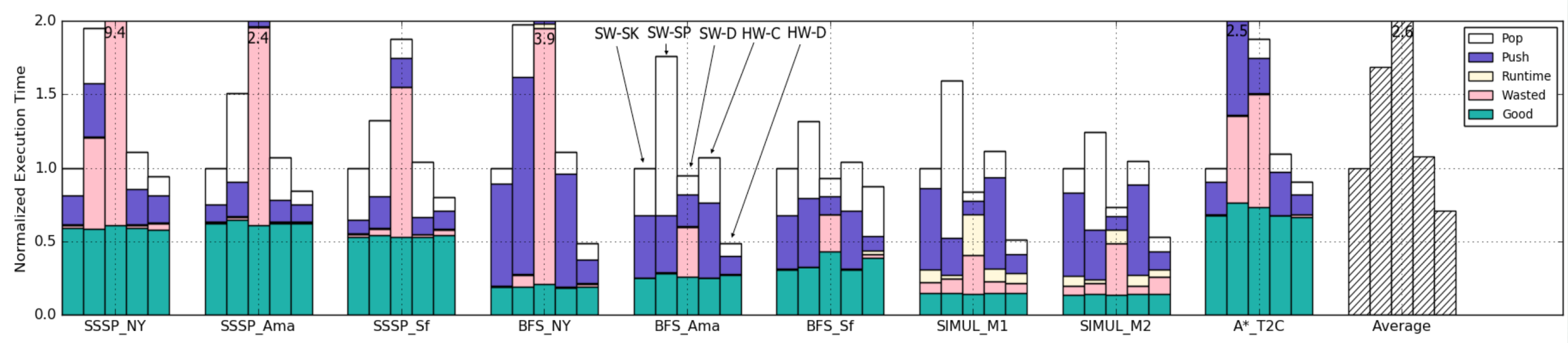
PQs evaluated:

- SW-SK: Concurrent skiplist[1] implementation (baseline)
    - Always dequeues the highest priority task
- SW-SP: Concurrent spraylist[2]
    - Dequeues are sprayed over a range of high priority tasks
- SW-D: Distributed concurrent skiplist with work-stealing
    - Local enqueues, local dequeues
- HW-C: Centralized version of SNUG. No PickHead
- HW-D: SNUG
    - Local enqueues, global dequeues

[1] Skip Lists: A probabilistic Alternative to Balanced Trees, CACM 33, 1990
[2] The SprayList, A Scalable Relaxed Priority Queue, PPoPP, 2015

ILLINOIS TEL AVIV UNIVERSITY אוניברסיטת תל אביב

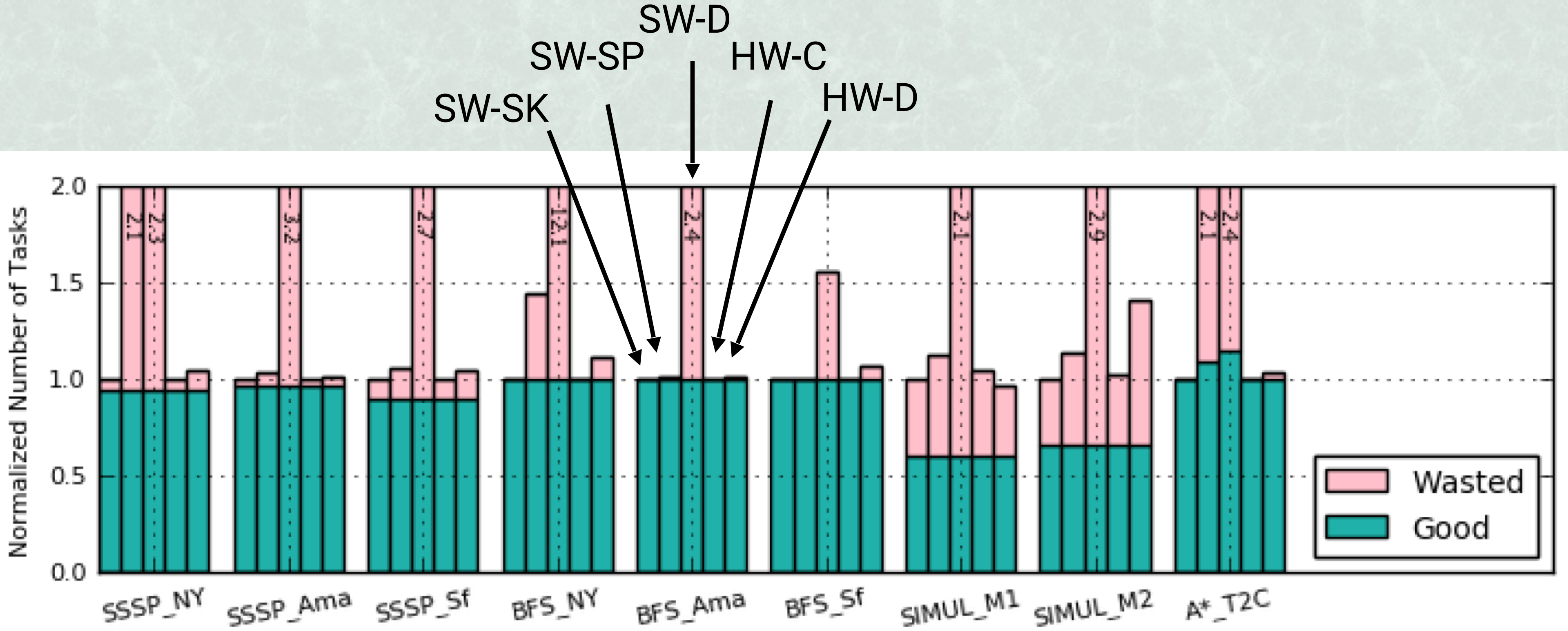
Execution Time



Push and pop contention in SW-SK and SW-SP. Wasted work in SW-D

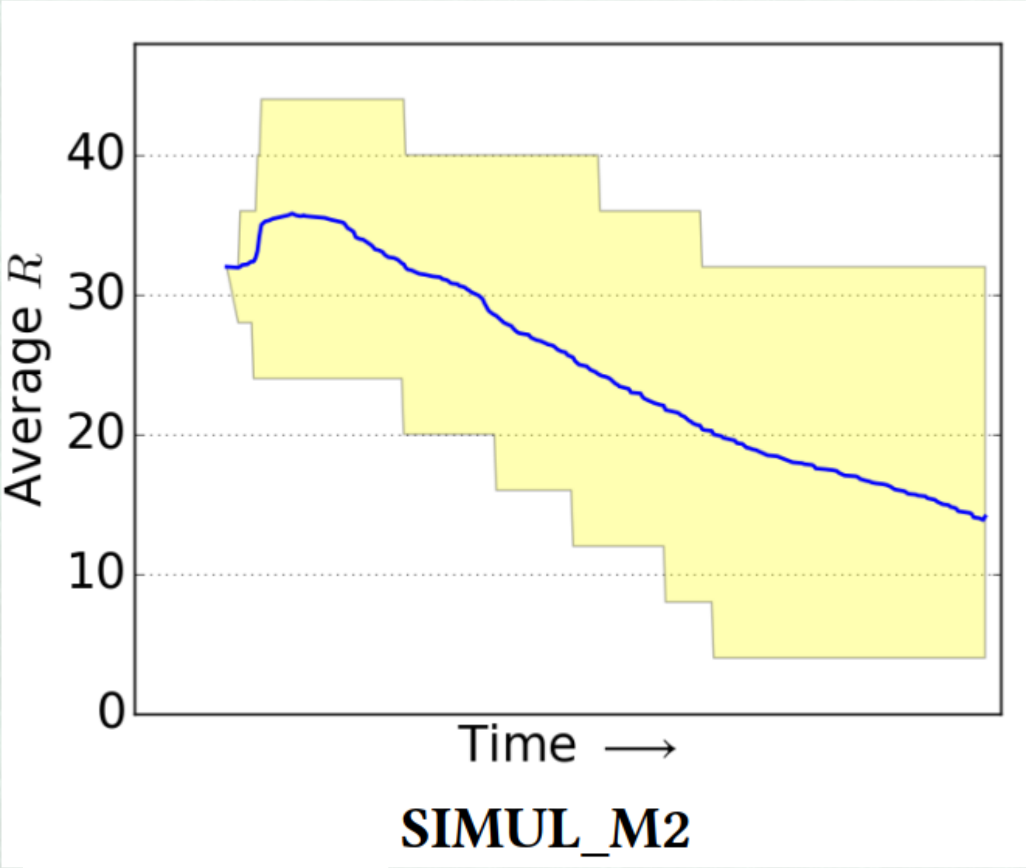
SNUG achieves 1.4x, 2.4x, and 3.6x speedup over SW-SK, SW-SP and SW-D

Breakdown of Tasks



SW-D suffers from wasted work due to local dequeues

Adaptation of the Relaxation Count



R is able to adapt to the CAS contentions

More in the Paper

- Analysis of Network Traffic
- SNUG Scalability
- Sensitivity Analysis of SNUG Parameters
- Characterizing R Adaptivity
- Power and Area Analysis
- …

ILLINOIS
TEL AVIV UNIVERSITY אוניברסיטת תל אביב

Key Takeaways

- Relaxed PQ alleviates synchronization overhead but is prone to wasted work.
- SNUG distributes PQ in hardware and minimizes both wasted work and synchronization overhead simultaneously.
- SNUG's relaxed PQ adapts to the rate of synchronization failures over time.
- For 64 cores: SNUG achieves avg. speedup of 1.4x, 2.4x, and 3.6x speedup over skip PQ, spray PQ, and distributed concurrent skiplist.

Thank You!

SNUG: Architectural Support for Relaxed Concurrent Priority Queueing in Chip Multiprocessors

Azin Heidarshenas*, Tanmay Gangwani*, Serif Yesil, Adam Morrison, and Josep Torrellas

* Co-first authorship

ICS

International Conference on Supercomputing 2020

ILLINOIS

TEL AVIV UNIVERSITY אוניברסיטת תל אביב