

NUMA Memory Management

CS592, Fall 2026

Reminder: Sign up for presentations!

Goals for today

- A classical NUMA memory management paper
- Today's picture: Hardware, Linux NUMA

Operating System Support for Improving Data Locality on CC-NUMA Compute Servers

Ben Verghese, Scott Devine, Anoop Gupta, and Mendel Rosenblum

Computer Systems Laboratory, Stanford University, CA 94305

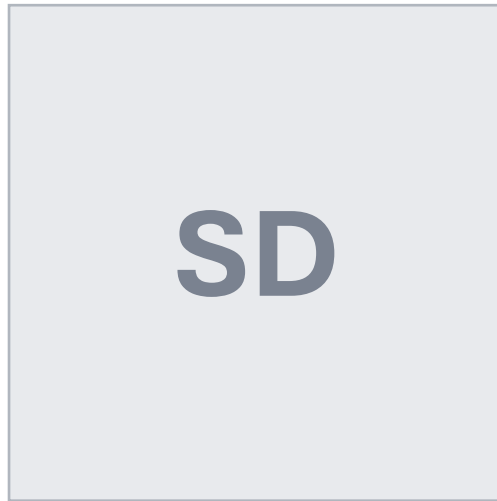
<http://www-flash.stanford.edu>

ASPLOS 1996

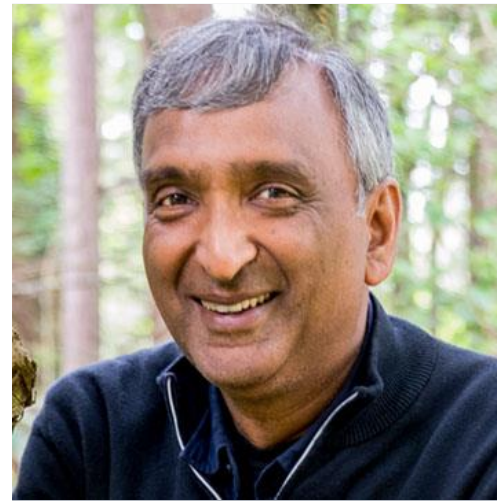
Paper authors



Ben Verghese



Scott Devine



Anoop Gupta

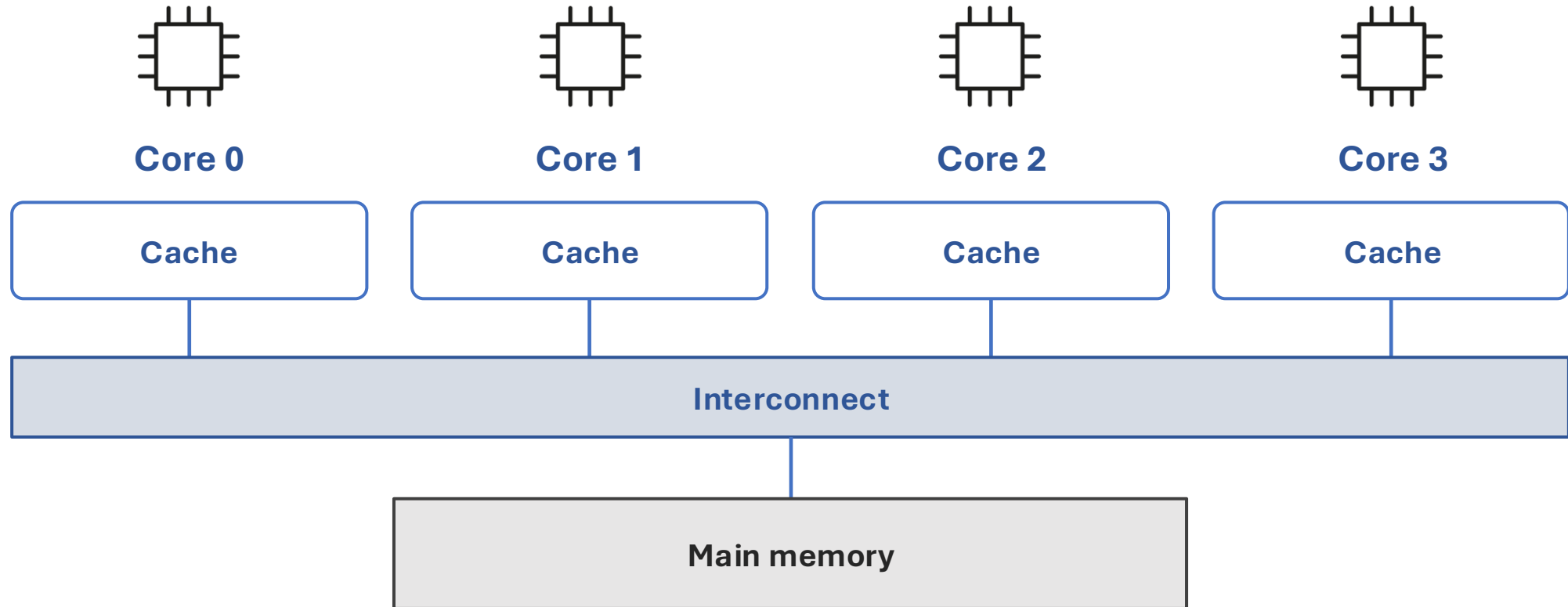


Mendel Rosenblum

Computer Systems Laboratory, Stanford University

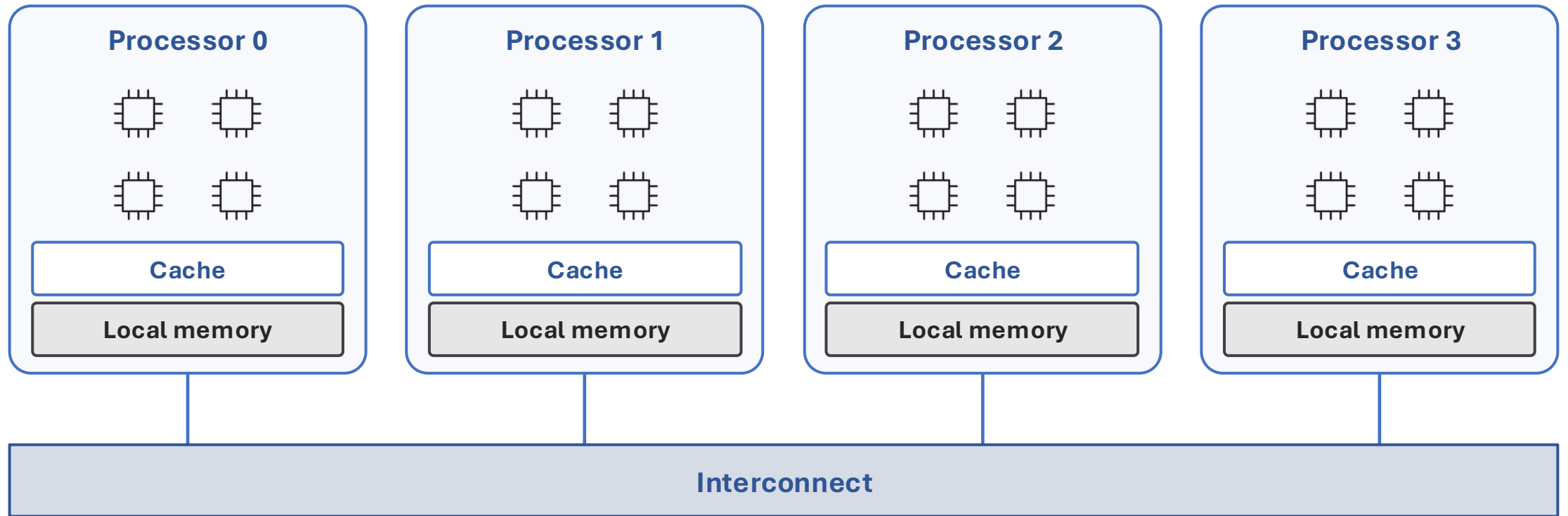
“Operating System Support for Improving Data Locality on CC-NUMA Compute Servers,” ASPLOS-VII, 1996

Cache-coherent (CC) shared memory



Each core may cache its own copy of shared data; (informally) coherence keeps all copies consistent.

CC-NUMA and CC-NOW



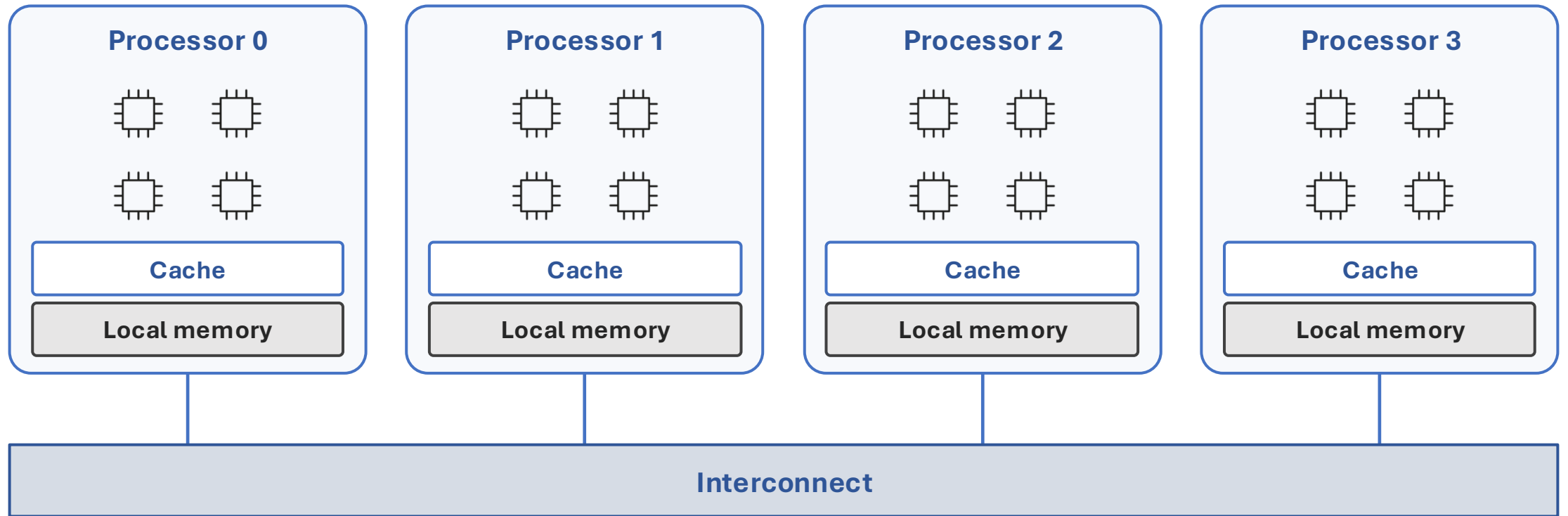
Coherence spans all processors; access to another processor's memory costs more than local access.

Aside: Interconnects

- Custom interconnects
 - Intel UPI
 - AMD Infinity Fabric
 - Nvidia NVLink
 - Infiniband
- General-purpose interconnects
 - Ethernet

Properties and performance

CC-NUMA and CC-NOW

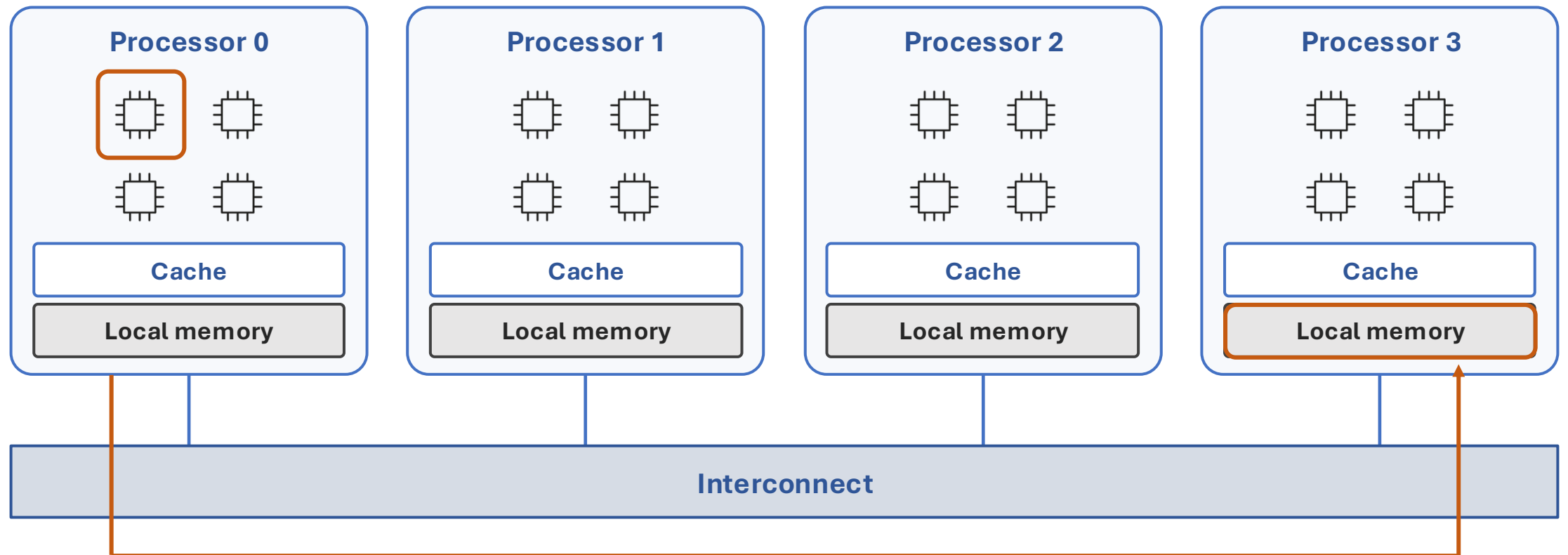


Coherence spans all processors; access to another processor's memory costs more than local access.

What problem is the paper trying to solve?

- “Our goal is to minimize the runtime for the user’s workload by reducing the component due to memory stall”

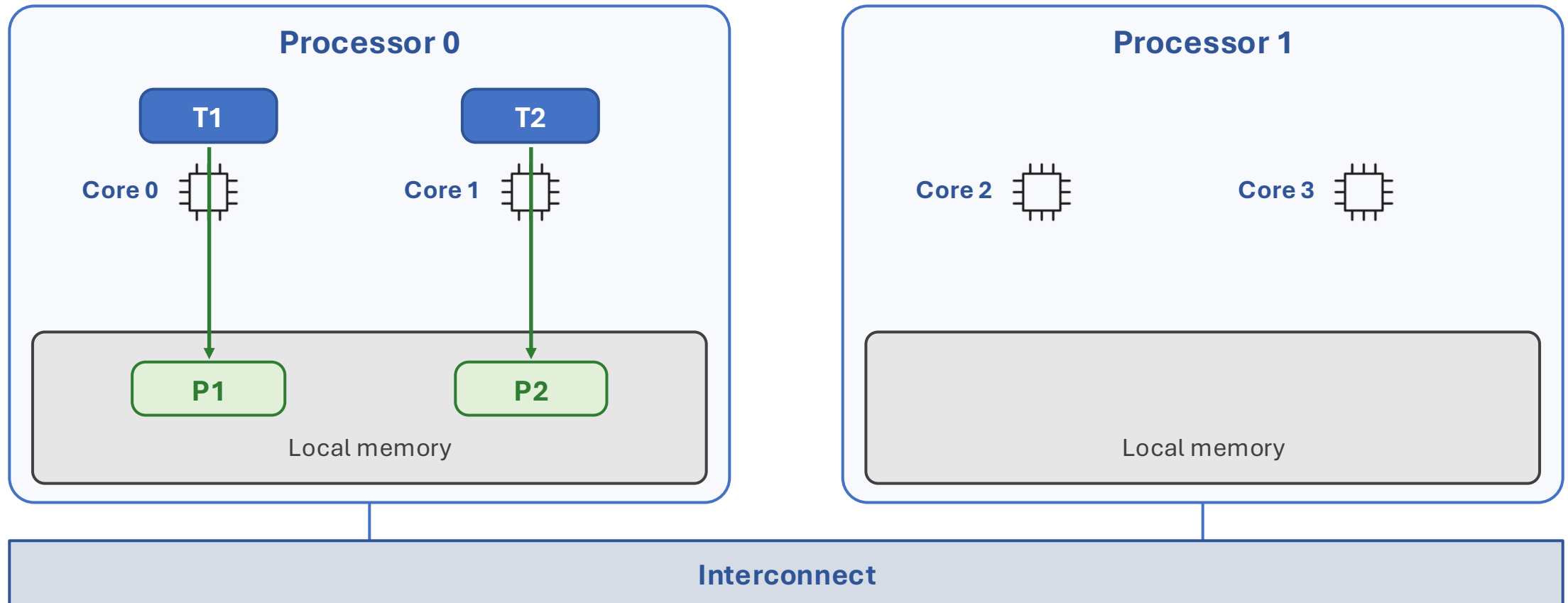
Remote memory access



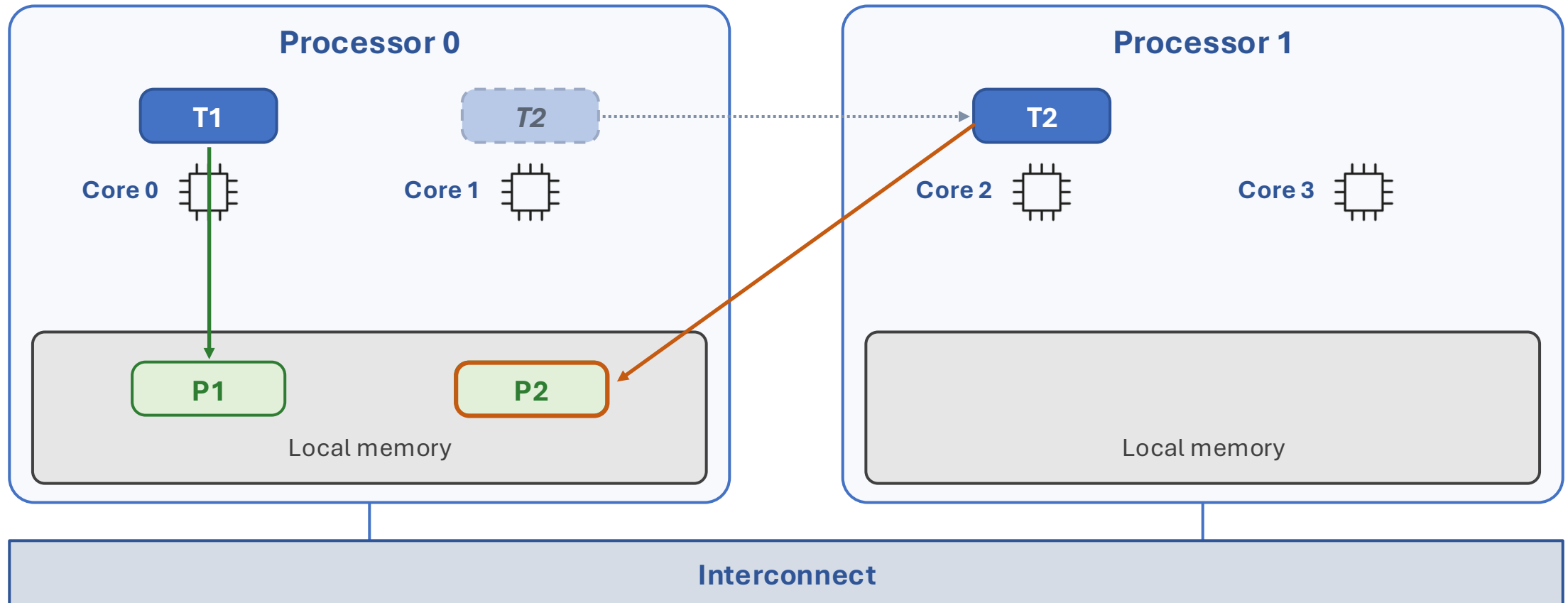
Remote memory access latency: 3-5x of local for CC-NUMA, 10-20x for CC-NOW

(Paper's expectations)

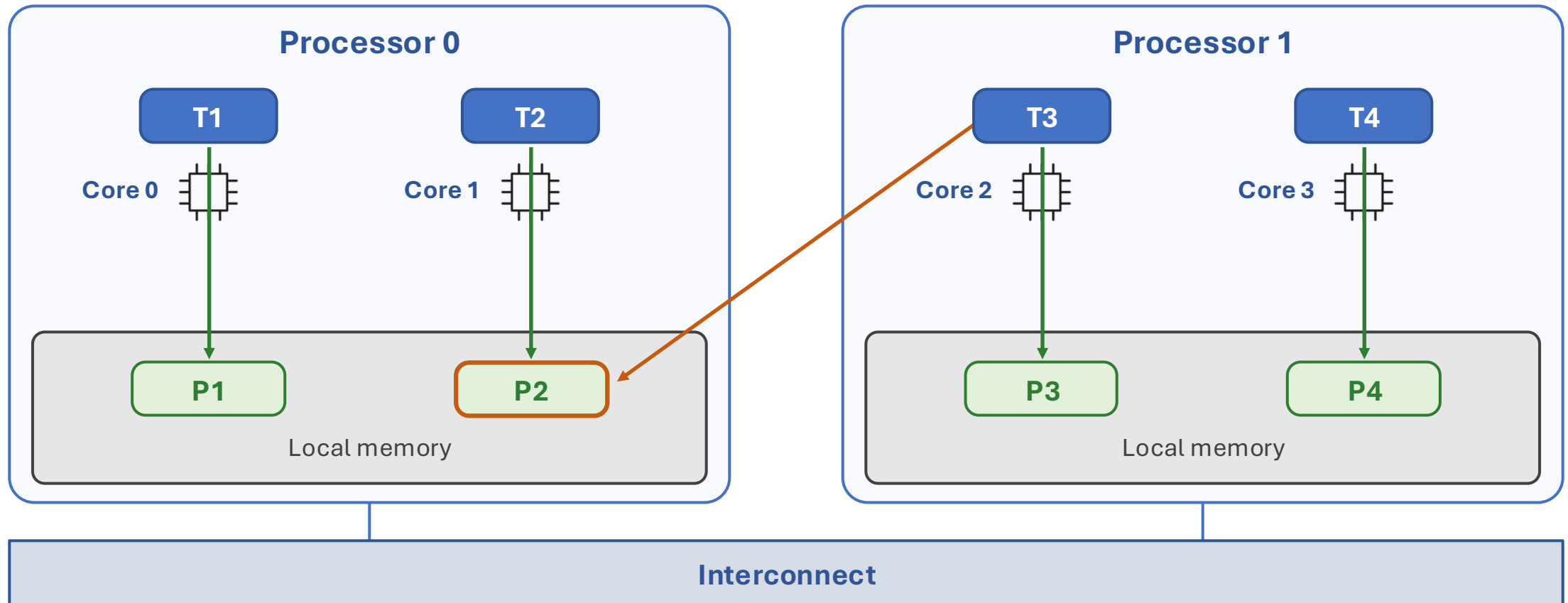
Allocate memory near threads?



Allocate memory near threads?



Allocate memory near threads?

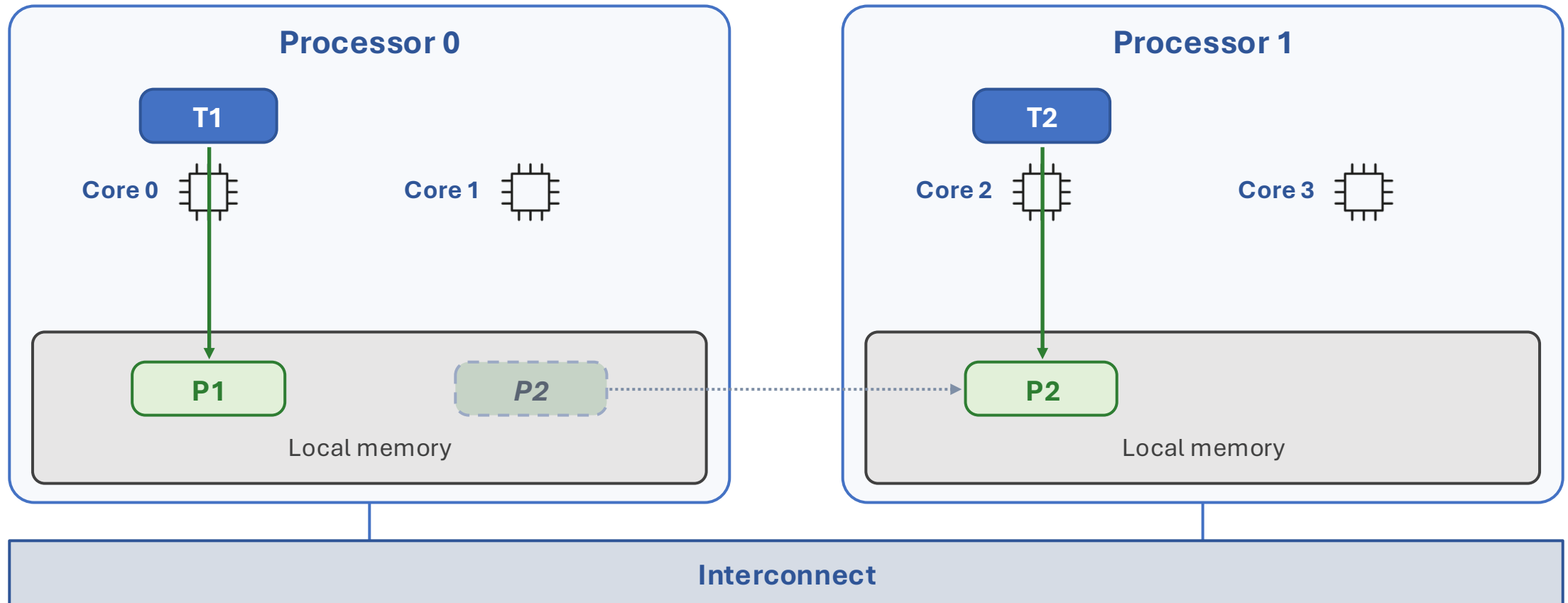


Mechanisms and policies

- Mechanisms: **Page migration, replication**, access tracking
- Policy
 - When to migrate or replicate?
 - Which pages to migrate or replicate?

Transparency enabled by virtual memory abstraction

Page migration



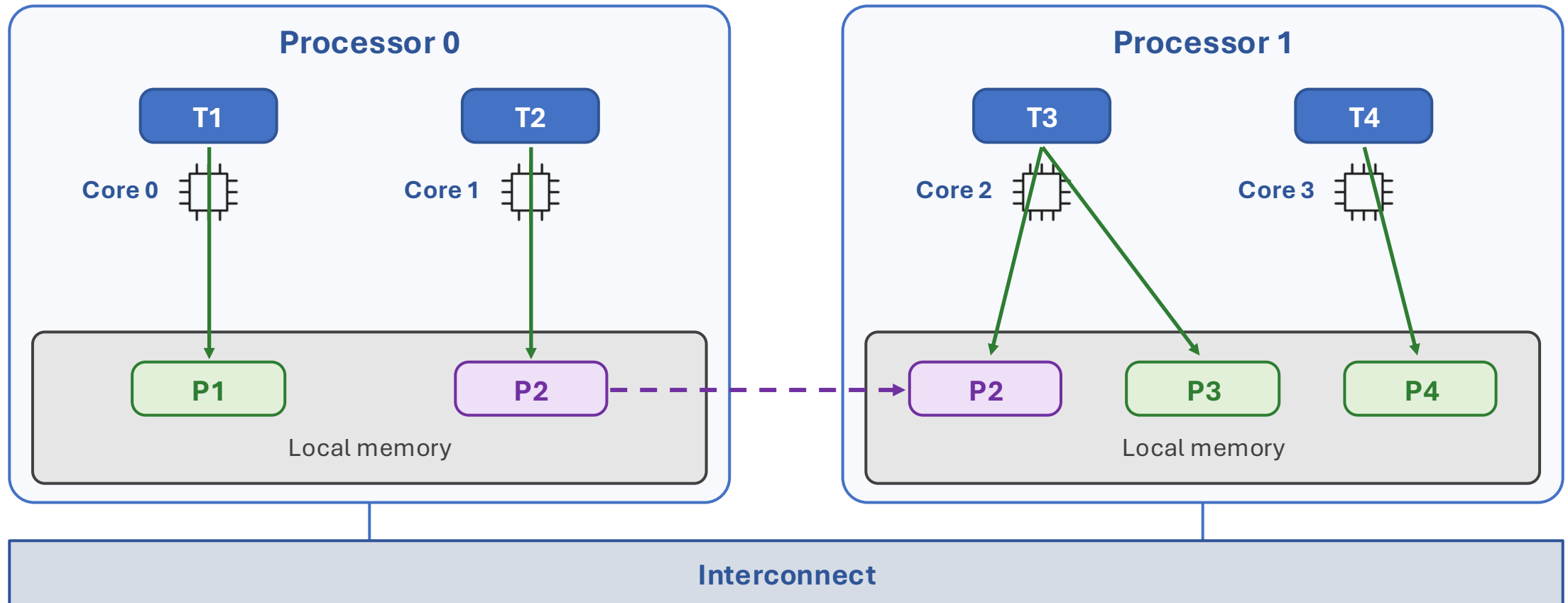
P2 is migrated into Processor 1's memory, so T2's access becomes local

Page migration procedure (simplified)

- Allocate new frame
- Lock the page
- Unmap page table entries
- Flush TLBs
- Copy data from old frame to new frame
- Update page table entries to new frame
- Unlock the page
- Free old frame

Discussion: Costs of page migration?

Page replication



Page replication

- How to maintain consistency across replicas?
 - Pages “write protected” (trigger page fault on write)
 - Collapse replicas on write

Discussion: Costs of page replication?

Access tracking

- (Per-page, per-processor) miss counters
- Memory controller counts misses
 - Interrupts CPU when counter exceeds threshold

Discussion: Costs of access tracking?

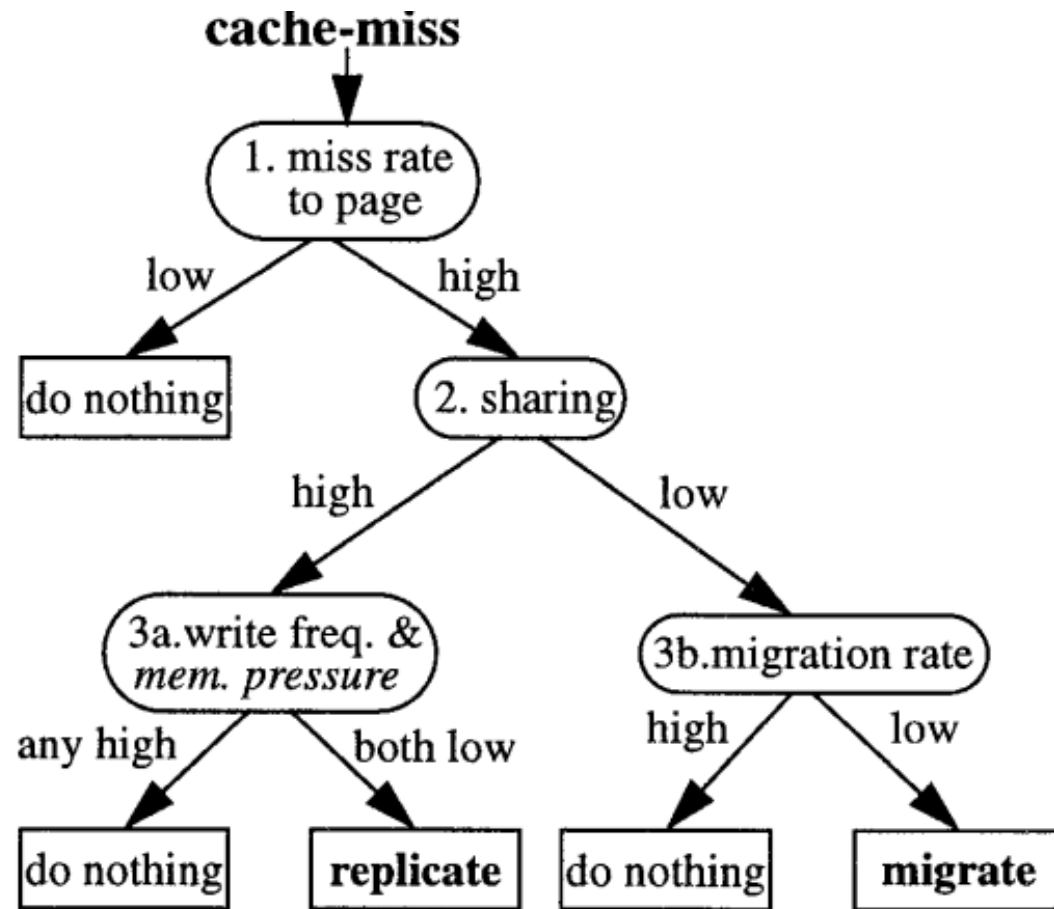
Key policy challenge

- Balancing benefits of migration and replication with costs

Policy: Classification of pages

- Primarily accessed by a single process
- Accessed by multiple processes, but read-mostly
- Accessed by multiple processes, both read/write

Policy decision tree



Discussion: When would the policy fail?

Evaluation takeaways

- Dynamic page migration and replication can significantly improve application performance on NUMA systems
- Migration and replication are complementary; using both is generally better than either alone
- Improving memory locality also reduces contention in the memory system and interconnect
- TLB misses not a reliable proxy for memory accesses; effective page placement requires tracking cache misses / actual memory references
- OS overheads such as TLB shootdowns, page allocation, and synchronization are major costs of dynamic page placement

Today's picture: Hardware

- CC-NUMA: Multi-socket servers common today
- What happened to CC-NOW?
- Revival of CC-NOW with modern interconnects (e.g., CXL)?

Linux NUMA memory management

- AutoNUMA or NUMA balancing
- High level comparison to today's paper
 - Access tracking: doesn't assume memory controller hardware support
 - Page migration: yes, similar approach
 - Page replication: no
 - Policies: Adapted, co-designed with CPU scheduler

AutoNUMA: Hint page faults

- Access tracking mechanism
- Periodically scan pages of every thread and mark pages
 - Pages scanned in chunks (default: 256MB) every “scan period”
 - Scan period measured in thread runtime and not wall clock time
 - Pages are marked with PROT_NONE protection bit
 - Any subsequent access will result in page fault (called “hint page fault”)
- Upon hint page fault
 - Update fault statistics maintained per-thread per-NUMA node
 - Potentially migrate page to different NUMA node (NUMA page migration policy)
 - Store CPU id, last bits of pid in struct page (used for future migration decisions)

AutoNUMA: Page migration policy

- Quadratic filter: only consider migration if page is accessed twice from same NUMA node
 - Heuristic to prevent excessive migrations
 - Two cases:
 - “Private fault”: Consecutive accesses by same thread
 - Always allow migration
 - “Shared fault”: Consecutive accessed by different threads
 - Allow migration only under certain conditions
 - Threads accessing shared pages are grouped (task groups)
 - Only allow migration from more “heavily used” node to less “heavily used” node
 - Computed based on fault statistics aggregated over task group

Next week

- Virtual Machine (VM) memory management
- Memory offloading (TMO)