

Topology-aware TLB Shutdown Protocols: It's more complicated than you think

Ning Ding

Margo I. Seltzer

Reto Achermann

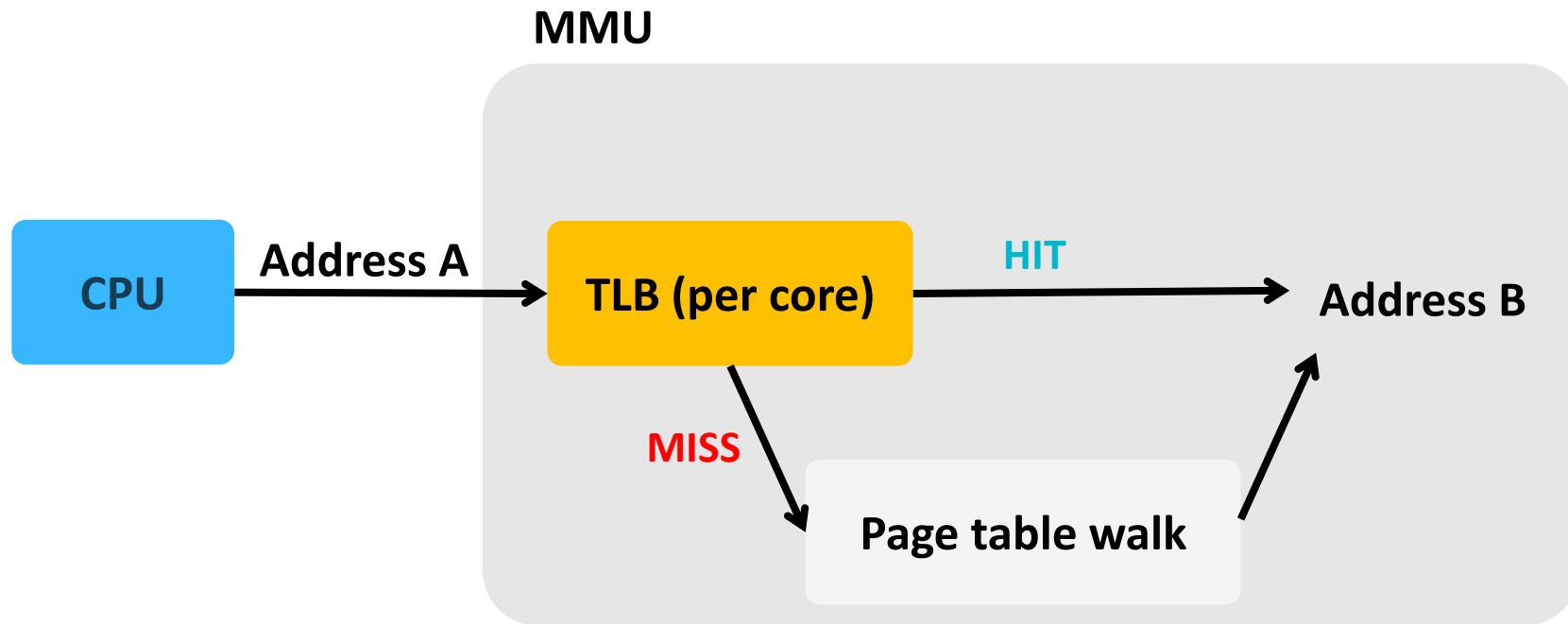
University of British Columbia

Technical University of Munich



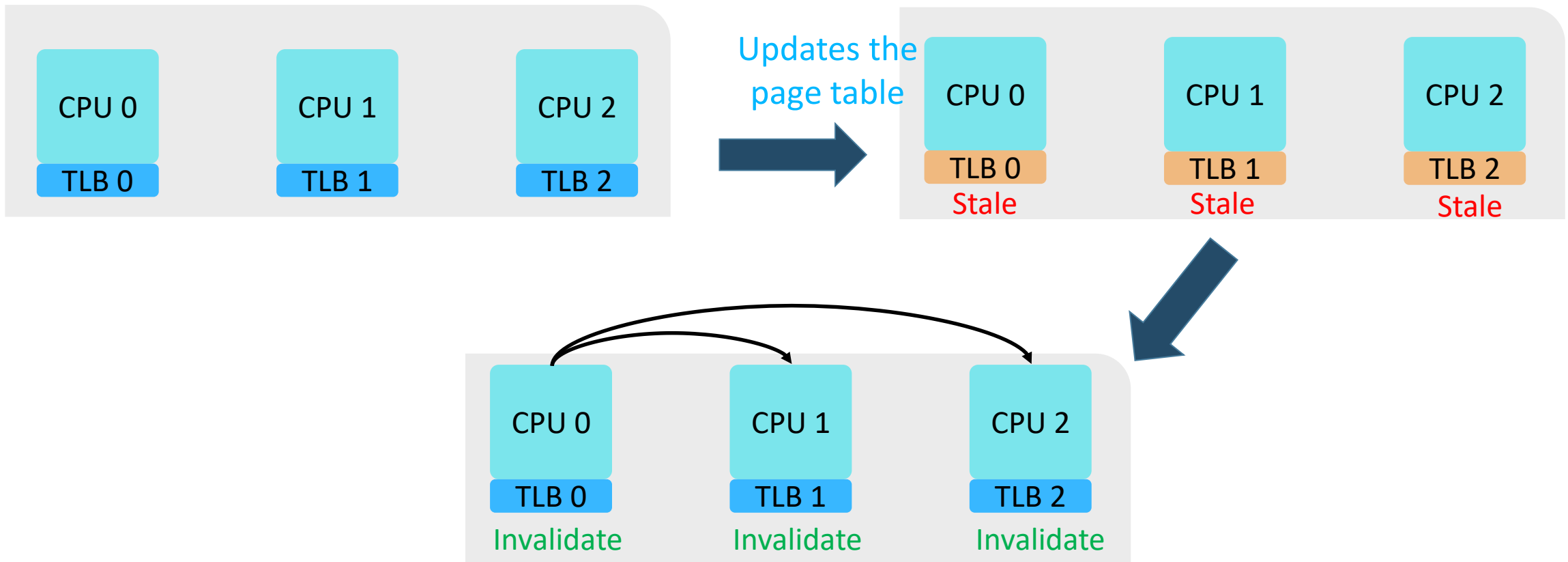
Background: address translation and the TLB

- Modern operating systems use virtual memory.
- Every memory access need translation through Memory Management Unit.
- Page table walk is expensive, so we need a cache: Translation Lookaside Buffer.



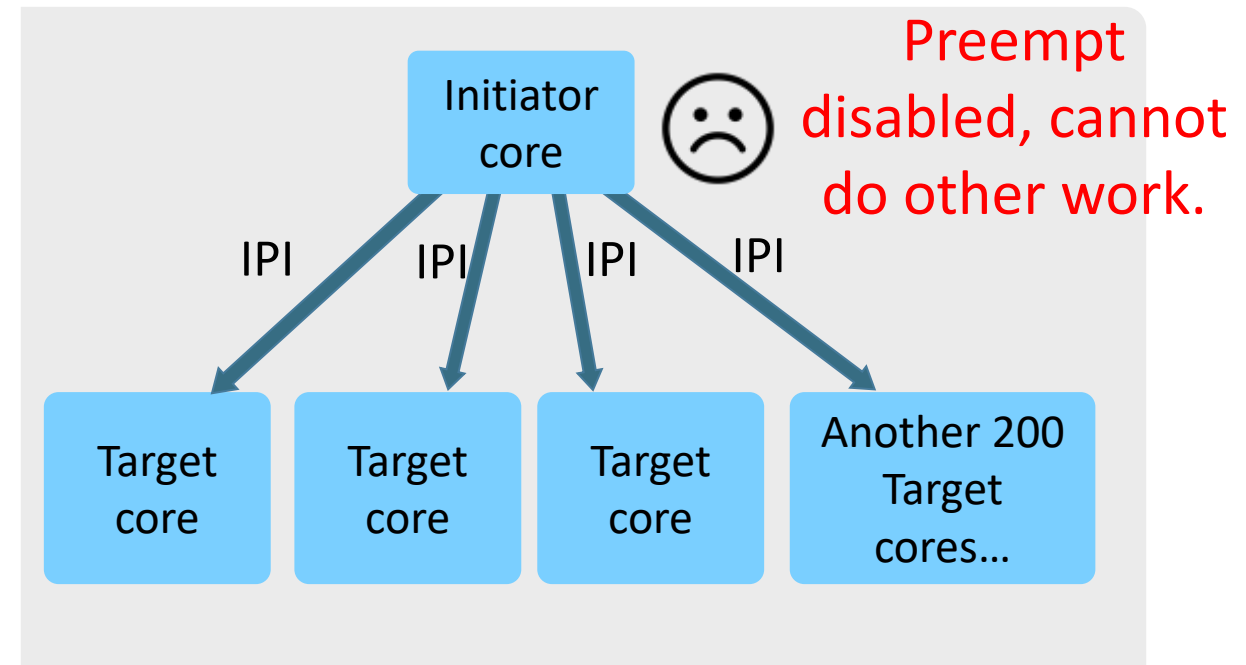
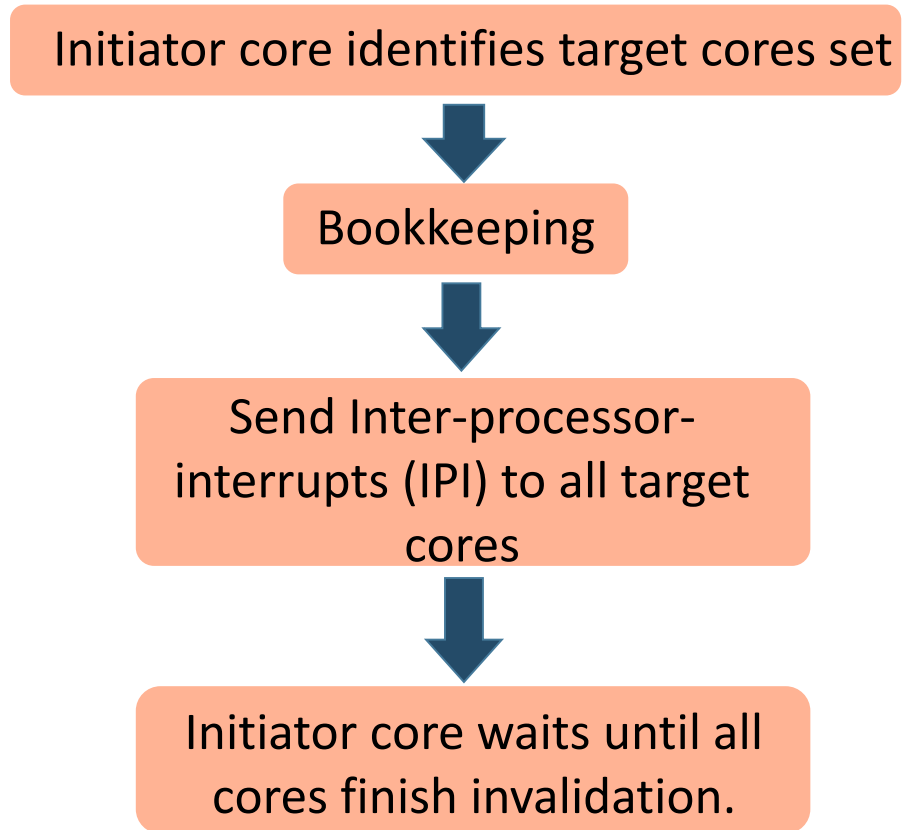
Background: What is TLB shutdown?

- TLB: A non-coherent per CPU cache.
- Stale mappings can lead to security issues.
- TLB coherent protocol: TLB shutdown



Background: How does Linux do shutdown? Why does TLB shutdown matter?

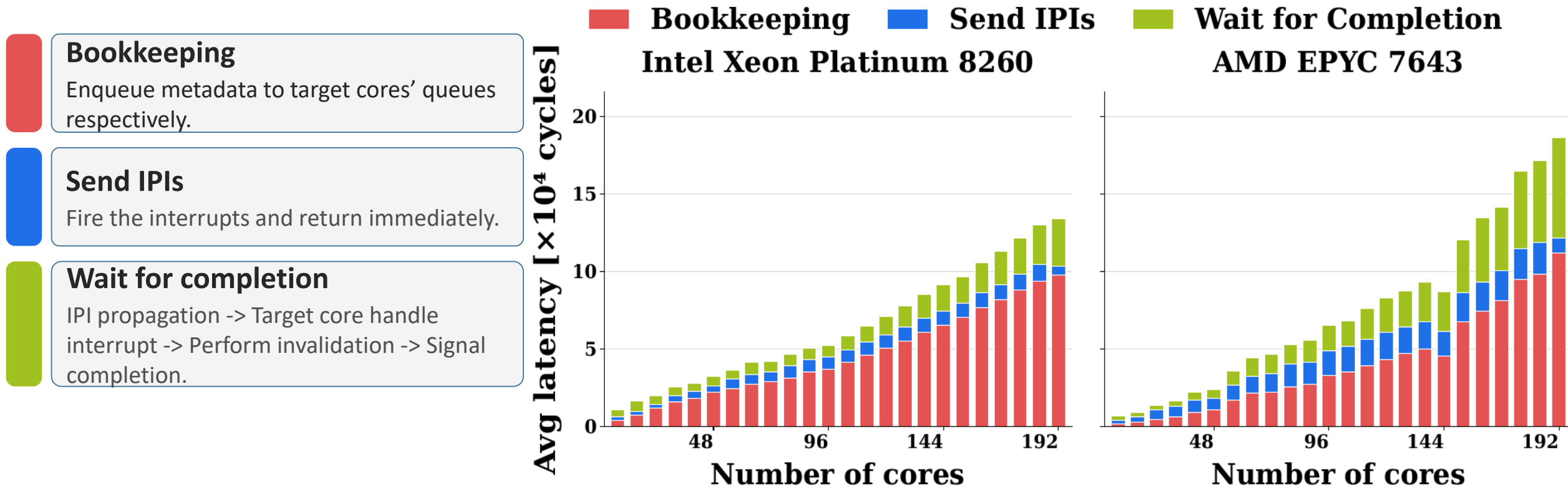
- TLB shutdown is on the critical path.
- Highly synchronized: initiator must wait.
- Latency grows with system scale.



**How can we make TLB shutdown
latency scalable on large machines?**

Where did the shutdown time go on large machines?

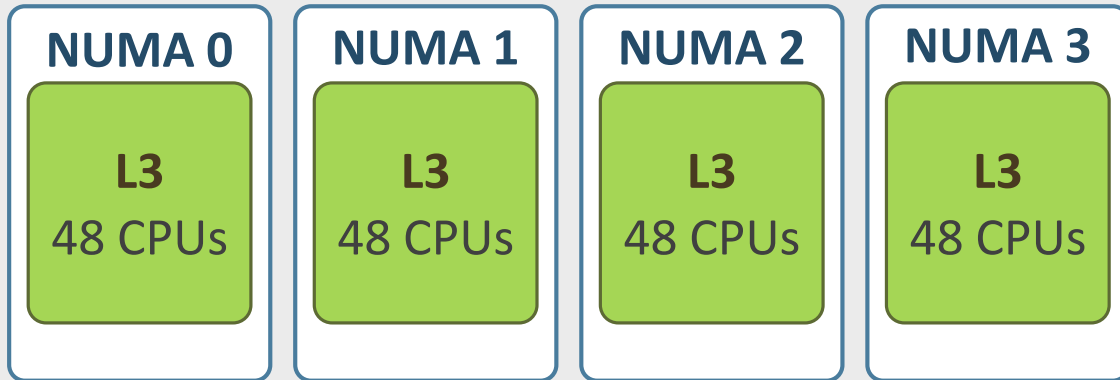
- Platforms: both with 192 logical cores.
- Workloads: `madvise(DONT_NEED)` micro-benchmark.



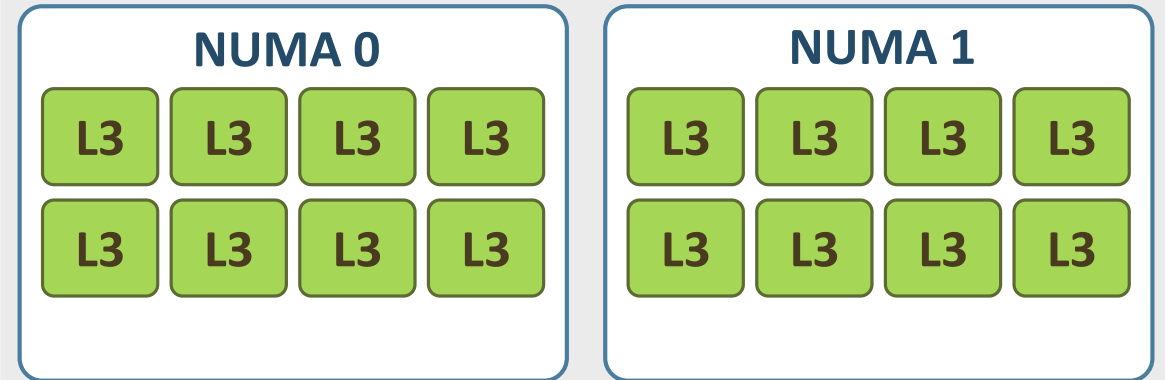
How does hardware effects on large machine affect TLB shutdown performance?

- Large machines have complex hardware topology.

Intel Xeon Platinum 8260



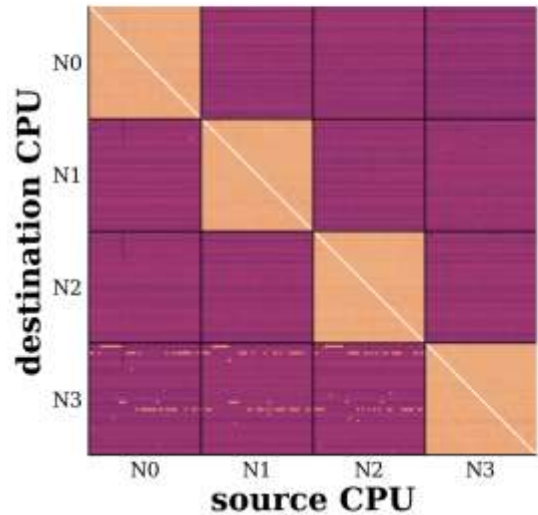
AMD EPYC 7643



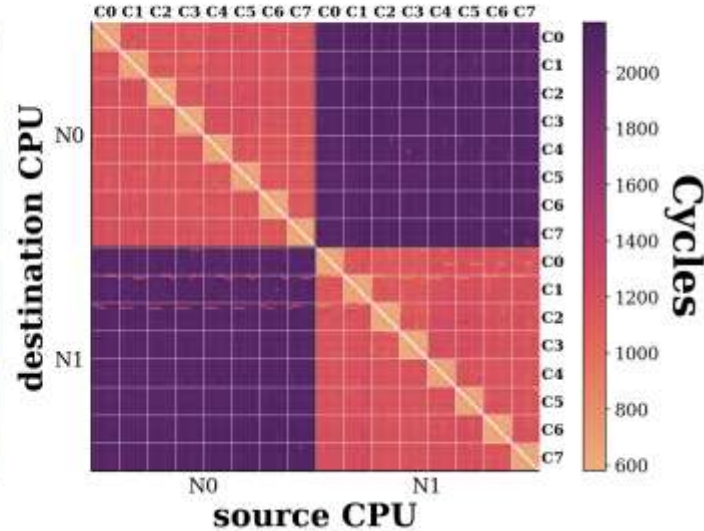
Is every target core equally expensive in TLB shutdown on large machines?

Mean Bookkeeping Time in Cycles

Intel Xeon Platinum 8260

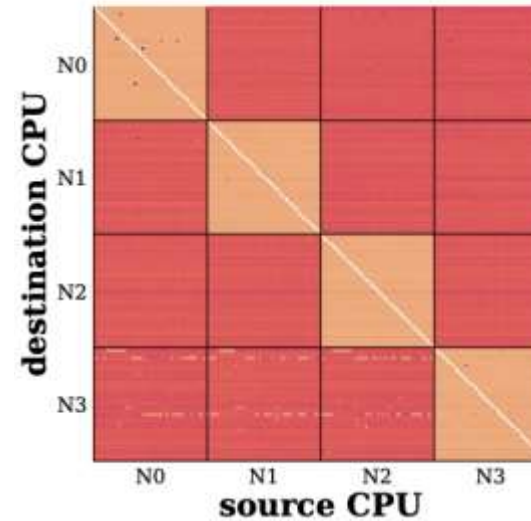


AMD EPYC 7643

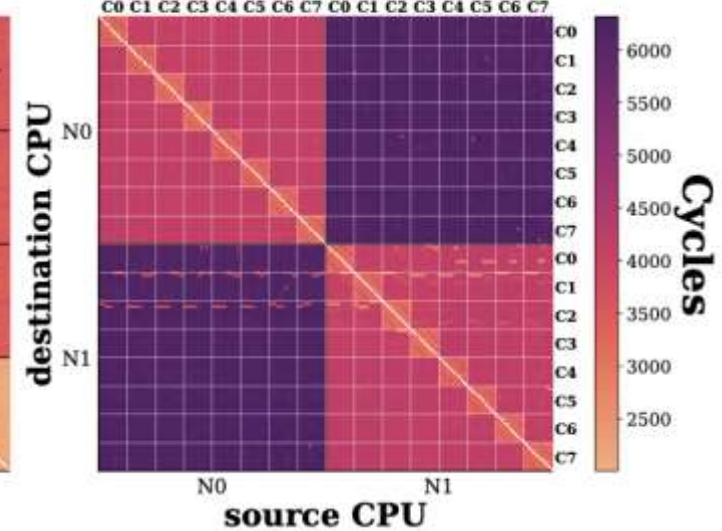


Mean Waiting Time in Cycles

Intel Xeon Platinum 8260



AMD EPYC 7643



Hardware Topology affects both bookkeeping, and waiting time in TLB shutdown. Making each target core cost uneven.

Design: A new hardware topology-aware TLB shutdown protocol

1

Distribute the bookkeeping.

One core do all the work ---> A few propagator cores do it in parallel.

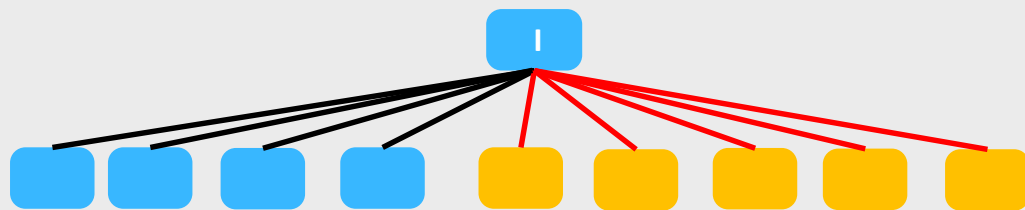
2

Keep communication local.

Each propagator forwards to nearby cores.

- 1. Reduces expensive remote memory accesses
- 2. Keeps the extra forwarding step cheap.

Native Linux



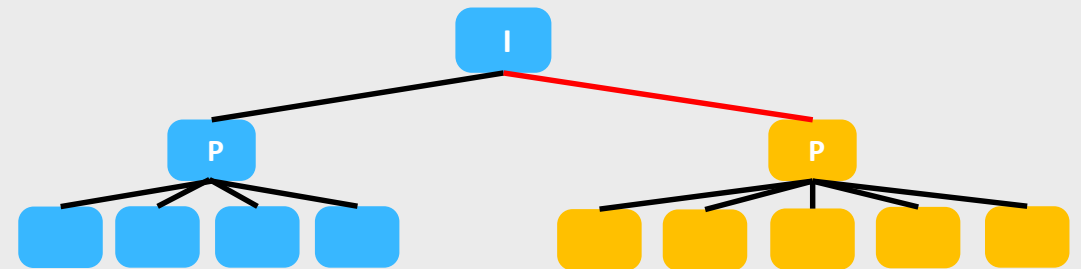
■ Cores in
NUMA 0

■ Cores in
NUMA 1

— Inter-NUMA communication
(expensive)

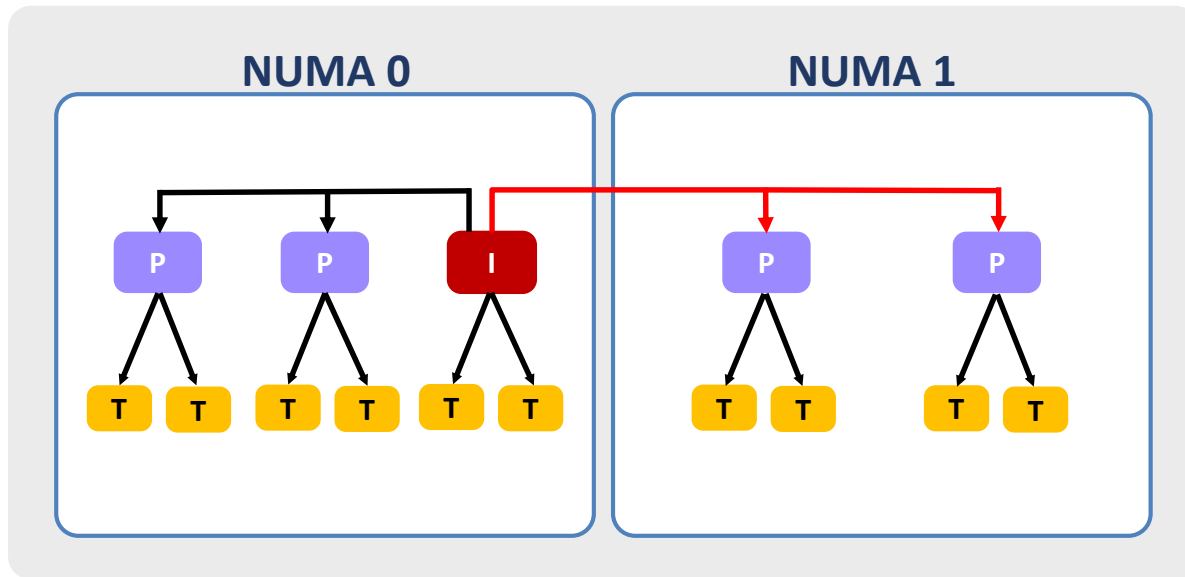
— Intra-NUMA communication
(cheap)

Ours: a few propagators, along cheap links



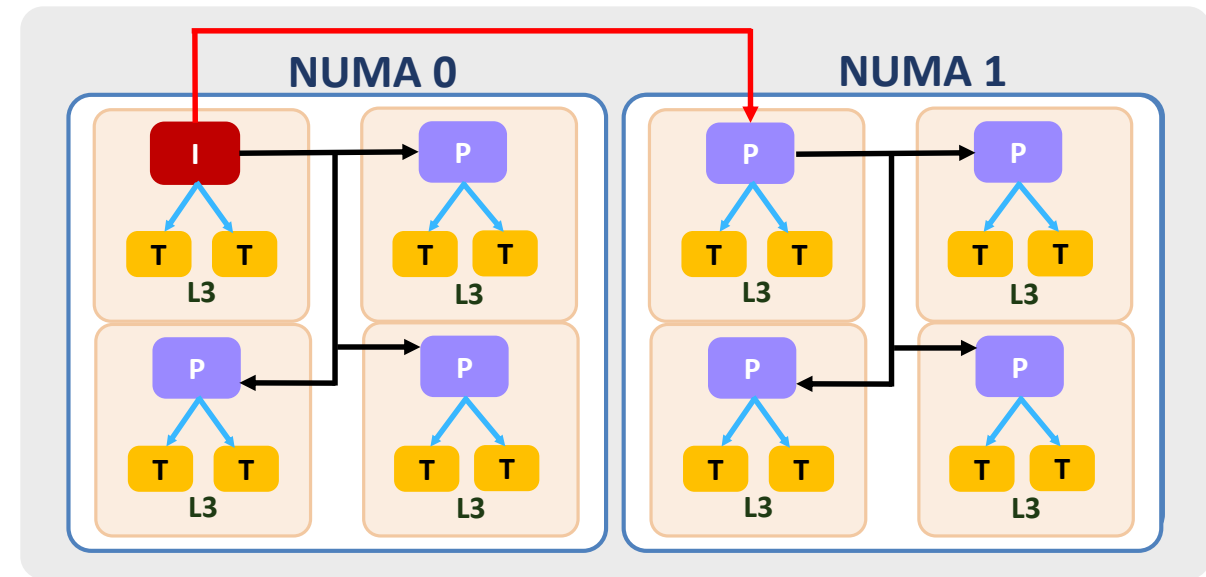
Strategy 1: NUMA-clustered

- Initiator chooses several propagators in each NUMA-node, delegate the forwarding tasks to them.



Strategy 2: L3-clustered

- L3-clustered adopts the same idea, but with one more layer to group by L3 cache domain



P propagator core **T** other target cores
I initiator core

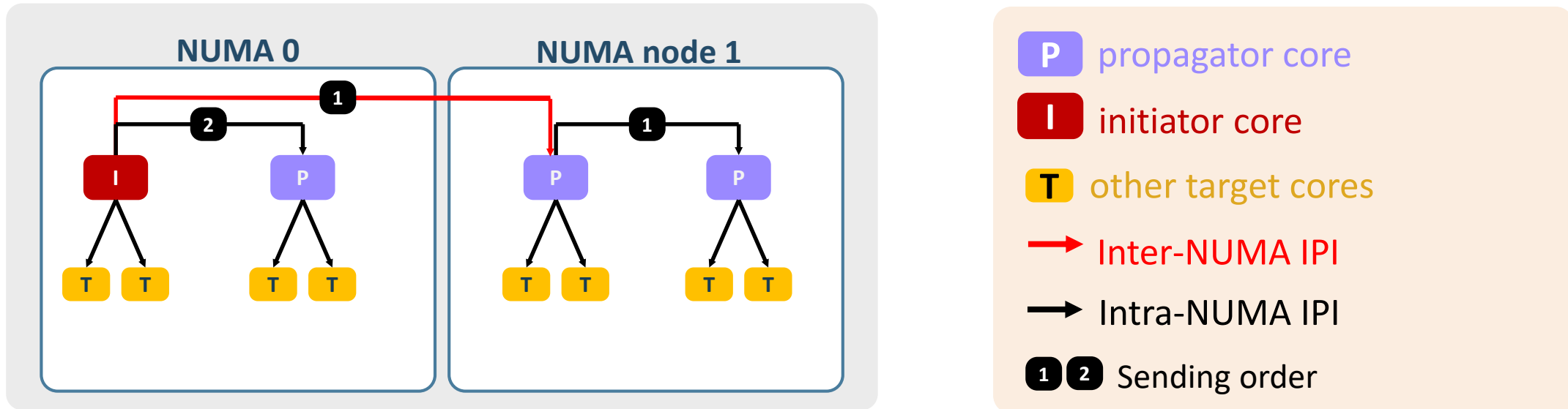
→ Intra-L3 IPI

→ Inter-NUMA IPI

→ Intra-NUMA IPI

Strategy 3: Adaptive

- We use the SMELT library, which minimizes broadcast latency, to build a broadcast tree.
- Each shutdown prunes the full tree with all cores to a subtree with only target cores.

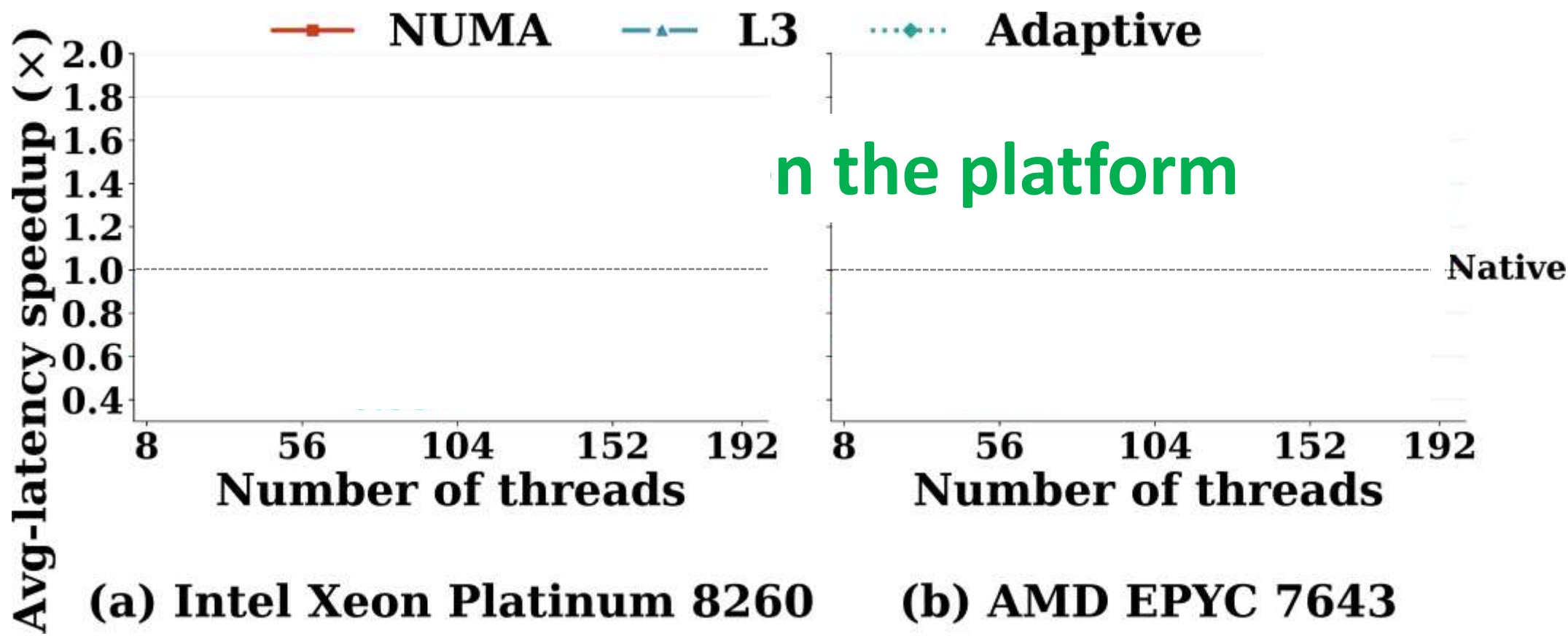


Send IPI over expensive remote links first. While that IPI is in flight, the sender keeps notifying nearby cores.

Evaluation

madvise(): Can our design reduce syscall latency in a controlled environment?

One thread change page table with `madvise(DONT_NEED)`; the others keep the address space active, so the target cores set stays stable.



Apache web server: Do these improvements translate into higher throughput in an application?

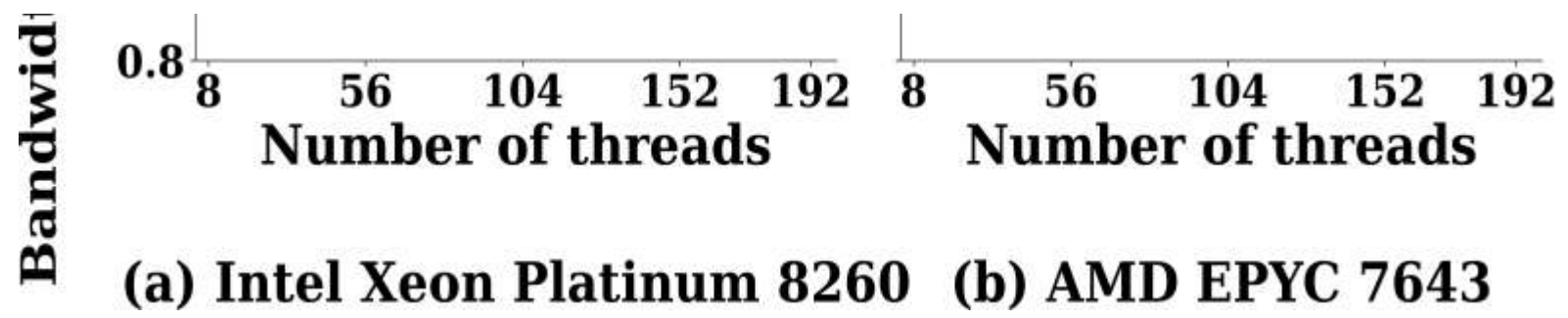
- Event MPM: for each requests, map a region, write the file content into network socket, then `unmap` when the request finished.



Modest
But most
More are

o p
p

ology.



Takeaways

- **TLB shutdown is hard to scale on large machines.**
- **- Topology.**
- **- Number of cores.**
- **Phases proportion is not a fixed number.**
- **No universal winner.**