CXL: All Roads Lead to Coherent Fabrics

PJ Waskiewicz

Systems Architect, Jump Trading

Netdev 0x1A, Rome 2026

July 16, 2026

What should you expect today?

- **A recap: what is CXL?**
  Protocols, device types, and the road from expansion to fabrics
- **The bandwidth wall**
  Why “far NUMA node” is the wrong mental model for CXL memory
- **What pooling buys today**
  CXL 2.0 reality check: partitioned, not shared — and the skeptics
- **The moonshot, layer by layer**
  Switching, firmware, and the kernel memory + networking subsystems
- **Where should coherency live?**
  Five candidates, one conjecture: tiered coherence with edge home agents

Jump TRADING

Who am I?

Linux Kernel Developer, co-maintainer, 20+ years in the kernel

Systems Architect at Jump Trading

I like 3D printing and designing my own stuff in CAD

Jump TRADING

The Recap

What is CXL, and where is it headed?

CXL: The Background

Compute eXpress Link

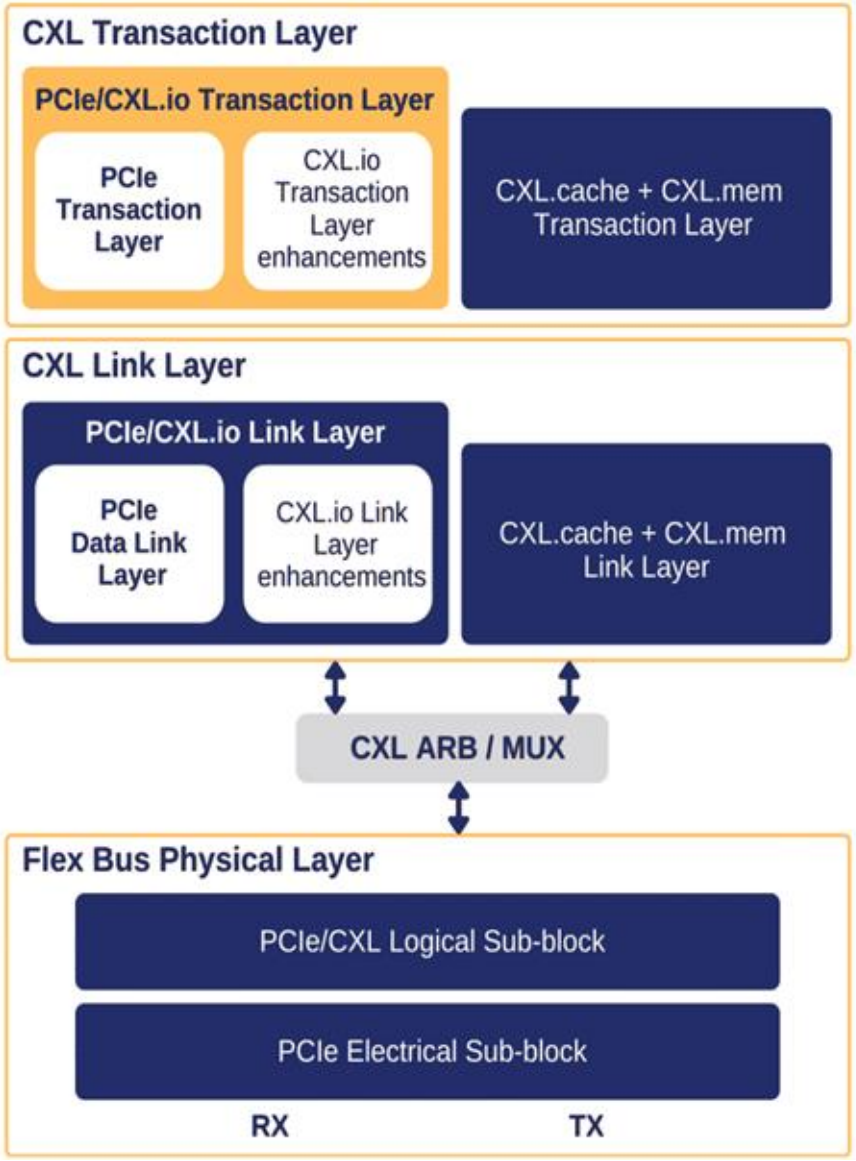
"New" interconnect protocol

- CXL.io: PCIe within CXL flit (framing)
- CXL.mem: endpoint memory coherent with the host
- CXL.cache: endpoint cache coherent with the host

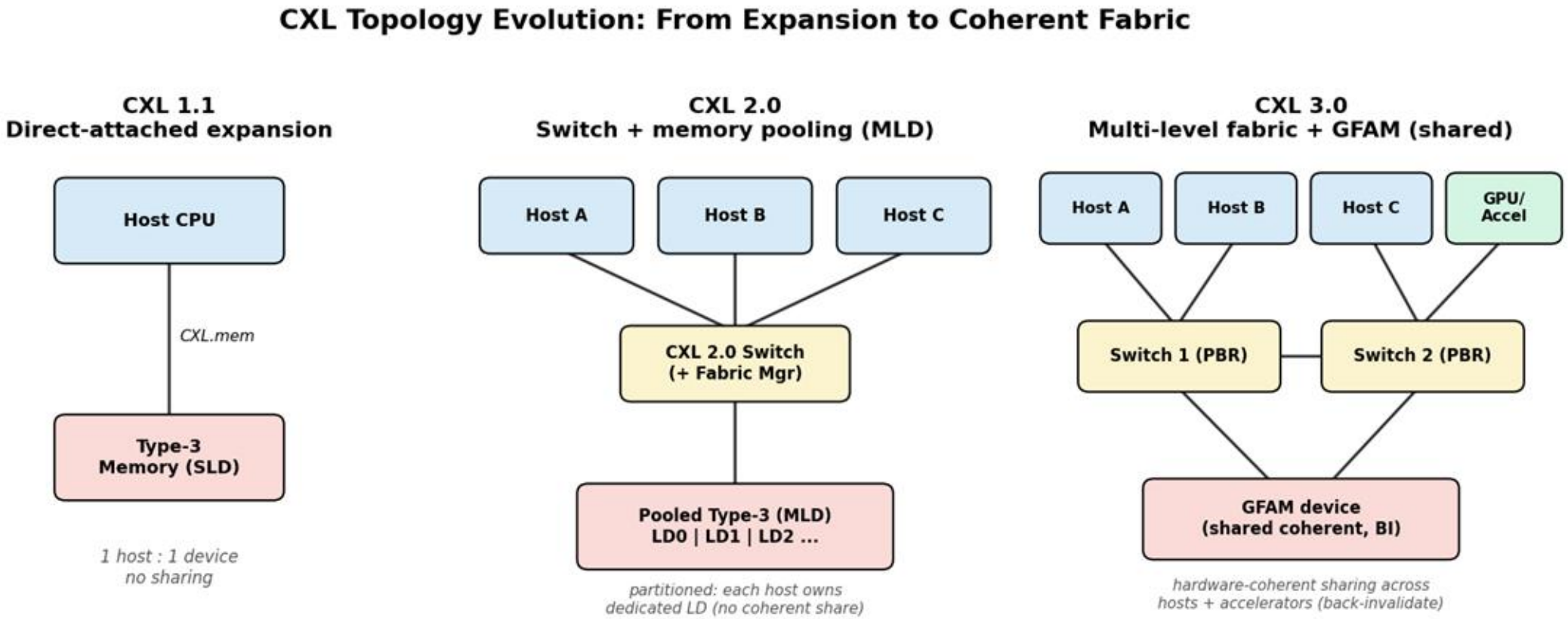
Runs on PCIe hardware (PHY)

Type-3 device = a CPU-less NUMA node

...and there the trouble starts



CXL in three acts: expansion, pooling, fabrics



Jump TRADING

CXL: The Roadmaps

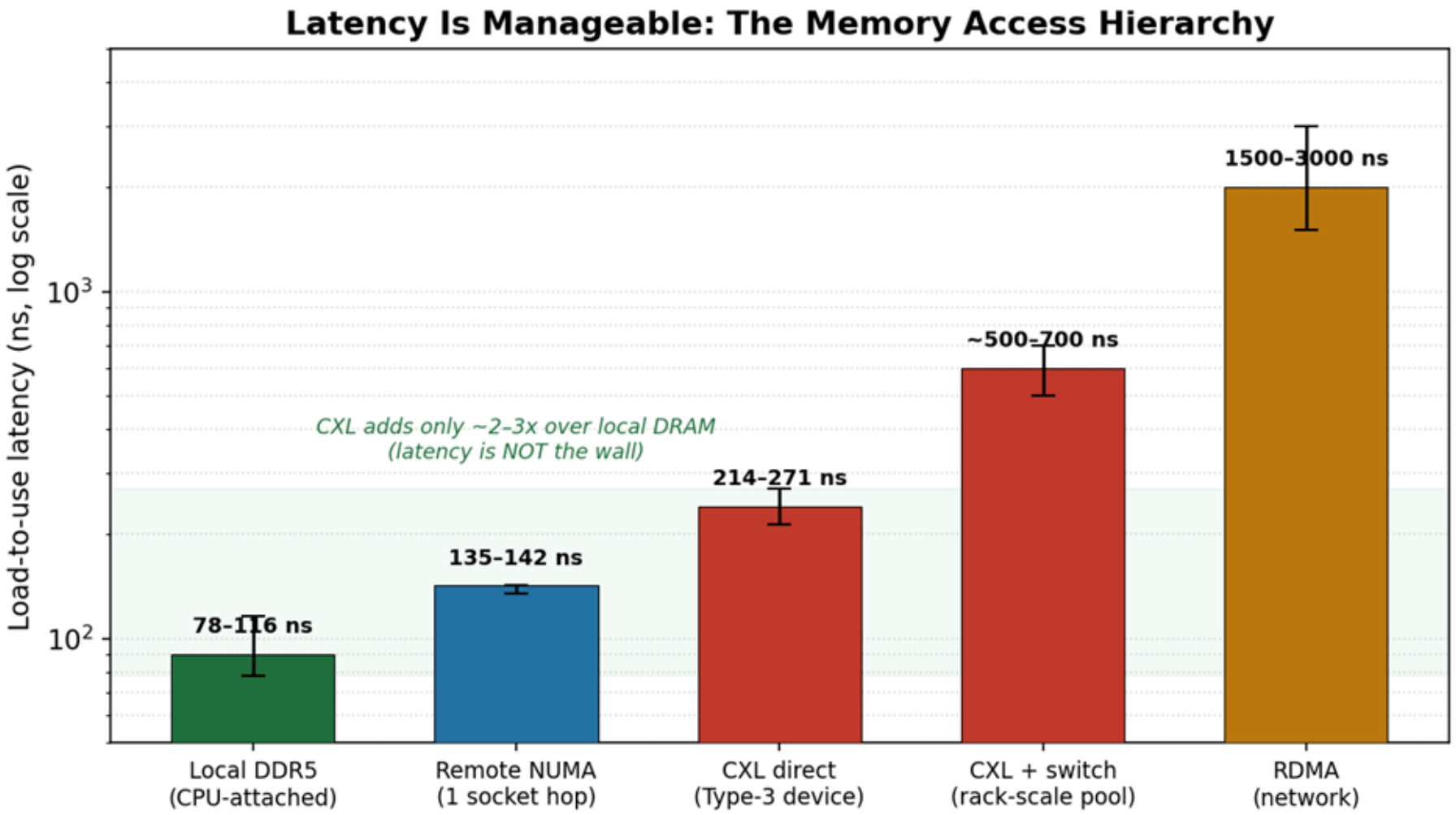
CXL Version	Intel	AMD
CXL 1.1: Initial version, no hotplug, fully reliant on BIOS	Sapphire Rapids / Emerald Rapids	Genoa
CXL 2.0: Initial memory pooling, hotplug, OS support, etc.	Granite Rapids	Turin
CXL 3.x: Fabrics + PBR, BI coherency, GIM (3.1), CHMU (3.2), PCIe 6.0	Diamond Rapids	Venice
CXL 4.0: PCIe 7.0 at 128 GT/s, bundled ports — bandwidth features	???	???

Jump TRADING

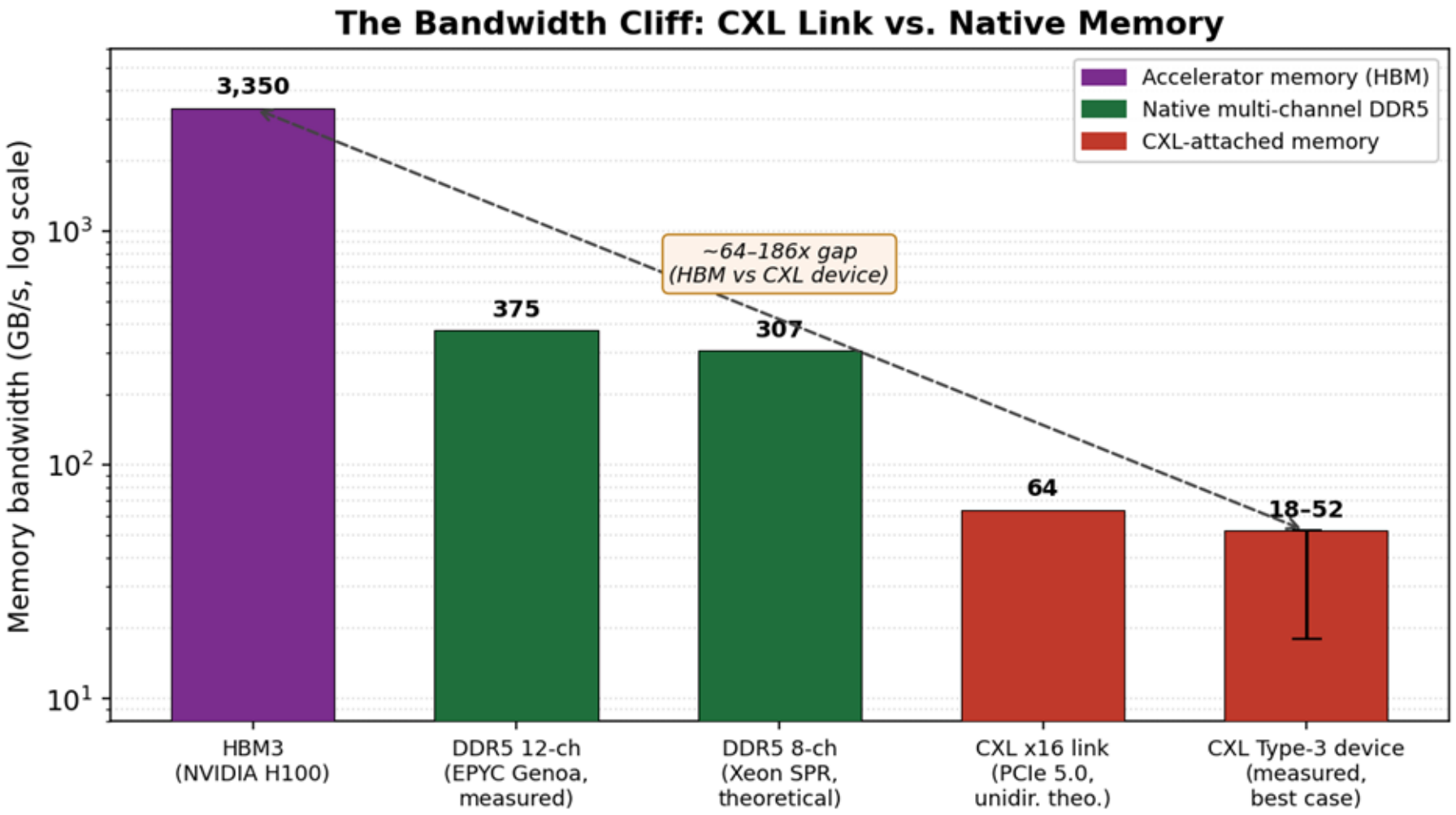
The Bandwidth Wall

Here comes the wall…

Latency: the part the industry gets right

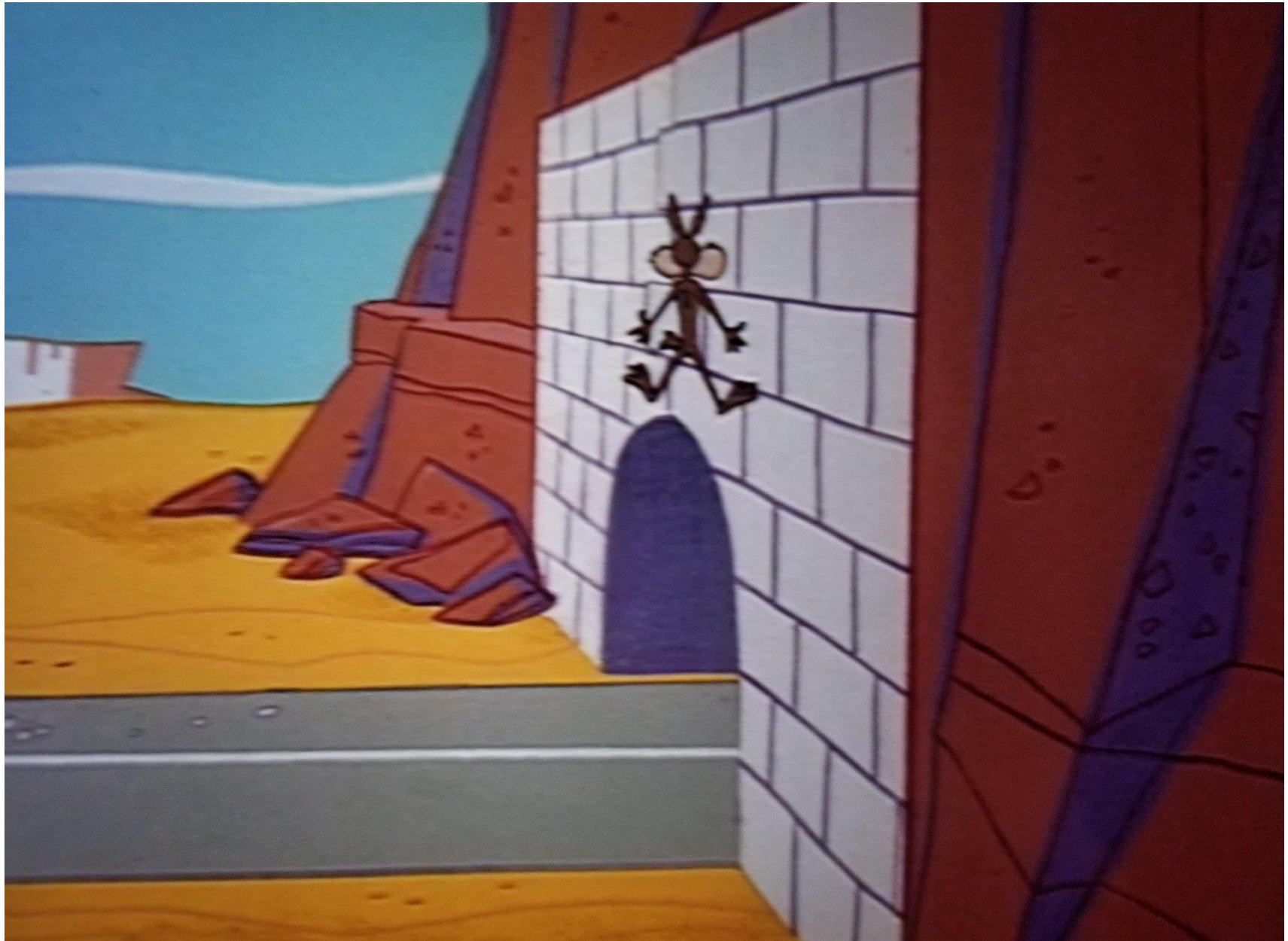


Bandwidth: where the analogy collapses



Jump TRADING

Meet the bandwidth wall



Jump TRADING

The gap, stated precisely

Versus a single DDR5 socket (~375 GB/s measured)

- A CXL device (18–52 GB/s measured): 7–20× slower — one order of magnitude

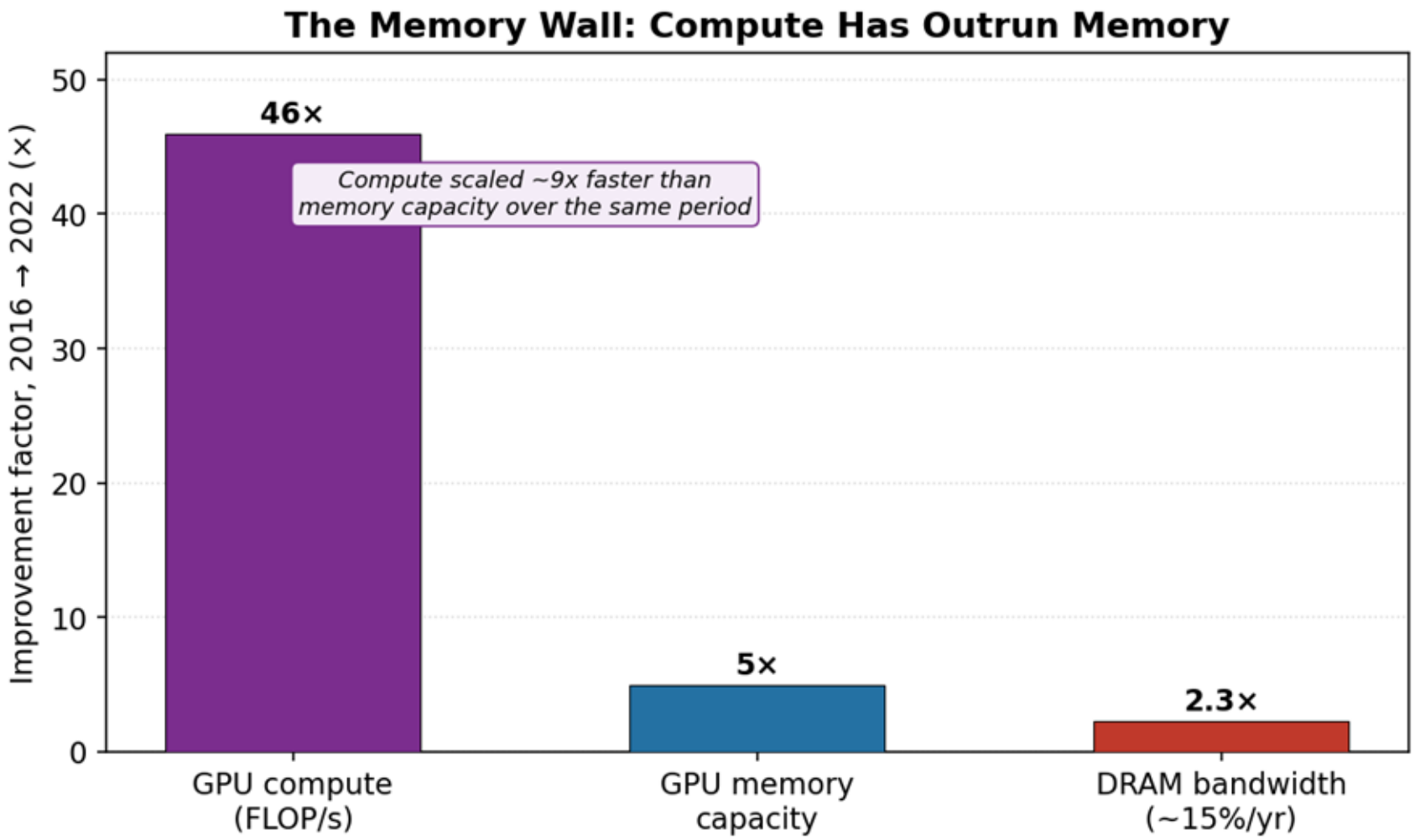
Versus HBM3 on an H100 (~3,350 GB/s)

- 64–186× slower — nearly two orders of magnitude

Shared across a pool, per consuming core

- Comfortably 2+ orders of magnitude
- Adding devices doesn’t save you — the PCIe lane budget is the wall

Why this changes the economics of AI



Jump TRADING

Pooling Today

What did CXL 2.0 actually buy us?

Jump TRADING

Partitioned, not shared

CXL 2.0 pooling = capacity multiplexing

- MLD (Multi-Logical Device): one device, up to 16 LDs + one FM's
- FM binds LDs to hosts via the CCI (Component Command Interface)

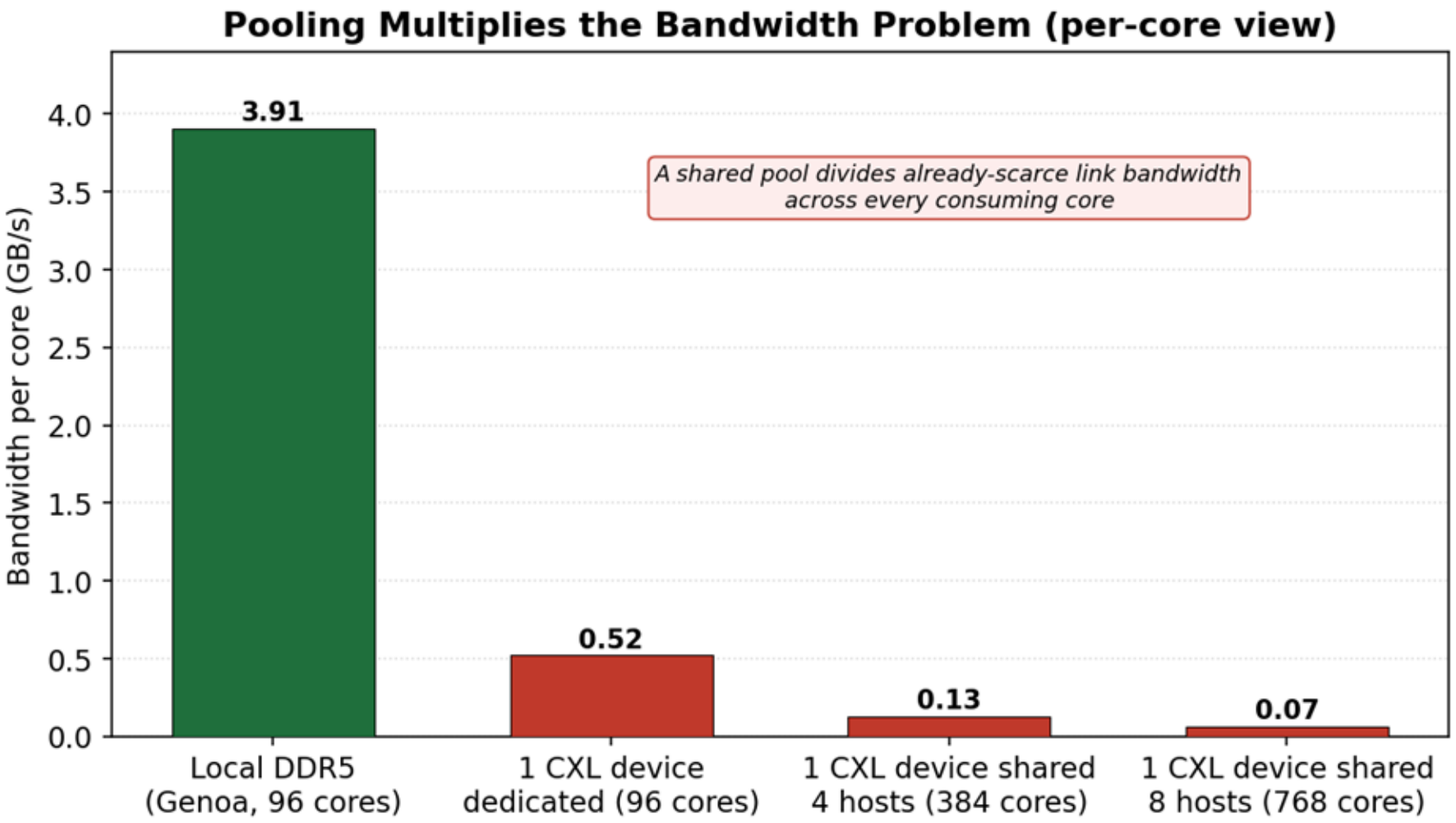
Two hosts on one MLD share a device, not a region

- No hardware-coherent cross-host sharing anywhere in 2.0
- Capacity multiplexing — not collaboration on common data

The best case for it: Pond (ASPLOS'23)

- 8–16 socket pools: +75–90 ns, ~7% less DRAM fleet-wide

Pooling divides what little you have



Jump TRADING

The skeptic's case, stated fairly

- "A Case Against CXL Memory Pooling" — Google, HotNets'23
  - Utility: packing strands little memory — ≤1% gain at 16× VM inflation
  - Complexity: needs explicit staging copies; fine-grained access +140 ns
  - Cost: a parallel cabling and switching plane alongside Ethernet
- I think they're right — on their own terms
  - Pooling-for-utilization is contested, and that's fine
  - Bandwidth + coherence for cluster execution is a different question
  - That question is this talk

Jump TRADING

The Moonshot

Layer by layer: what has to be built?

Layer 1: switching and fabric

Coherent switch silicon is nascent

- 3.x fabrics + Port-Based Routing exist on paper (4,096 nodes)
- Even 2.0 single-switch pooling is only now shipping

Physical reach is the hard limit

- Signal integrity caps CXL at roughly 2 m — a rack-scale object
- Beyond that you bridge to RDMA — complement, not replacement

Switches cost latency and become the failure domain

- ~600 ns through a switch; per-port bandwidth turns first-order

Layer 2: firmware — the Fabric Manager

Today's FM is a provisioning tool, not a runtime scheduler

- Inventory and binding via the CCI — MMIO in-band, MCTP out-of-band
- Runs anywhere: host, BMC, switch firmware, device state machine
- CXL 3.1 adds a PBR-switch FM API; OCP CFM wraps it in Redfish

The gaps are kernel-shaped

- Host ↔ FM control path is out-of-band and out of spec
- Address-space remapping can still require a reset or quiesce
- Multi-tenant orchestration (QoS, isolation) is immature

Layer 3: the kernel — memory subsystem

The good news: the tiering plumbing exists

- Demotion in 5.15; TPP (Transparent Page Placement) in 5.18
- Weighted interleave in 6.9 — a bandwidth mitigation, not capacity
- DAMON self-tuning; CHMU (CXL Hotness Monitoring Unit) RFCs on-list

The gap this talk cares about

- Promotion is slow and reactive; bandwidth-aware placement is early
- Coherent shared multi-host memory has no first-class kernel model
- No semantics for cross-host regions, ownership, or fault handling

Jump TRADING

Layer 4: the kernel — networking subsystem

When hosts share a coherent region, network copies vanish

- cMPI (MPI over CXL memory): 49× latency, 72× bandwidth vs TCP
- Competitive with RDMA at small and medium cluster scale
- MPI-over-CXL: same MPI API, transport swapped underneath

The gaps are squarely kernel and fabric work

- Cross-node atomicity often unenforceable on pooled memory
- SW coherence costs ~2.8× latency; HW coherence doesn’t scale
- POSIX SHM and tmpfs are poor fits — cMPI built its own arena

Jump TRADING

Where Does Coherency Live?

The crux of the moonshot

Jump TRADING

Two managers — do not conflate them

The Fabric Manager: a control-plane entity

- Inventory, bind/unbind, composition, QoS config, event handling
- Not in the coherence datapath — it composes, it does not cohere

The coherency manager: a data-plane entity

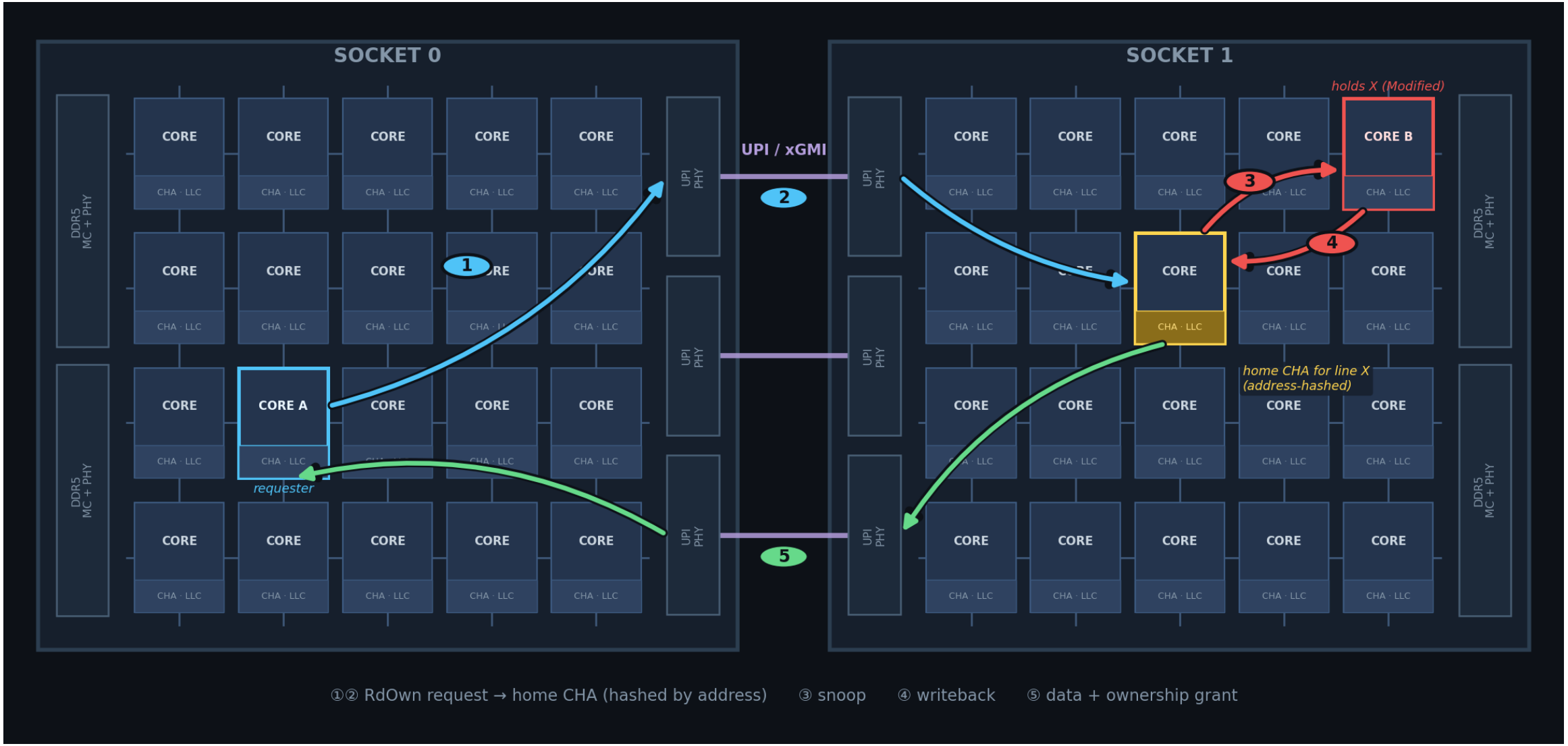
- Home agent + directory + snoop filter enforcing shared cache lines
- CXL 3.x default: snoop filter and Back-Invalidate live on the device

Is the device the right home when no host owns the fabric?

Where can coherency live?

Placement	Strength	Why it breaks at scale
On the device (spec default)	BI must live there; no extra hop	Precise 64 B snoop filter impractical at TB scale; imprecise = BI storms
In the switch	Coherence at the fan-out point; one ordering point	Tiny SRAM; switch becomes the failure + scaling domain
On a host CPU	Works on shipping silicon	CHA contention; multi-hop; a host owns the fabric
On a DPU / SmartNIC	Sits with the NIC; natural cross-rack bridge	Little published; cross-fabric semantics undefined
Hierarchical / distributed	The only family argued to scale out	Copyset encoding + inter-rack consistency unsolved

We know how to do this today...



The conjecture: Fabric-Resident Tiered Coherence

One monolithic HW coherence manager is the wrong target

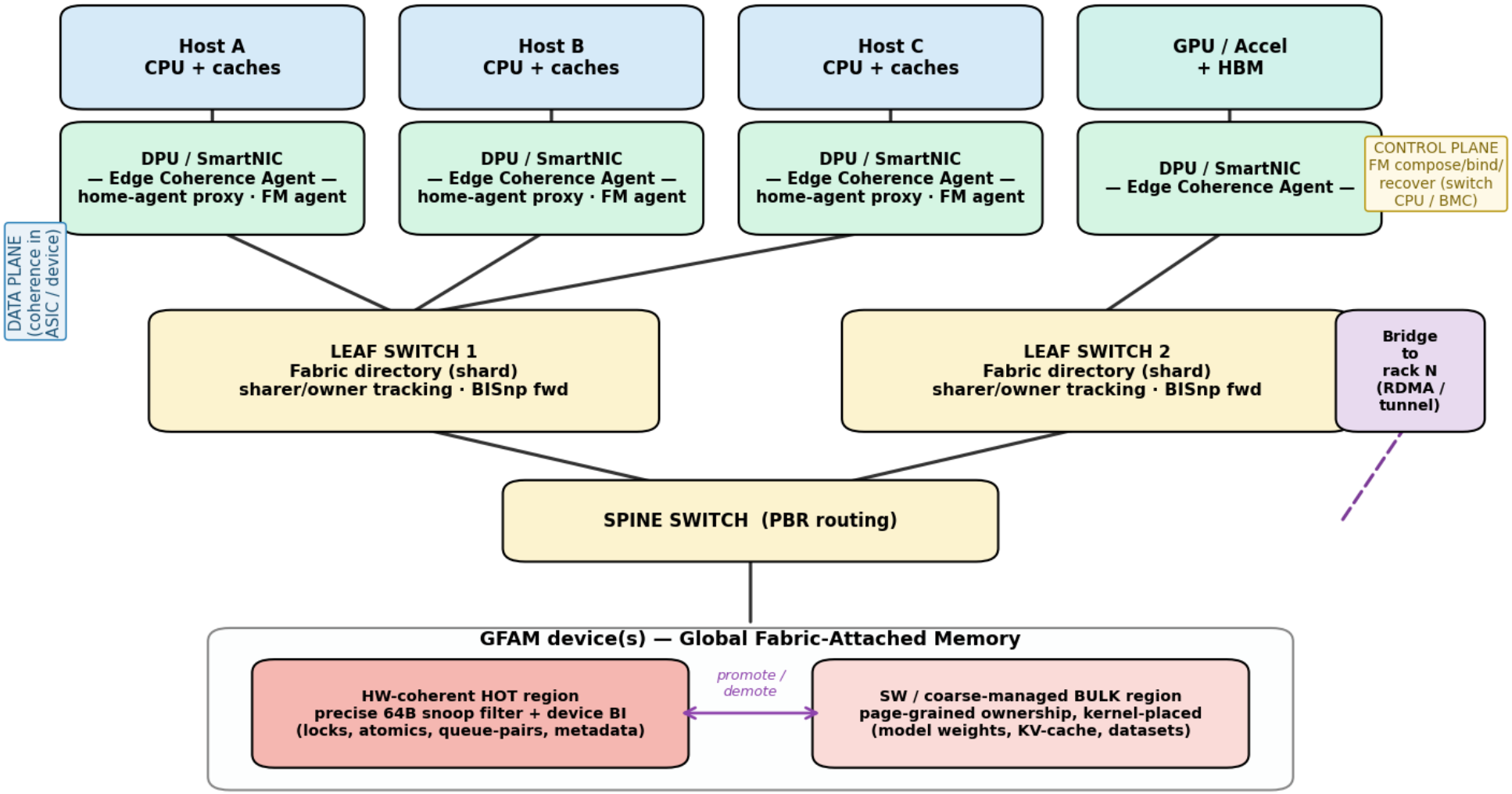
- A single point of failure — likely un-buildable at datacenter scale

FRTC: fabric-resident, distributed, tiered

1. Edge Coherence Agents (ECAs) on each host's DPU/SmartNIC
2. Coherence directory sharded across leaf/spine switch ASICs
3. GFAM: small HW-coherent HOT region + big SW BULK region
4. Promotion and demotion between tiers under kernel control

Hardware precision only where it must be

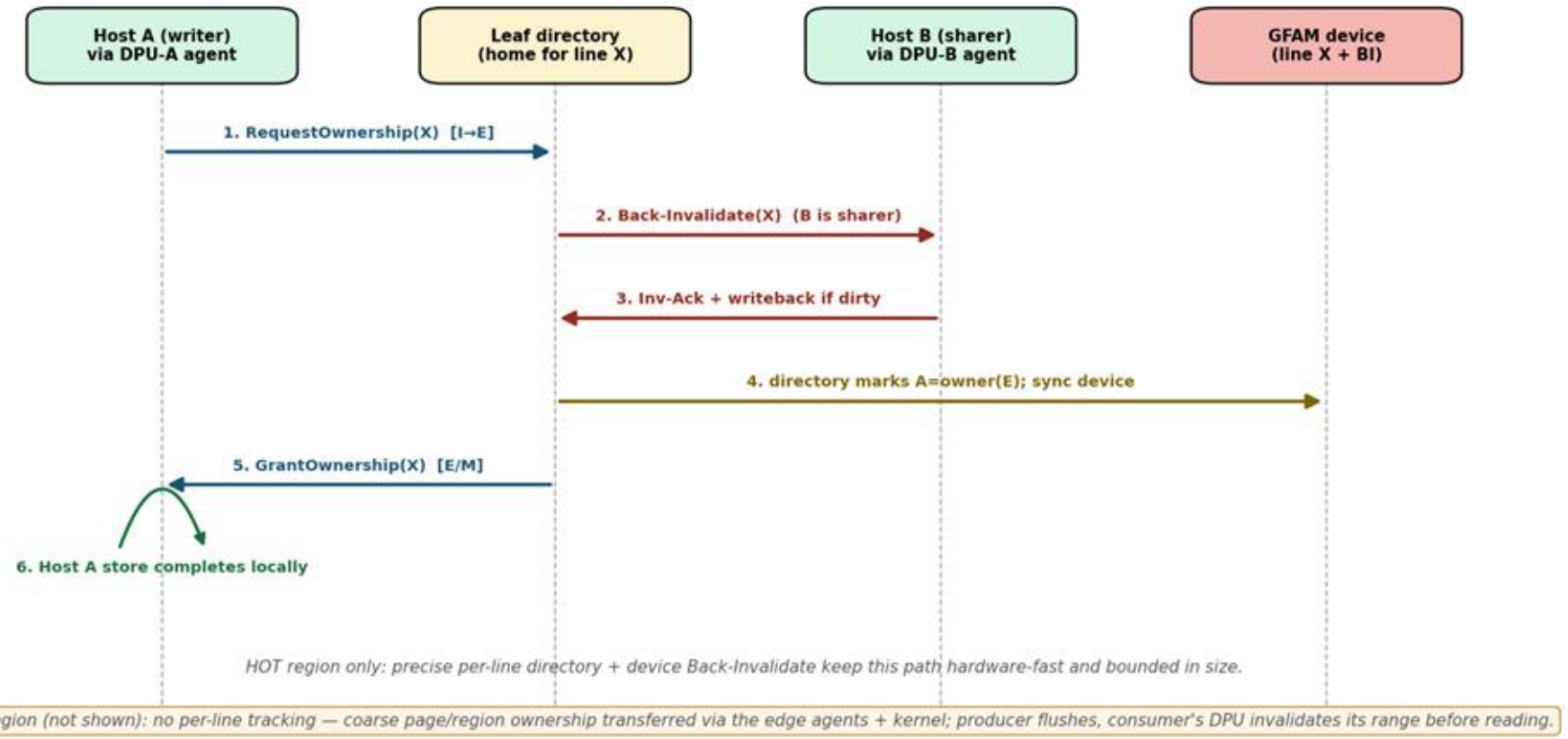
Figure 6. Fabric-Resident Tiered Coherence with Edge Home Agents (conjecture)



No single coherence manager: directory is sharded across switches · home agents distributed to per-host DPUs · coherence is tiered (HW-precise where it must be, SW/coarse where it can be)

A coherent write, sketched

Figure 7. High-level protocol sketch: coherent write to a HOT-region line (conceptual)



Breaking down the wall

netdev: the DPU as a coherence agent

- A new device role — not another NIC offload
- Ownership state + back-invalidate plumbing
- The local-CXL vs RDMA transport decision

mm: coherence as a region attribute

- HOT vs BULK on mappings (extended mmap)
- cgroup accounting for the scarce HOT region
- DAMON/TPP driving coherence tiers too

Neither problem solves the other

Does coherence fix bandwidth? No — it can't widen the pipe

- Every consumed byte must cross the link once — coherent or not

Does bandwidth fix coherence? No — CXL 4.0 proves the point

- Bundled x16 @ 128 GT/s: ~768 GB/s per direction (~1.5 TB/s duplex)
- Fatter links = 10× the ownership churn — a monolithic manager fails faster

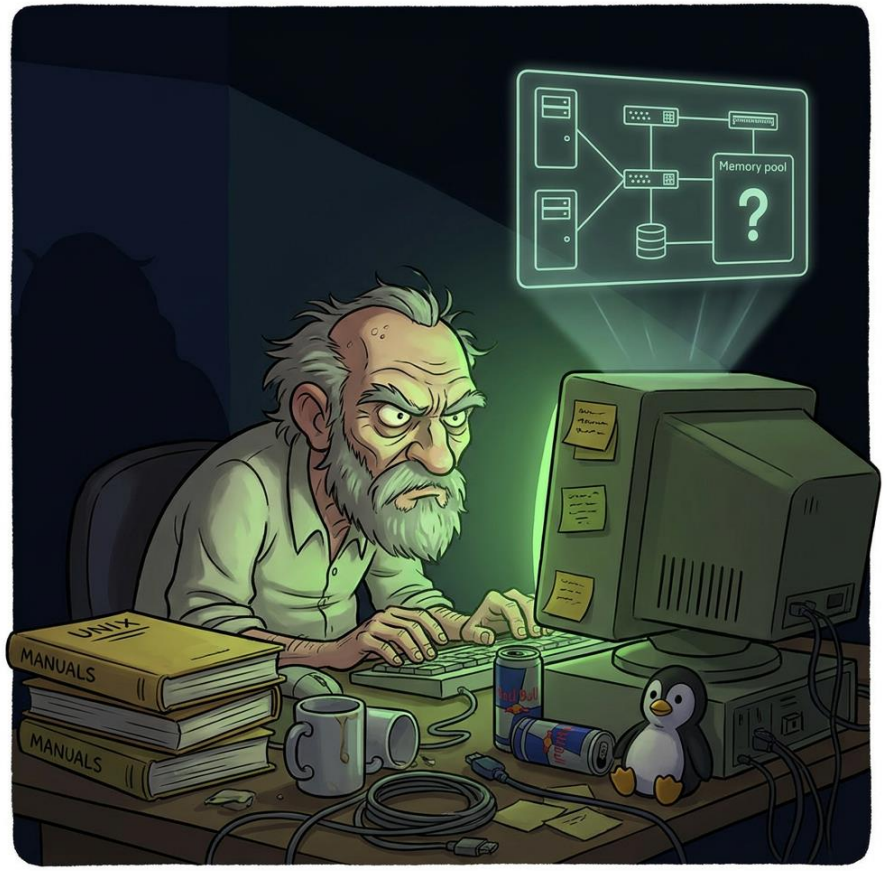
The reuse factor decides — and it's why FRTC is tiered

- Streaming (weights, KV-cache): reuse ≈ 1 — coherence adds only overhead
- Locks, queue-pairs, metadata: hot + shared — coherence pays for itself

Open questions — on purpose

The discussion this talk wants to start

- Directory sharding + address-to-home under multipath
- Compact sharer/copyset encoding at thousands of nodes
- How big can the HOT region actually be?
- Consistency and atomicity across the HW/SW tier boundary
- A correctness model — CXL0 and CXLMC are a start



None of these are solved. That’s what makes it a moonshot.

Jump TRADING

You’re awake, what did you miss?

Latency is fine; bandwidth is the wall — 1 OoM vs DDR, ~2 vs HBM

CXL 2.0 pooling is partitioned capacity, not coherent sharing

The moonshot spans switches, firmware, and both kernel subsystems

Coherency should be fabric-resident, distributed, and tiered

It’s a conjecture — come argue with it

Jump TRADING

Conclusion

All roads lead to Rome — and to coherent fabrics

ppwaskie@kernel.org
pwaskiewicz@jumptrading.com