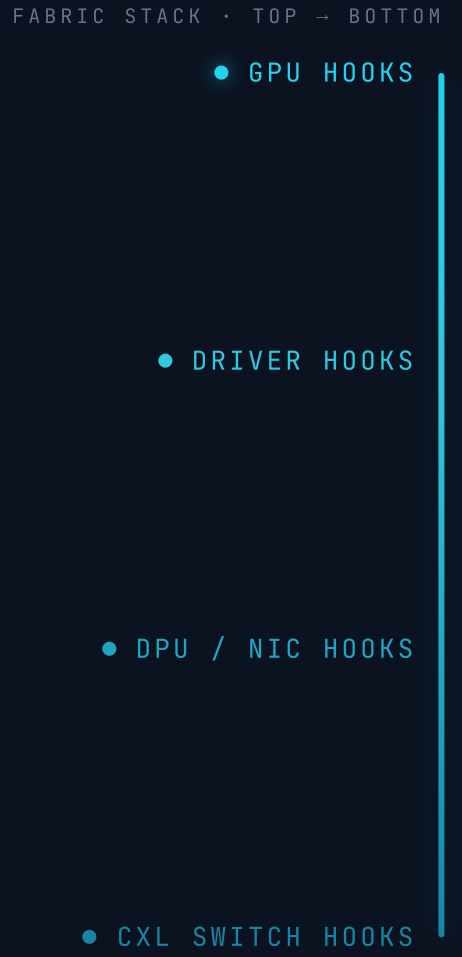
PAPER TALK · SYSTEMS

The Fabric Is the Cluster Driver

Cross-Layer eBPF Policies for GPU–CXL Fabrics



Anonymous Author(s)

PREPRINT · FABRIC_EXT

GPU clusters are no longer accelerators behind a host CPU.

They are fabrics — and every layer exposes a different slice of the truth.

A single LLM serving request today starts as prefill on one GPU, decodes on another, fetches remote KV blocks from CXL memory, routes tokens through MoE experts over RDMA, and shares the same fabric with background training and checkpoint traffic.

GPU kernels expose execution phase, warp behavior, page faults, KV blocks, and tensor semantics. NICs and DPUs expose queue pairs, work queues, completions, retransmits, congestion. CXL switches and memory devices expose placement, sharing, ordering, persistence, and near-memory movement.

Yet today's control planes split these signals across mutually blind policy systems.

4

fabric layers: GPU, host/driver, DPU/NIC, CXL switch

14

LLM fabric workloads evaluated (Qwen-style, MoE, LoRA, scale-out 2→32 GPUs)

86

semantic events compiled into BPF objects — all recompile on BlueField DPU

The semantic split.

GPUs know execution meaning; the fabric knows fabric state. Neither sees the other.

GPU SIDE

Knows execution meaning.

- stream is `prefill / decode / all-reduce / checkpoint / MoE routing`
- warp stalls, kernel phase, page faults, `KV block reuse`
- `sequence lifetime`, `expert ID`, `token batch`
- whether a transfer is on the request `critical path`

CANNOT SEE: RDMA retransmissions, remote QP depth, CXL switch congestion, remote GPU readiness, memory-pool pressure.

X

FABRIC SIDE · DPU / NIC / CXL

Knows fabric state.

- `WQE / CQE timing`, queue depth, retransmissions, timeouts
- routing, `memory-window ownership`, persistence domains
- placement conflicts, congestion, flow timeouts
- `BlueField DPA threads`, RDMA primitives, near-memory compute

CANNOT SEE: whether bytes are decode, prefill, MoE tokens, gradients, checkpoint records, or CXL-memory migration.

→ **No single layer can express a policy that combines these facts.**

The host CPU sees everything — too late, too coarse, too far from the wire.

It is the wrong critical-path policy point for most per-event decisions.

01

Adds latency

Host-mediated control round-trips through the CPU scheduler; by the time it reacts, the queue has already built up and the slack is gone.

02

Loses event granularity

Per-WQE, per-page-fault, per-CQE events collapse into coarse host callbacks. The semantic boundary (phase change, expert ID, deadline) is gone before the host sees it.

03

Far from the wire

Pacing a congested RDMA queue or redirecting away from an overloaded expert replica must happen at the NIC/DPU or CXL switch — not at the host.

fabric_ext keeps the host as policy owner, but moves per-event decisions to the fabric hook where the decision actually executes.

One logical eBPF policy, split across fabric execution points.

Bulk data stays on RDMA / GPUDirect / CXL paths; policy decisions move to the nearest safe hook.

GPU HOOKS

- kernel_phase
- page_fault
- stream_boundary

DRIVER HOOKS

- stream_bind
- region_cap
- qp_cap

DPU / NIC HOOKS

- rdma_wqe_submit
- rdma_cqe_complete
- congestion_update

CXL SWITCH HOOKS

- cxl_place
- cxl_copy_complete
- ownership_transfer

↓

CONSISTENCY-CLASSED BPF MAPS

Telemetry · eventual | Hint · bounded-staleness | Ownership · epoch-atomic | Command · exactly-once

↓

DAMER COMPILER

`JSON events + switchlets` → movement graph → per-device `BPF C` + objects + `bpftime/dputime` manifests + verifier summaries.

CROSS-DEVICE EFFECT VERIFIER

bounded helpers · bounded `RDMA/CXL` ops per event · no GPU–fabric dependency cycles · tenant/QP/region capabilities · retry TTLs · atomic epoch handoff.

A semantic movement graph, not packet classifiers.

Nodes are where data lives or transforms can run; edges carry bytes, stride, reuse distance, ownership, and an optional transform.

GPU HOOK · BPF C — mark_phase publishes a fabric hint

```
SEC("gpu/kernel_phase")
int mark_phase(struct gpu_kernel_ctx *ctx) {
  struct fabric_hint hint = {
    .tenant   = ctx->tenant,
    .stream   = ctx->stream,
    .phase    = GPU_PHASE_DECODE,
    .deadline = ctx->deadline,
    .priority = LATENCY_CRITICAL,
  };
  bpf_fabric_publish(ctx->stream, &hint);
  return 0;
}
```

GPU hook publishes a semantic hint; a fabric hook on the DPU/NIC/CXL switch consumes it.

9 MOVEMENT ACTIONS — every edge carries an optional transform

Move plain transfer	**Move+Quantize** shrink HBM	**Move+Compress** cut bandwidth
Move+Checksum integrity	**Move+Filter** drop on GPU	**Move+Reduce** aggregate
Move+Scatter/G. fan-out	**Move+Replicate** cache copy	**Move+Persist** durable CXL

Expressive enough to cover common GPU-fabric transfers; constrained enough for effect verification.

Different policy state needs different consistency.

A shared cross-device BPF map is insufficient. fabric_ext makes consistency part of the type system.

TELEMETRY

`eventual consistency`

≈

Approximate counters like bytes moved, queue depth, retry rate. Stale is fine — the value is observational, not authoritative.

GPU + DPU + CXL write · all devices read

HINT

`bounded-staleness epochs`

Δt

Decode deadline, expert load, GPU readiness. Carries an epoch; consumers must tolerate bounded staleness.

GPU writes · DPU / CXL read

OWNERSHIP

`epoch-atomic handoff`

↔

CXL window, GPU buffer, QP capability. A stale owner violates isolation or ordering — handoff is atomic per epoch.

epoch-checked helpers only · raw writes rejected

COMMAND

`exactly-once completion`

1×

Redirect, retry, persist, reduce. Unbounded retries are forbidden — every command carries a TTL and exactly-once completion semantics.

DPU emits · TTL-bound · commit-acknowledged

14 LLM fabric workloads. 86 events compiled into 86 BPF objects.

Modeled data-movement speedup over staged host/NIC control — not a token-latency claim.

AGGREGATE MODELED E2E SPEEDUP

3.47×

after copy-path optimization · up from 2.34× baseline

DPU inline (49) + GPU pre-transform (24) + direct GPU-peer/CXL (13)

MICROBENCH COVERAGE

1,000 / 1,000

generated positives accepted

9 movement kinds · 5 node classes

FUZZ MIX PASS

512 / 512

404 accepted · 108 rejected

0 unexpected

COMPILE TIME (P50 / P95)

8.5 / 9.7 µs

lightweight enough for runtime policy iteration

TARGETED NEGATIVES

3 / 3

transform bound

RDMA/CXL fanout bound · retry TTL

LARGEST RELATIVE GAINS

Qwen long-context CXL · 4.10×

MOE ALL-TO-ALL

2.67× extra after copy-path

SCALE-OUT 2→32 GPUS

1.96×–2.76× preserved as fanout grows

The programmable object in modern GPU systems is the fabric — not an individual device.

GPU semantics + fabric state, safely combined at the hook where the decision executes.

- **Cross-layer eBPF:** one logical policy compiles into GPU, driver, DPU/NIC, and CXL-switch hooks.
- **Consistency-classed BPF maps:** telemetry / hint / ownership / command — each with its own contract.
- **Across-device effect verifier:** bounded helpers, no GPU–fabric cycles, capabilities, retry TTLs, epoch-atomic handoff.