INDIS Workshop - SC26

Characterizing Network Behavior of Distributed LLM Inference on a Four Node Edge Cluster

Manas J. Das

Department of Computer Science |
Center for Crime Science and Violence Prevention |
Southern Illinois University Edwardsville

Question from a HBMble* AI researcher / User

- **NVIDIA GB10 (DGX Spark)**: 128 GB Unified memory, fits on a desk
- A few nodes can serve a **70B model** and bigger model in your own lab
- **Datacenter habit**: buy the fastest fabric, use pipeline parallelism across nodes

GB10
273 GB/s memory ÷ 25 GB/s Network
≈ 11:1

Datacenter GPU + 400 Gb/s
3350 GB/s memory ÷ 50 GB/s Network
≈ 70:1

Question: For inference on a small cluster of GB10 machines connected by commodity Ethernet, what does the network actually do, and what does it actually need to be?

* HBM + Humble

Testbed and Method

- 4 × GB10 nodes, ConnectX-7, 2 × 100 Gb/s ports (dual rail, 200 Gb/s per node), 200G QSFP56 Cable
- 1 × MikroTik CRS812 switch, factory default (no PFC, no DCQCN), single-switch star
- **Stage 1** - Envelope: perftest, iperf3, nccl-tests under dual-rail RoCE, single-rail RoCE, TCP
- **Stage 2** - Inference (vLLM): Llama-3.1-8B, Llama-3.3-70B, Qwen3-30B-A3B (MoE)
- App latency + RDMA byte/congestion counters + GPU utilization, measured together

Model	Parallel Layout	#
Llama-3.1-8B (dense)	1 node, TP2, TP4, PP2, PP4, TP2×PP2	6
Llama-3.3-70B (dense, ≥ 2nodes)	TP2, TP4, PP2, PP4, TP2×PP2	5
Qwen3-30B-A3B (MoE)	1 node, TP2, TP4, PP2, PP4, TP2×PP2	6
Qwen3-30B-A3B + expert parallelism	TP2+EP, TP4+EP, TP2×PP2+EP	3
Configuration		**20**
× 5 workloads (decode at concurrency 1, 8, 32 & prefill)		**100**
+ TCP re-runs (Llama-70B TP4, Qwen3-30B TP4)		**10**
Total Cells		**110**

The Communication Envelope

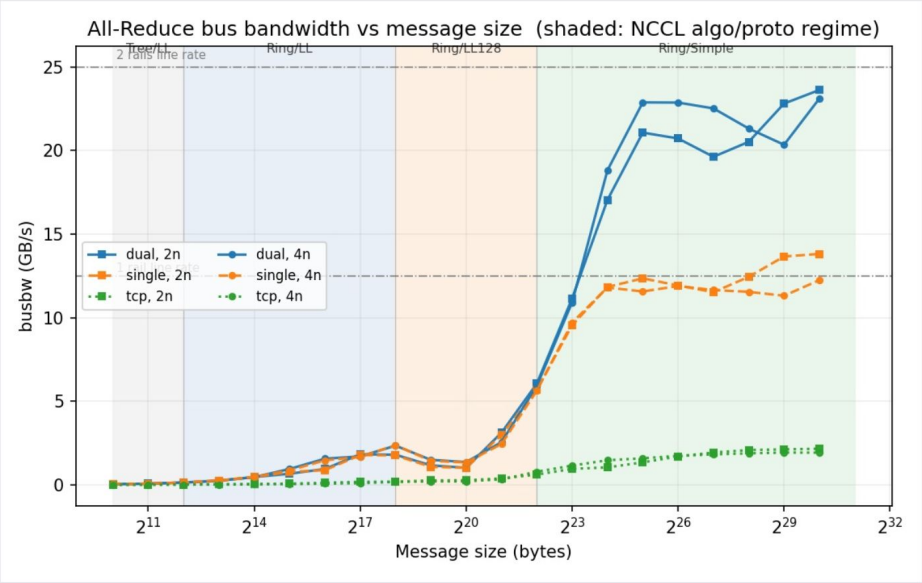


Fig. 1 — All-reduce bus bandwidth vs message size

- Dual-rail RoCE peaks at 23.1 GB/s (~92%of line rate) → fabric is healthy
- Single rail: 12.2 GB/s · TCP: 1.9 GB/s
- **Decode** sends small messages (16–128KB): RoCE ≈ tens of µs, TCP ≈ 1 ms
- Second rail only helps big messages (1 MB all-reduce: dual/single = 1.02; ~1.8–1.9×only above ~64 MB)
- Dip at 256 KB–1 MB = NCCL picking the wrong protocol (LL128), not a hardware limit

Tensor Parallelism Accelerates Decoding

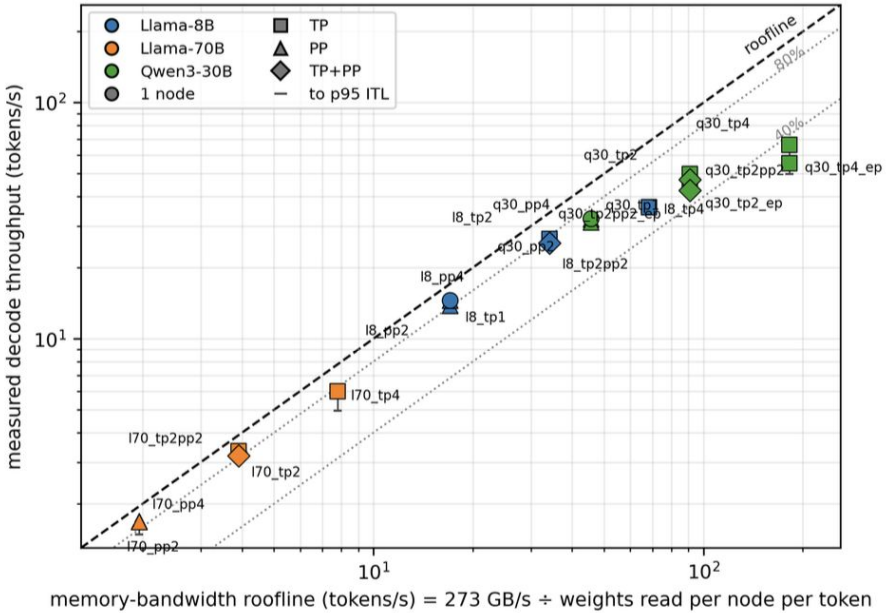


Fig. 3. Measured decode throughput versus the memory-bandwidth roofline, concurrency 1. One-sided whiskers extend to the throughput implied by the 95th-percentile inter-token latency (tail latency, not a confidence interval).

- Decode reads every weight once per token
  → roofline = 273 GB/s ÷ weight bytes per node per token
- Single-node & PP reach **0.69–0.86** of roofline → decode is memory-bound
- TP shards weights → each node reads less → faster:
  - Llama-8B TP4 **2.47×**
  - Qwen3-30B TP4 **2.05×**
  - Llama-70B TP2→TP4 **1.79×**
- PP: 0.95–1.00× (each token still walks through every layer in order)
- **Reverses** the usual advice (DeepSpeed Inference: use PP across nodes)

Bandwidth Isn’t the Bottleneck, Latency Is

- Decode traffic is modest in terms of the 200 Gb/s link
- Prefill is burstier: peaks ~40 Gb/s (~20%)
- Zero congestion events in all 110 cells (no ECN marks, no out-of-sequence packets) on a factory-default switch
- Latency still matters: TCP penalty largest for the least compute-bound model (Qwen3-30B **7.07×** vs Llama-70B **3.43×**)

Measurement Level	RoCE vs TCP
Microbenchmark (32 KB all-reduce)	~26×
Isolated communication time	5–9×
End-to-end token latency	**2.0–7.1×**

When the Second Rail Engages

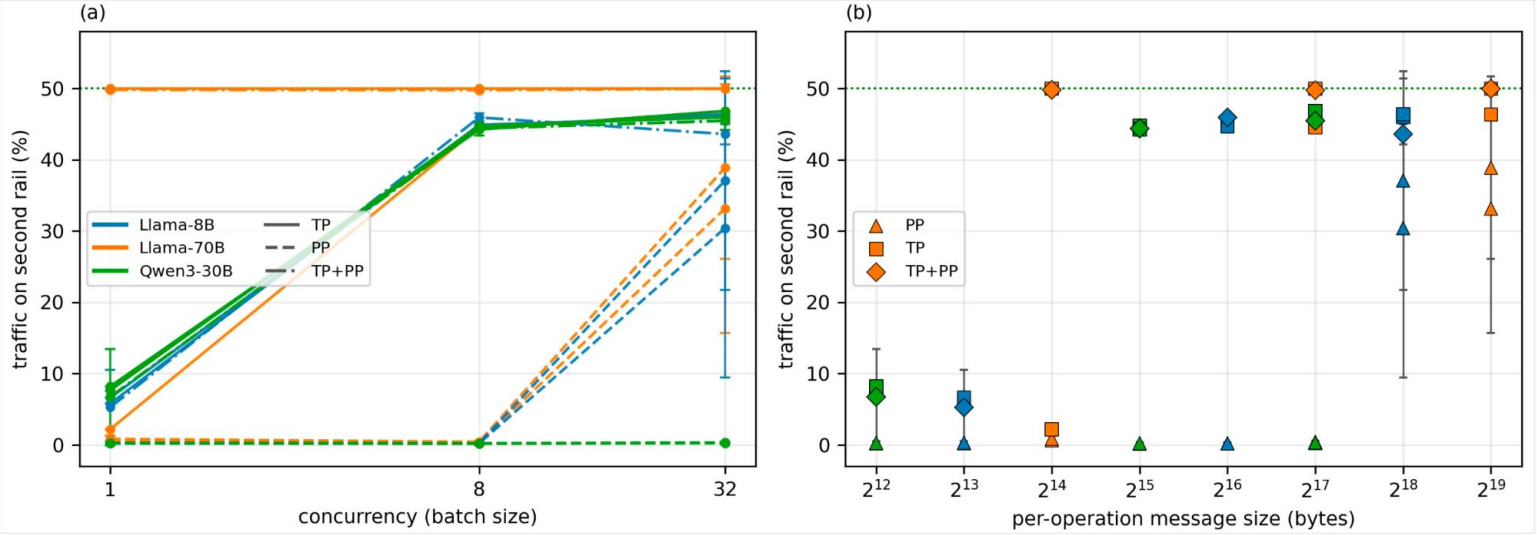


Fig. 5 — Share of traffic on the second rail: (a) vs concurrency · (b) vs message size

- 4-node TP: 2.2–8.2% at batch 1, ~even split by batch 8 · PP: only at batch 32 (≥ 256 KB)
- Utilization ≠ benefit: at 32–128 KB collectives are latency-bound; the second rail helps only above ~1 MB

Expert Parallelism Surprise

- Expected (from literature): EP creates all-to-all dispatch traffic
- Observed: inter-node traffic changed < 0.5% with EP on vs off
- Why: vLLM's default `allgather_reducescatter`backend moves the same bytes as an all-reduce
- EP even added 7–16% inter-token latency (kernel overhead, load imbalance)

Lesson: size the network for the backend you actually deploy (DeepEP etc. not tested)

Limited Takeaways from Preliminary Results

1
Don’t overprovision link bandwidth

2
Prefer tensor parallelism

3
Do use RDMA (RoCE)

4
Comodity switch was enough

Thank you | If you have any questions let’s chat over lunch!