

RDMA for AI

ECE/CS598HPN

Radhika Mittal

Acknowledgements: slides made with help from AI (Claude)

Today's paper: SpectrumX from NVIDIA

- This is a paper on congestion control written by NVIDIA (Mellanox)!
- Traditional networks had clean separation of concern.
 - network topology, network hardware, and congestion control mechanisms evolved more or less independently.
- Stronger co-dependence with hardware offloads.
- Flexibility and updates still needed.
- Reliance on a single vendor.

Distributed Machine Learning Training

- Training split across multiple GPUs (across multiple servers).
- Two forms of parallelism:
 - Data parallelism: too much data
 - each GPU operates on different shards of data.
 - each GPU holds full copy of the model.
 - Model parallelism: too big model.
 - model is split across GPUs, each GPU works on same data.
 - Pipeline parallelism: model too deep (too many layers)
 - Tensor parallelism: layer too wide.
- In practice today: hybrid – data, tensor, and pipeline parallelism.
 - 3D parallelism
 - Recent frameworks: Megatron (NVIDIA), Deepspeed (Microsoft)

Distributed Machine Learning Training

- Communication Collectives:
 - inter-GPU communication abstraction
 - been around for a while:
 - part of MPI (message passing interface) standard

Distributed Machine Learning Training

- Examples of collectives:

1. Broadcast

One process sends the **same data** to all others.

Before:	P0: [A]	P1: [_]	P2: [_]	P3: [_]
After:	P0: [A]	P1: [A]	P2: [A]	P3: [A]

Used for: Syncing initial model weights, sharing hyperparameters, broadcasting batch metadata.

Distributed Machine Learning Training

- Examples of collectives:

2. Scatter

One process sends **different chunks** to each process.

Before:	P0: [A B C D]	P1: [_]	P2: [_]	P3: [_]
After:	P0: [A]	P1: [B]	P2: [C]	P3: [D]



Used for: Distributing dataset shards, sending different micro-batches to pipeline stages.

Distributed Machine Learning Training

- Examples of collectives:

3. Gather

One process **collects chunks** from all others — inverse of scatter.

Before: P0: [A] P1: [B] P2: [C] P3: [D]

After: P0: [A|B|C|D]

Used for: Collecting predictions, assembling sharded outputs on a root process.

Distributed Machine Learning Training

- Examples of collectives:

4. All-Gather

Every process ends up with **all chunks** from all processes.

Before:	P0: [A]	P1: [B]	P2: [C]	P3: [D]
After:	P0: [A B C D]	P1: [A B C D]	P2: [A B C D]	P3: [A B C D]

Used for: ZeRO-3 parameter reconstruction before forward pass, Megatron tensor parallel weight gathering.

Distributed Machine Learning Training

- Examples of collectives:

5. Reduce

One process collects data from all and **applies a reduction** (sum, max, avg).

```
Before:  P0: [A]    P1: [B]    P2: [C]    P3: [D]
After:   P0: [A+B+C+D]
```

Used for: Collecting loss values to a master process, aggregating metrics.

Distributed Machine Learning Training

- Examples of collectives:

6. All-Reduce

Every process gets the **reduced result** — most important collective in deep learning.

Before:	P0: [A]	P1: [B]	P2: [C]	P3: [D]
After:	P0: [A+B+C+D]	P1: [A+B+C+D]	P2: [A+B+C+D]	P3: [A+B+C+D]

Used for: Gradient synchronization in data parallelism (DDP), Megatron tensor parallel output combination.

Distributed Machine Learning Training

- Examples of collectives:

7. Reduce-Scatter

Every process contributes data; result is **reduced and scattered** so each process gets one chunk.

Before: P0: [A0|A1|A2|A3]
 P1: [B0|B1|B2|B3]
 P2: [C0|C1|C2|C3]
 P3: [D0|D1|D2|D3]

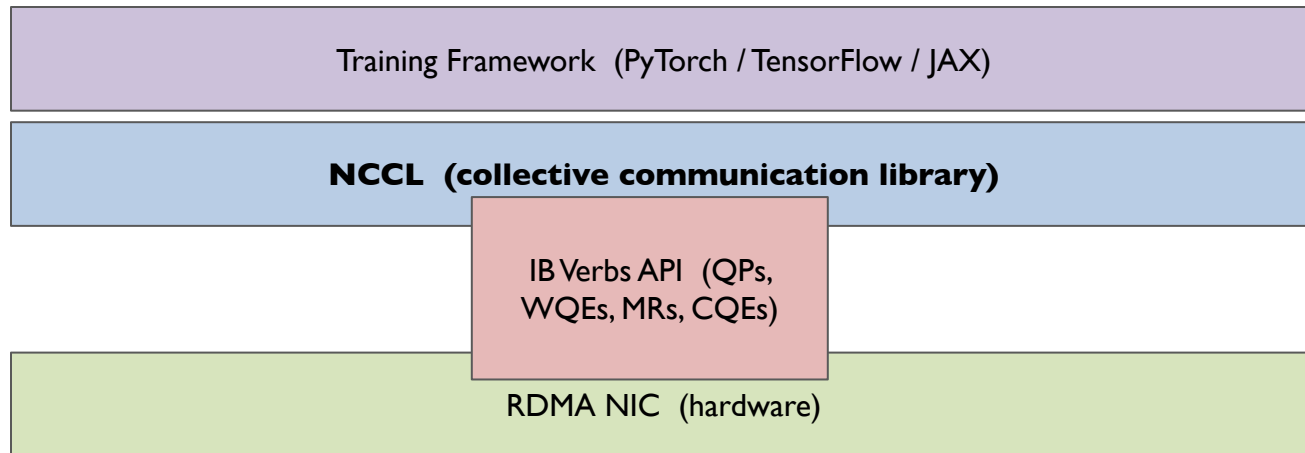
After: P0: [A0+B0+C0+D0]
 P1: [A1+B1+C1+D1]
 P2: [A2+B2+C2+D2]
 P3: [A3+B3+C3+D3]

Used for: ZeRO gradient sharding — each GPU only keeps the gradient shard it owns.

Key insight: All-Reduce = Reduce-Scatter + All-Gather

Collective Communication Library

- Builds topology-aware patterns (rings, trees) across GPUs, realizing each hop as NVLink/NVSwitch within a node, or RDMA across nodes.
- For cross-node hops, NCCL itself manages queue pairs and posts RDMA ops via the IB verbs API — it generates the verbs traffic this whole unit is about.



Realizing a collective operation

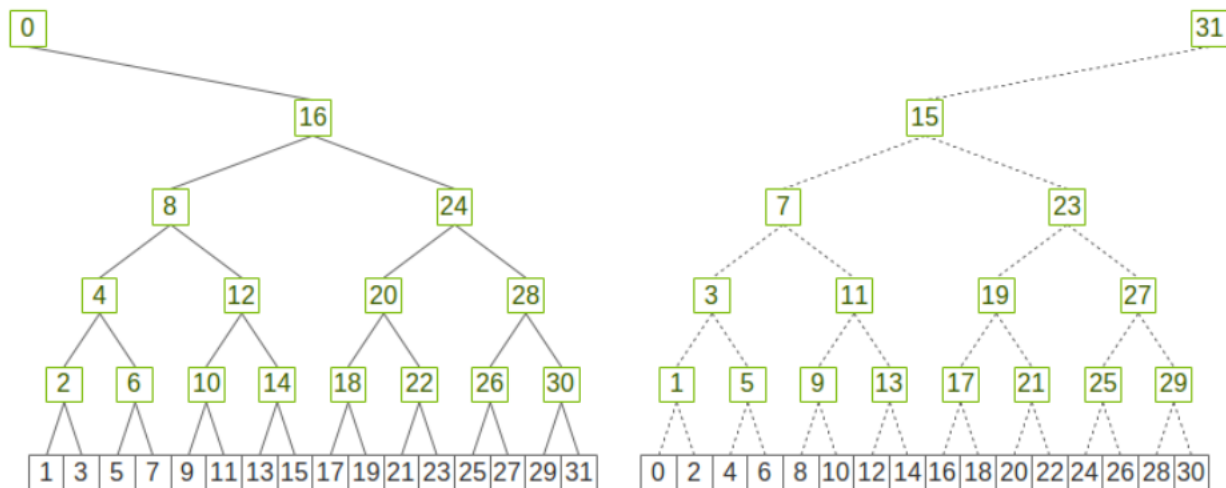
6. All-Reduce

Every process gets the **reduced result** — most important collective in deep learning.

Before: P0: [A] P1: [B] P2: [C] P3: [D]

After: P0: [A+B+C+D] P1: [A+B+C+D] P2: [A+B+C+D] P3: [A+B+C+D]

Tree AllReduce



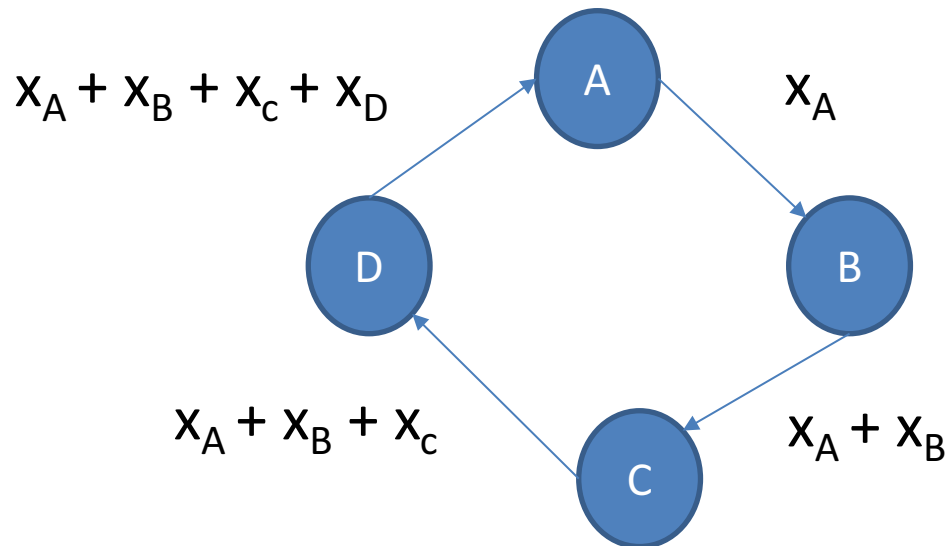
Realizing a collective operation

6. All-Reduce

Every process gets the **reduced result** — most important collective in deep learning.

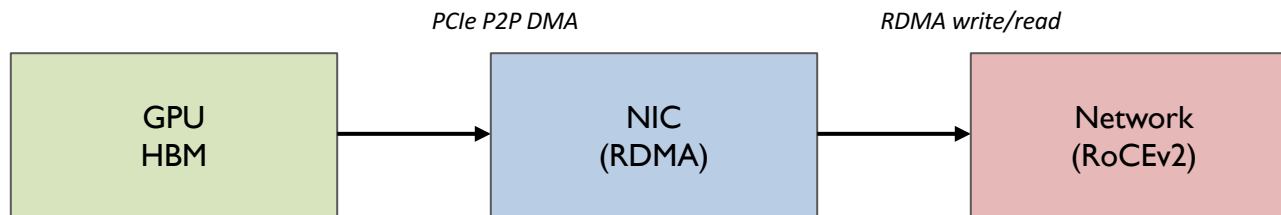
Before:	P0: [A]	P1: [B]	P2: [C]	P3: [D]
After:	P0: [A+B+C+D]	P1: [A+B+C+D]	P2: [A+B+C+D]	P3: [A+B+C+D]

Ring AllReduce



GPUDirect RDMA over RoCE

- GPU memory is registered like any RDMA memory region — the NIC DMAAs directly to/from GPU HBM.
 - Host CPU and system RAM are bypassed entirely on the data path.
- Same verbs / QP / PFC / DCQCN machinery.
- NCCL issues the RDMA writes/reads that implement AllReduce, AllGather, All-to-All across GPUs.



Today's paper: SpectrumX from NVIDIA

- This is a paper on congestion control written by NVIDIA (Mellanox)!
- Traditional networks had clean separation of concern.
 - network topology, network hardware, and congestion control mechanisms evolved more or less independently.
- Stronger co-dependence with hardware offloads.
- Flexibility and updates still needed.
- Reliance on a single vendor.

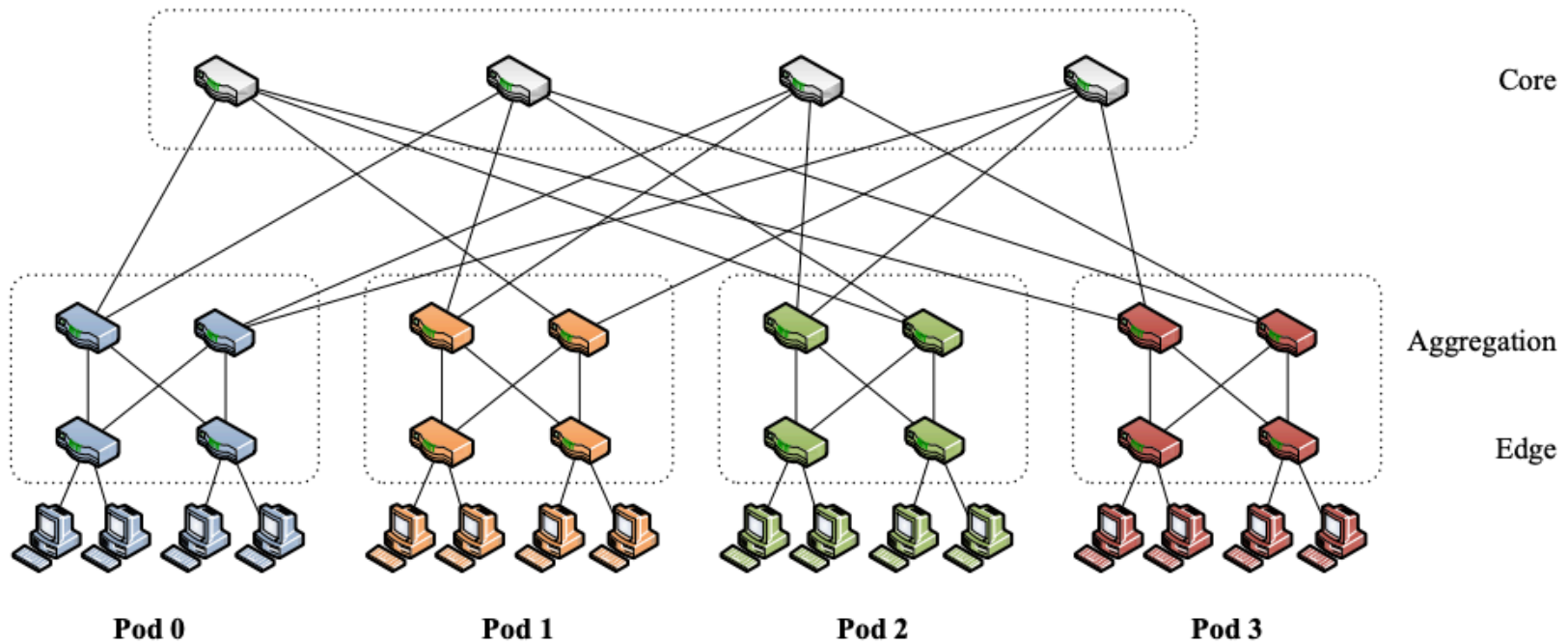
"Traditional" RoCEv2 networks

- “Single plane” fat-tree topology / Clos network.
- ECMP load balancing.
- DCQCN congestion control + PFC

"Traditional" RoCEv2 networks

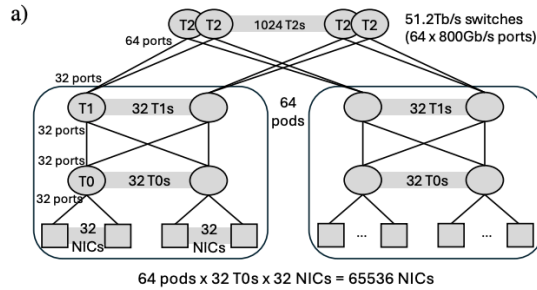
- “Single plane” fat-tree topology / Clos network.
- ECMP load balancing.
- DCQCN congestion control + PFC

Background: Fat-tree / Clos topology

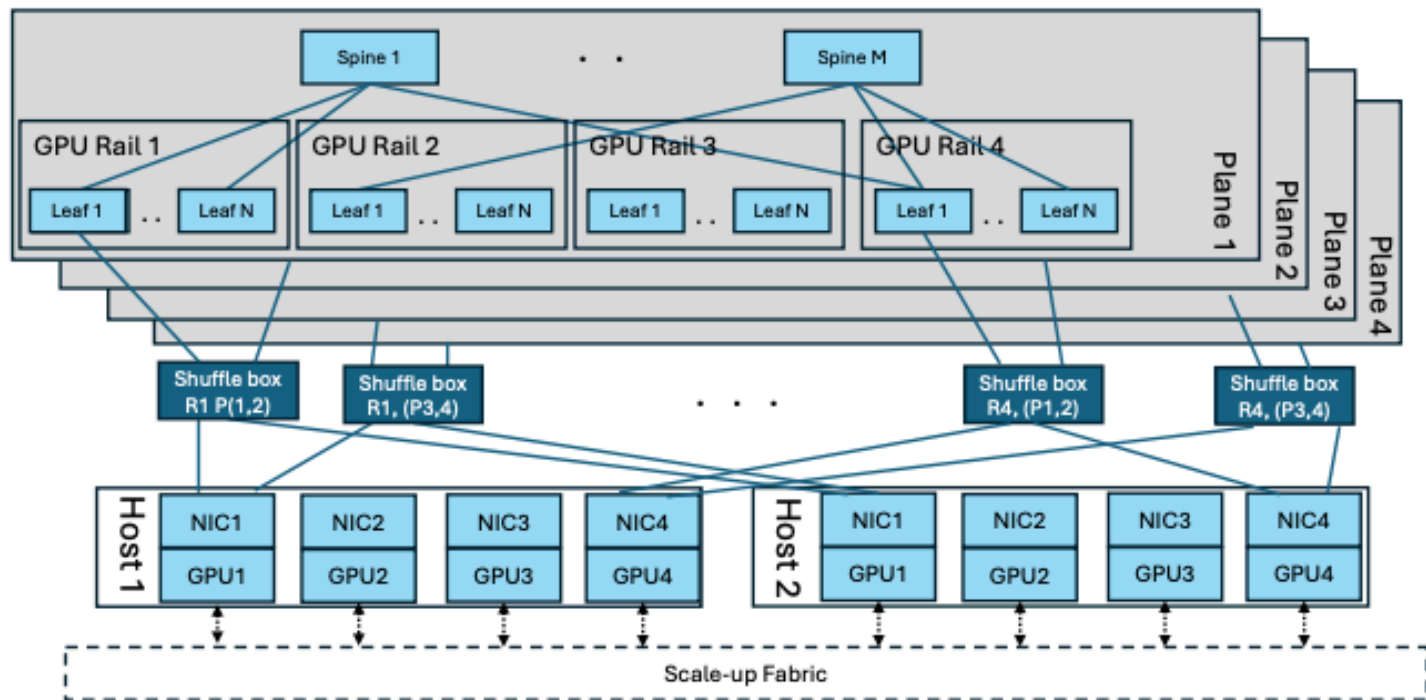


RDMA for AI: Multi-plane Design

	3-Tier Clos (single 800G plane)
Ports per 51.2 Tb/s switch	64 ports @ 800 Gb/s
T0 fan-out (down / up)	32 NICs / 32 to T1
Scale reached	Pod = 1,024 NICs; cluster = 64K NICs (T2, 64 pods)
To cover 100K GPUs	Need a 4th tier, OR oversubscribe



Spectrum X

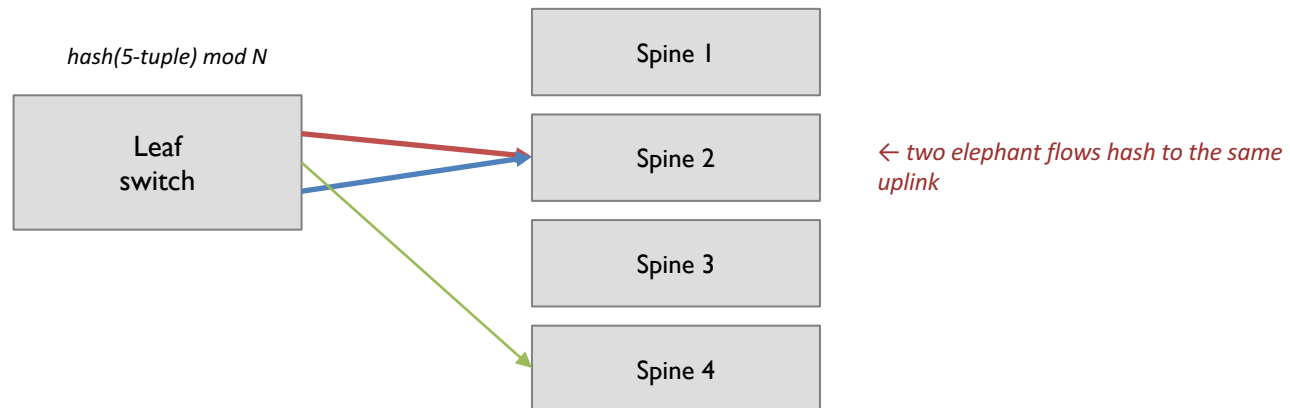


"Traditional" RoCEv2 networks

- “Single plane” fat-tree topology / Clos network.
- ECMP load balancing.
- DCQCN congestion control + PFC

ECMP

- Equal-Cost Multi-Path routing — the standard way a Clos/fat-tree fabric spreads traffic across its many equal-cost leaf–spine paths.
- Each switch hashes the “5-tuple” (src/dst IP, src/dst port, protocol) to deterministically pick one next hop.
 - a flow always takes the same path (no reordering).
 - different flows spread across different paths.
- Oblivious to load: the hash is fixed regardless of how busy each candidate path is.
 - Can lead to elephant flows colliding.



Three Hardware Control Loops

1. Switch Adaptive Routing (AR) **Intra-plane**

Per-packet, stateless, quantized Join-Shortest-Queue within a plane.

Switches sample queue depth at sub- μ s intervals;
Excludes a failed local link in ~ 100 ns.

2. Congestion Control (CC)

Per-plane, RTT + ECN based

3. NIC Plane Load Balancing (PLB)

For each packet.

1. Check if more packet's are allowed on the plane based on its CC.
2. Pick the plane with shortest egress queue.

Fault Handling, by Timescale

- Local fabric link failure → excluded by switch AR in $\sim O(100 \text{ ns})$.
- Local host-side plane failure → detected by NIC PLB in hardware (sub-ms).
- Remote plane failure → detected via consecutive RTT-probe timeouts, typically within a few RTTs.
- Persistent / permanent capacity asymmetry → handled at software control-plane timescale:
 - A BGP-based control plane computes weights proportional to healthy capacity; switch AR hardware consumes those weights.

PFC enabled

- NIC supports out-of-order placement.
- Still uses PFC. Why?
- Reordering vs packet losses.

Your thoughts?

Other industry papers on RDMA for AI

- Meta's RoCE deployment for AI (SIGCOMM'24)
 - Centralized traffic engineering, PFC without any congestion control (difficult to tune DCQCN)
 - receiver-driven traffic admission at the library level.
- Alibaba HPN (SIGCOMM'24)
 - Dual plane, two tier topology for AI workloads.; path selection at library-level.
- MRC from OpenAI (arxiv paper)
 - Multiplane topology, packet spraying, adaptive load-balancing based on ECN marks, selective retransmits, and no PFC (hard to combine with packet spraying!)
- Vela by IBM (ASPLOS'25), Alibaba Stellar (SIGCOMM'25)
 - Virtualizing LLM training, use GPU-Direct RoCE