Routing for AI training (and inference) clusters

Petr Lapukhov

Agenda

- Quick historic detour: before GPUs
- The dawn of GPU networking
- The “AI” Topology building patterns
- What’s next and more

Historic Detour

The good old days of **~2011**

No AI craze back then (only SDN)

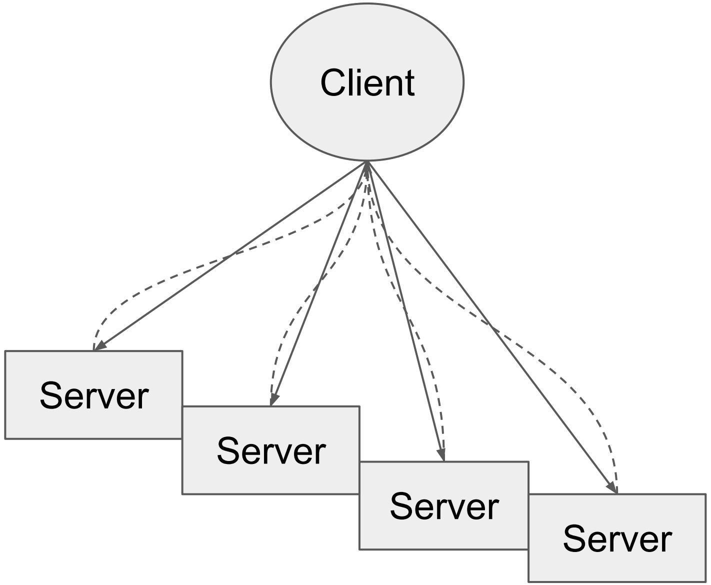
- …Yet industry already had 100K-port clusters!
- Except those were CPUs

Typical layout:

- Online-services: **query-response** traffic
- **Data-mining** (map-reduce/Hadoop)
- …

100-200W @ server → ~10-20 MW for 100K servers

Partition / aggregate

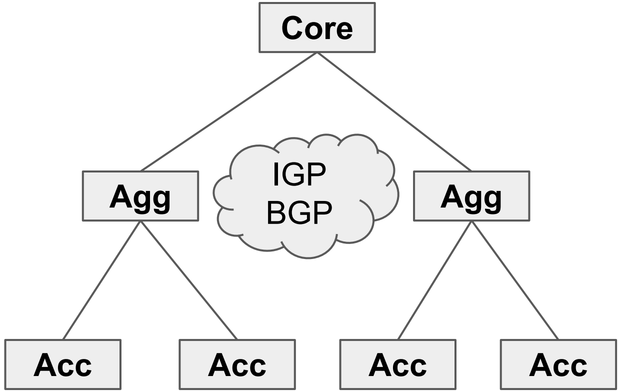


The victory of Layer-3

- L2 spanning had resiliency and scaling problems
- “Routable L2” (TRILL, FabricPath,...) did not gain momentum
- **Routed** tree architectures (IGP and/or BGP) won
- **Flow-based hashing** (ECMP) with all its joys

Surprisingly, (!) BGP became a “standard” in data-center

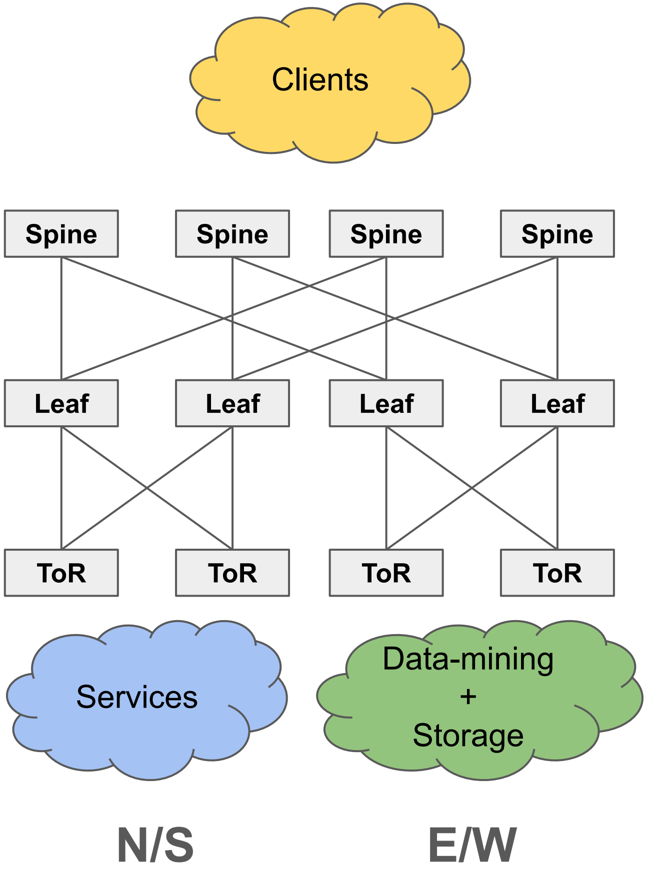
Spoiler alert: the same remained true in “AI training” clusters



The E/W traffic and fat-trees

- Map-reduce data-shuffling → **all-to-all** traffic
- Bandwidth: **fat tree** with **multi-pathing**
- Switch radix: 64-128 leading to **3- or 5- tier trees**
- Typical fan-out: 4-way or 8-way (planes)

One big "converged" fat tree for all traffic (online + data-mining + storage)



A word on the transport (~2011-2012): TCP is the king

- TCP was undisputed - with various tunings: ECN, DC-TCP…
- Memory bandwidth (memcopies) + CPU cores burning
- …but, NIC-**assisted** offloads (GSO, LRO, checksum) - to save CPU
- The **incast (fan-in)** and **speed mismatch** (e.g., 10G → 1G) problems
- Elephant flows? Not so much, because you can always add more flows

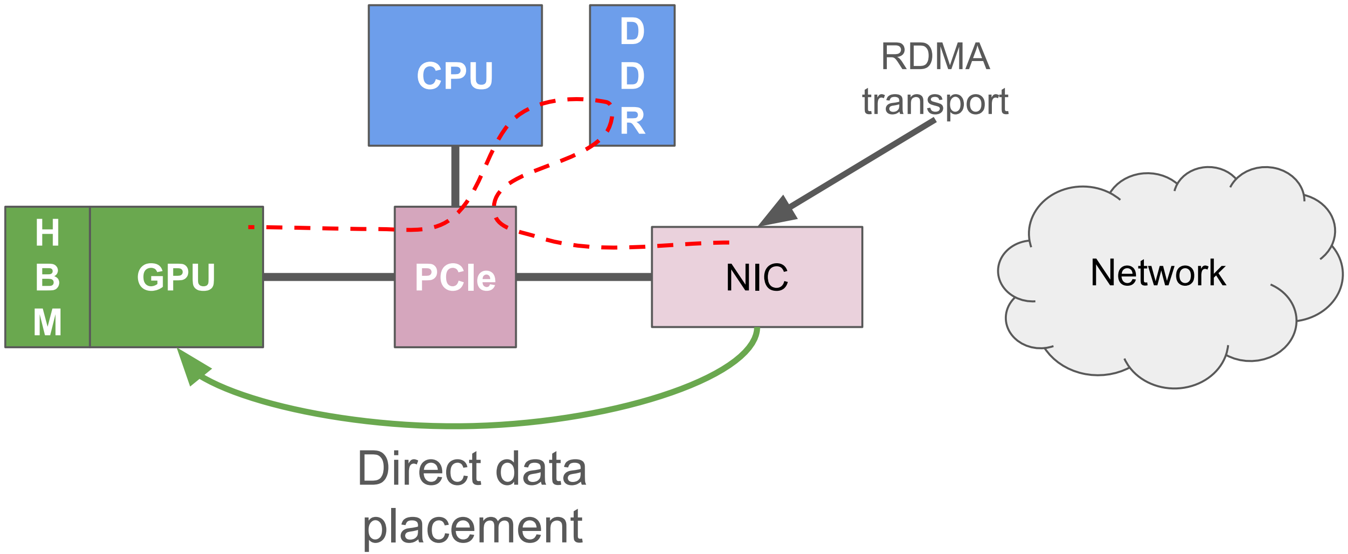
Despite TCP shortcomings, RDMA wasn’t popular.

RoCE v1 was around, mostly used for **storage**.
RoCE v2 (RRoCE) **started taking shape** (storage, again)
iWARP?

The dawn of GPU networking

From TCP to RDMA

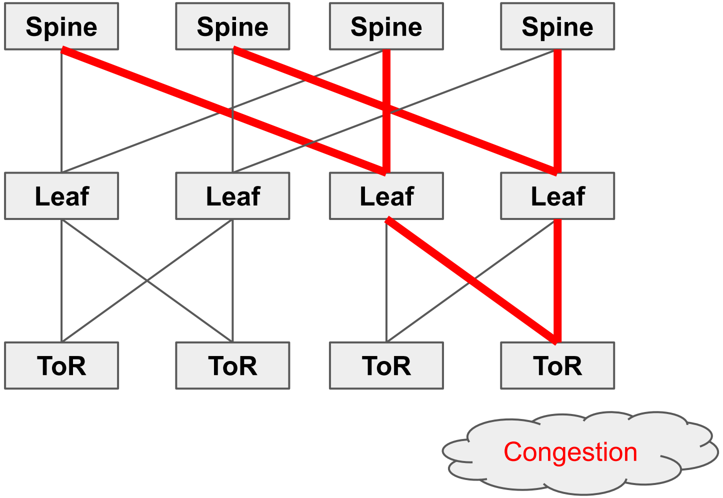
- GPU-based systems started entering scene starting 2014-2015
- Moving data **to GPU** was via CPU + NIC **via host memory** (with extra copies)
- Then came **GPUDirect + RDMA** NIC (CPU orchestrated)



From TCP to RDMA: per aspera ad astra

- **NIC-based transport** - hard(er) to inspect and debug
- Some drama: the RoCE vs IB debate (same RDMA behind)
- **Big fears of RoCE congestion spreading! (PFC)**
- …RoCEv1 was not designed to scale
- …RoCEv2 took some time to "standardize"

Debugging RDMA is much more complicated: your transport is in the NIC now



The collective communications (NCCL) [1]

Because you need the FLOPs: train/infer on many GPUs in **parallel**

- Training **parallelism** comes from either “sharding” or “replicating”
- In either case we get a ‘gang’ of GPUs communicating symmetrically

The collective comms replace Hadoop’s **map-reduce flow graphs**

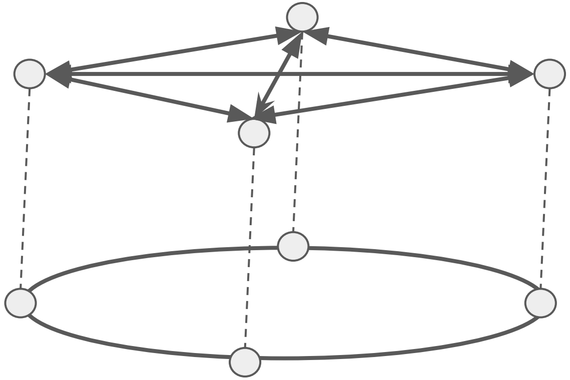
The collective communications (NCCL) [2]

We encounter familiar “logical” and “transport” patterns:

- **Logical:** All-to-all/all-reduce/all-gather/reduce-scatter
- **Transport:** ring, halving-doubling, binary trees

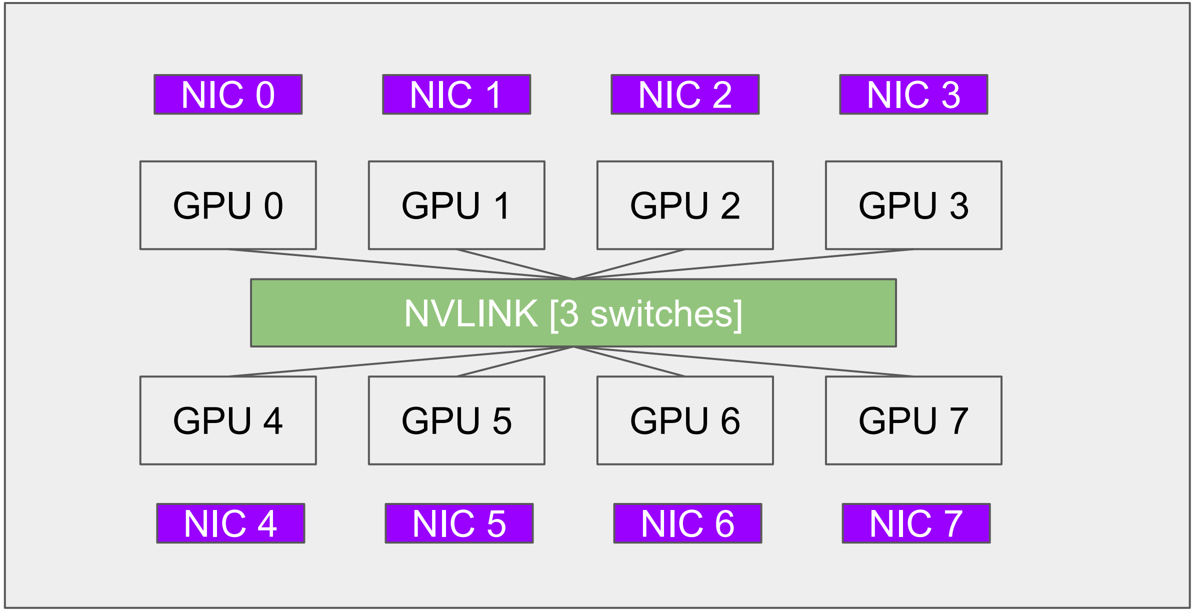
Why NCCL over… say MPI?

collective:
all2all shuffle



RDMA transport:
ring flow

A GPU node schematics (HGX H100 board)



E.g. NIC = 50 GB/s per GPU

—

NVLINK = 450 GB/s per GPU

NOTE: There are also CPUs somewhere… and PCIe switches

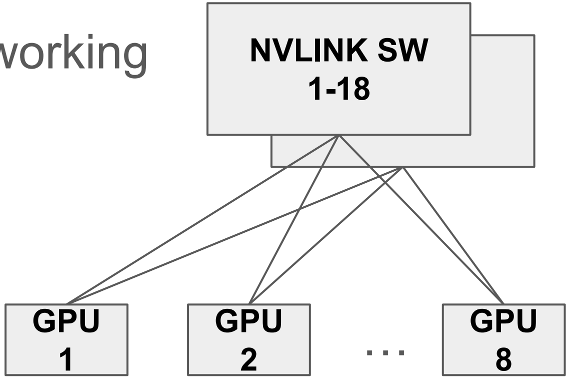
A nice bonus: the NVLink network (a mini fabric)

The “scale-up” network:

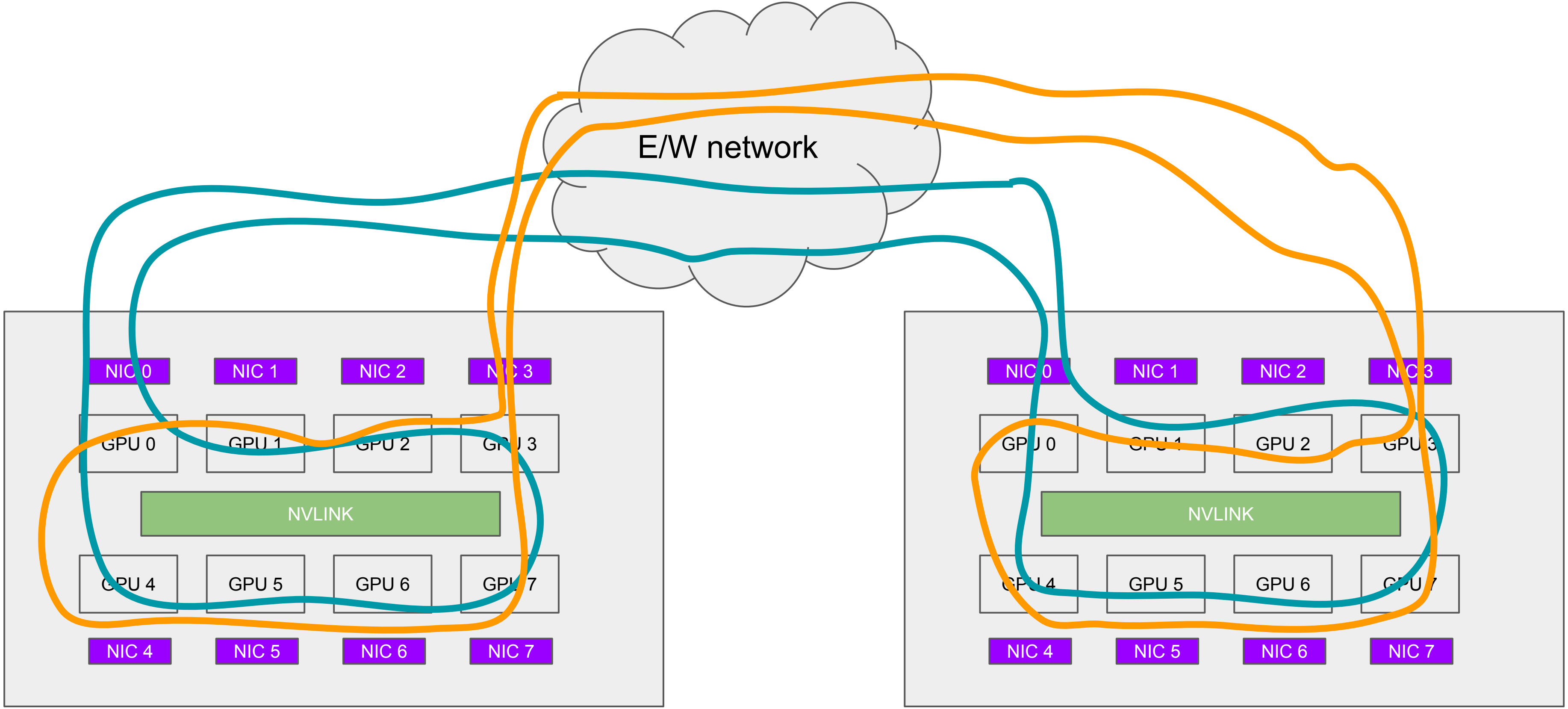
- Started as fast memory sharing link for Pascal GPUs
- Evolved into a switched network with 1 layer of switches
- … from 2 to 72 GPUs!
- Now every “rack” has a mini-fabric inside

All of that in addition to GPU-to-GPU “scale-out” (E/W) networking

Rule of thumb: scale-up bandwidth is ~9x > scale-out



A collective (all-reduce) traffic flow (ring-based)



The great fabric schism

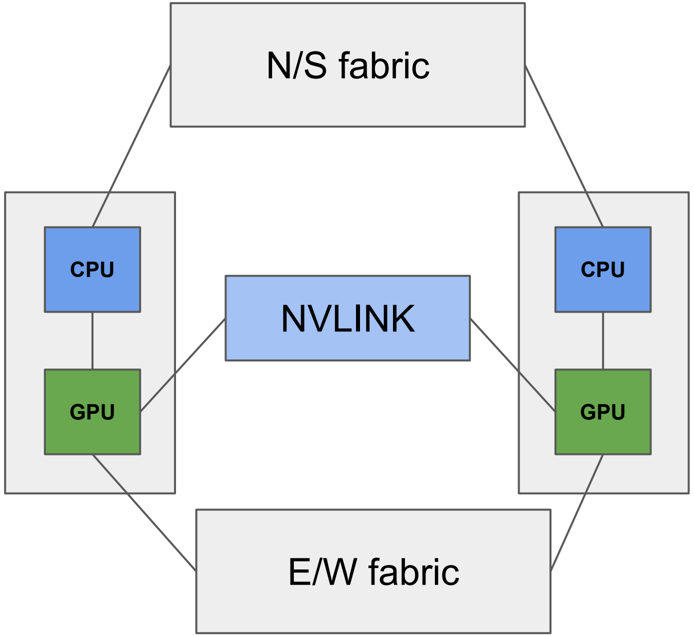
- So now we have a CPU + NIC and a GPU + NIC
- Can they use one NIC?
- Can they plug into same fabric?

Long-story short, many decided to **split the fabrics**

[1] NVLink private (always private - for now)

[2] GPU scale-out, or the **E/W fabric** (RDMA traffic)

[3] CPU **N/S** - storage and management

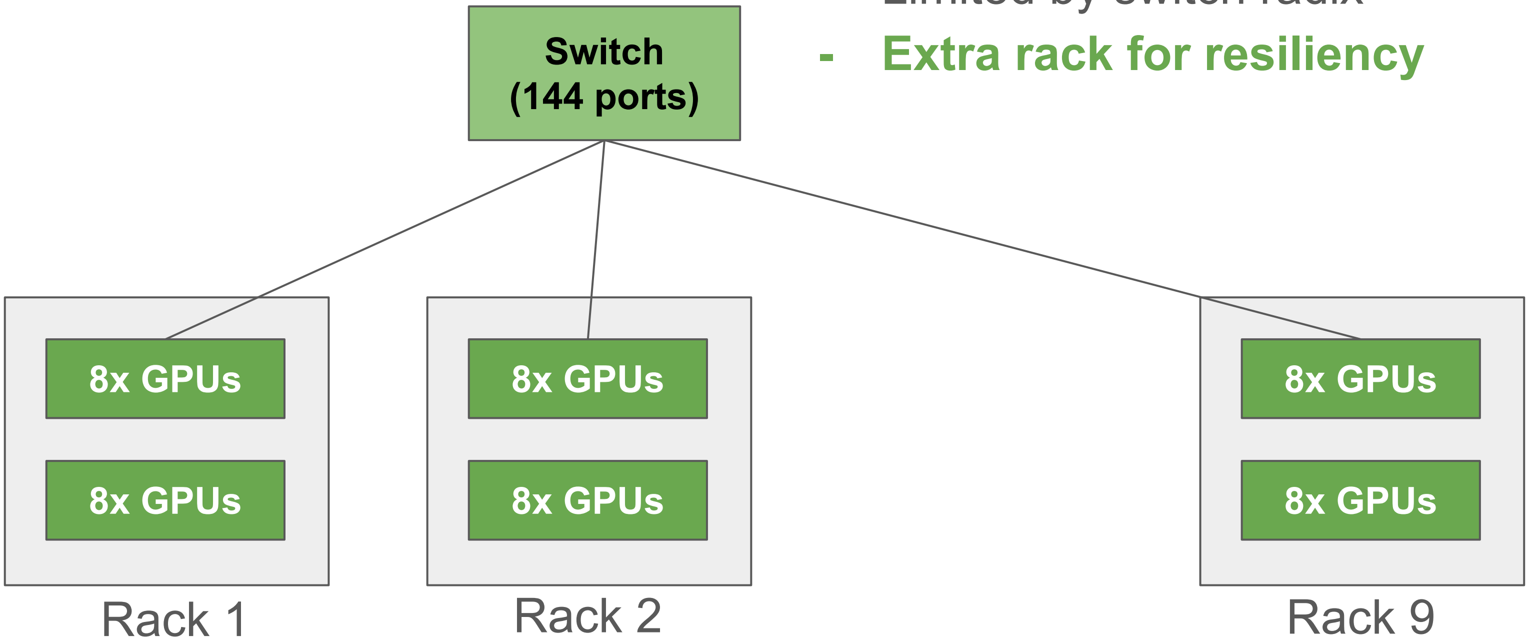


The “AI” topology building patterns

Your very first E/W cluster™

144 GPUs in one "pod:" single switch

- One-hop network to "deal" with RoCE flow control
- Does not need to be routed
- Limited by switch radix
- **Extra rack for resiliency**



The new world order and its problems (1)

So we have the new RDMA fabric… Now what? :)

The flow load-balancing problem:

- RDMA NICs push at line rate of 100G/200G/400G NICs
- **The “elephant” flows - do not play well with ECMP**

The new world order and its problems (2)

The effects from the "collective" comms:

- **Latency** accumulation in "ring-based" collectives
- **Congestion** in "log-" collectives (trees, halving-doubling) - **incast, again**

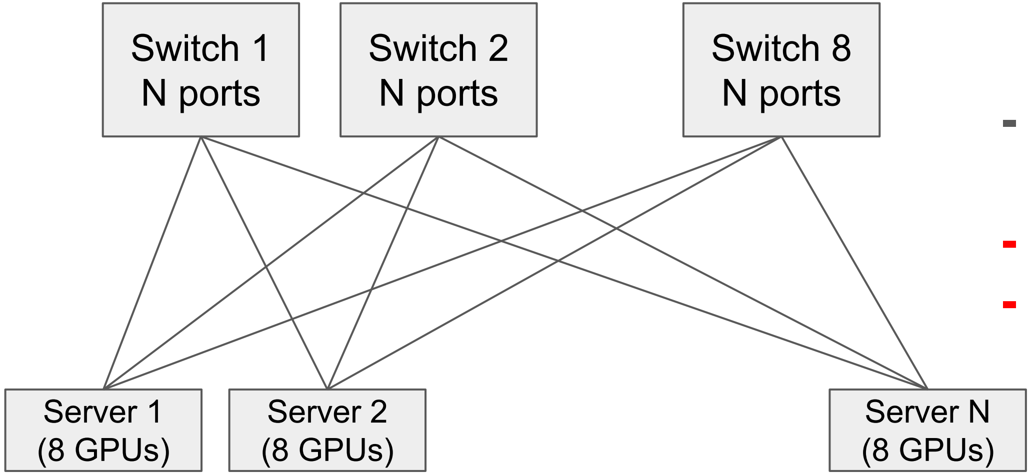
The new world order and its problems (3)

Resiliency:

- Effect of failures much more pronounced - “collectives” fail together
- E.g., a single GPU failing will take down the whole “training job”
- Capacity **losses** (link downs) more pronounced with elephant flows - **topology imbalances**

The first line of defence is “connection split” - add more flows to the network. But that only improves as ~sqrt(N) with ECMP

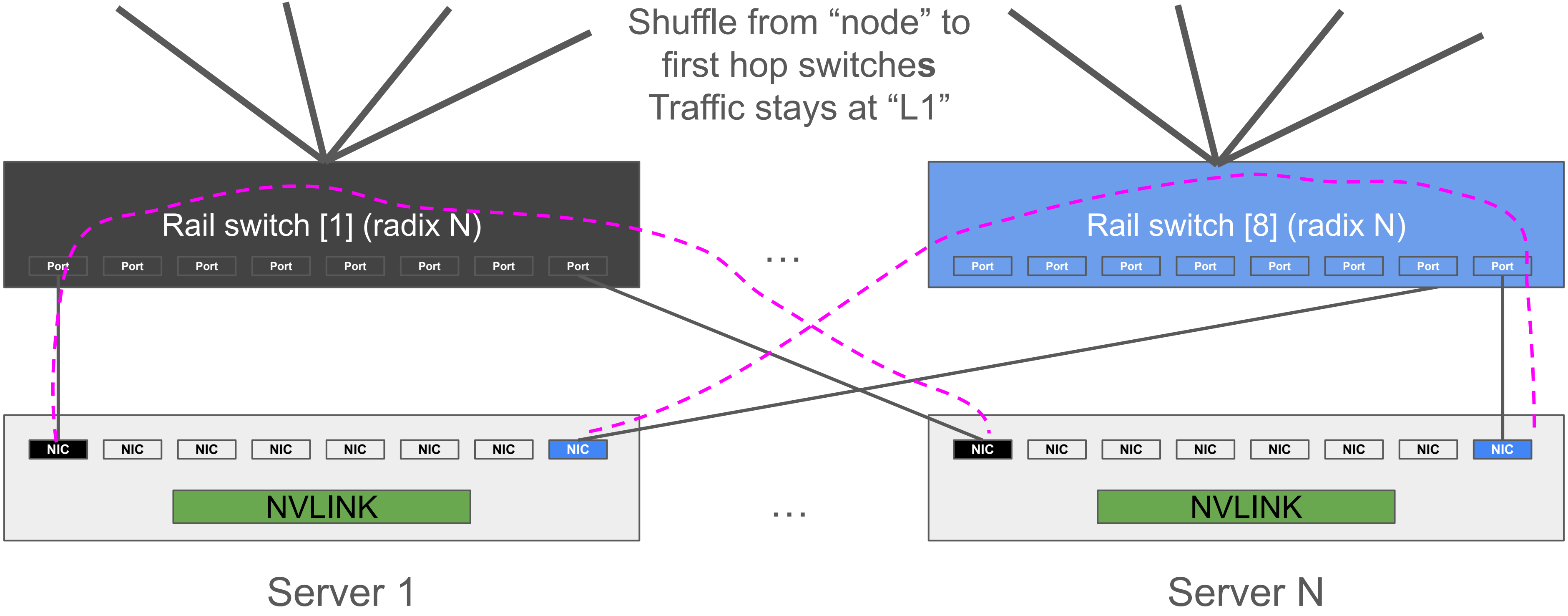
The “OG” load-balancing idea: rails



- Still one-hop for RDMA traffic
- Load-balancing by the collective library: NCCL **places flows on different rails!**
- **Rank-disjoint planes** - all-to-all traffic has to cross over NVLink
- **Optics in the NICs!**
- **No resiliency to switch failures**

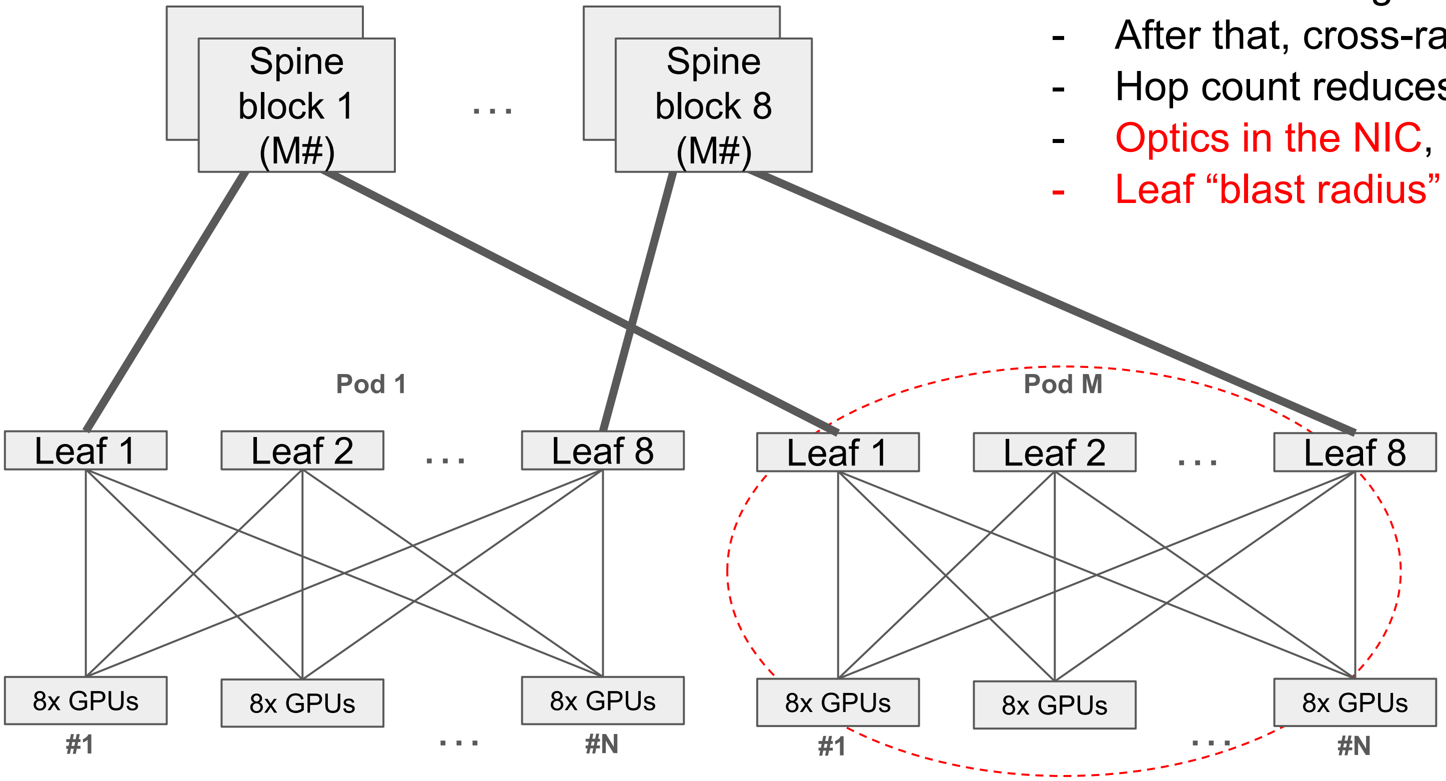
Here we get Nx8 GPUs in one “domain”

Node → Network connectivity: “Rails” style



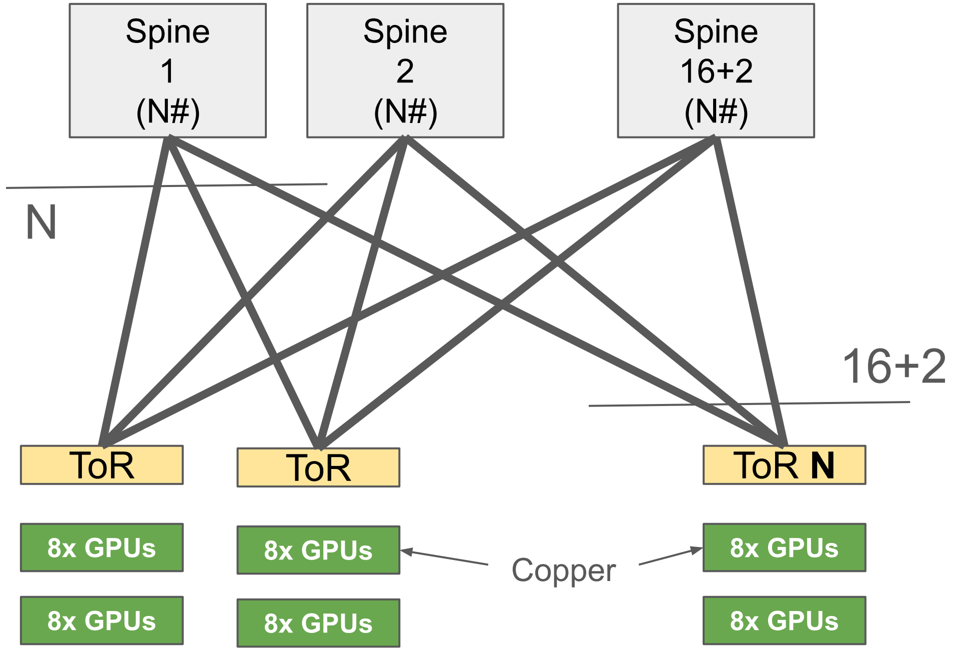
The fat tree for E/W: Rail optimized design

- Load-balancing from **servers** using rails
- After that, cross-rail “network routing”
- Hop count reduces inside “pods”
- Optics in the NIC, end-of-row leafs
- Leaf “blast radius”



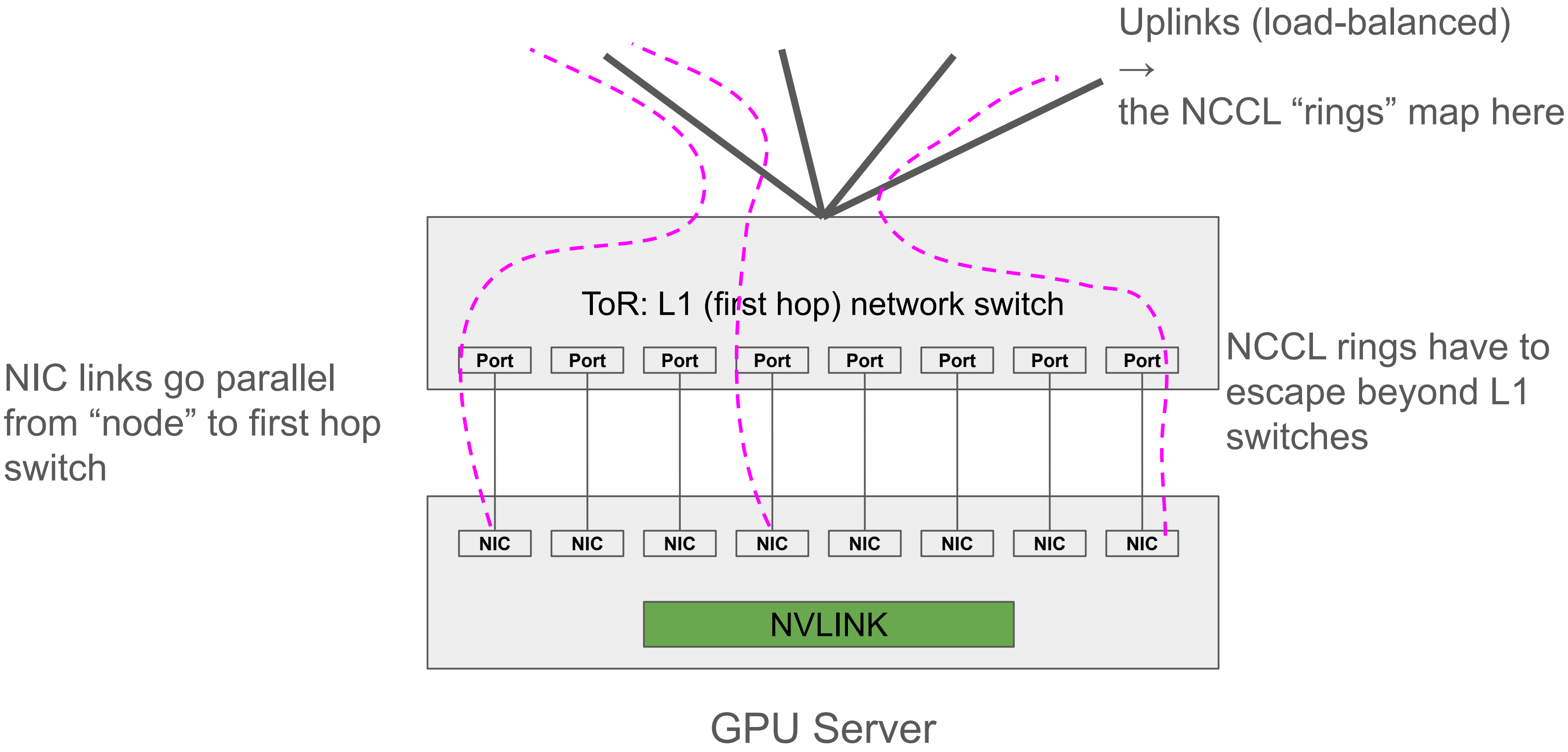
Optical links

The fat tree for E/W, redux: ToR-based design



- Requires routing + RoCEv2
- ToR does **flow load-balancing** (ECMP or something better)
- Uplinks may be 2x or more faster than downlinks - better stat-muxing
- NIC connections are **copper**
- There is more uplinks per ToR then downlinks - **resiliency**

Node to Network connectivity: “ToR” style



Fine-grained load-balancing & **adaptive routing**

By now we see that half of the problems is load-balancing :)

- Can we **split** elephant flows into smaller units?
- Flowlets, packet-spraying, etc
- Adaptive routing: distribute load based on network utilization
- Supporting out-of-order packets in the endpoint

[1] In-network vs. in-NIC

[2] Oblivious vs. adaptive

[3] No standards, really

RDMA congestion control and QoS

The “original” RDMA/RoCE needed lossy fabric: PFC

- Packet loss triggers Go-Back-N
- PFC remains an important mechanism!
- Few congestion control algorithms exist (DCQCN, ZTR) - proprietary

QoS needed to separate: CNP and NCCL RTS/CTS (rendezvous)

QoS could be useful to separate different collective types

What’s left out…

- **InfiniBand vs. Ethernet!**
- Adaptive routing + RDMA transport inter-play
- Very large clusters: resiliency and network power efficiency (CPO, LPO…)
- Very large clusters: geo-distribution - multiple buildings or regions
- What’s next for RDMA “transport”?
- Will NVLink and Ethernet/InfiniBand ever converge?

On 100K and beyond

Back to 100K clusters, but now with GPUs

Utility power becomes a precious resource

- Was ~100-200W per single-CPU, now ~1KW per GPU
- 100-150 MW for a 100K cluster!

Reliability now even more painful

- One big training job
- Resiliency can be built in training but there are limits

The network power wall

Network power starts to matter

- It’s not 10G links anymore…
- Few KW per switch
- ~ 22-25W for 1.6T OSFP

You end up with 10-20 of MW for the network for 100K+ cluster, climbing into 15% zone

The large-scale “yet-power-efficient” network

Reducing number of fat-tree tiers

- Fat-tree scale is O(N * log(N))
- Want as shallow of a tree as possible
- E.g. 2 tier fat-tree - yet covering the 100K scale

Shallow (er) fat-tree:

- Requires running switches at maximal “fan-out” - e.g. 512x ports with 51T switch*
- Now we have lots of thin links (100G?!), how we handle elephant flows?!

Requires fine-grained load-balancing from the host (NIC)