AI-ML DATA CENTER -NETWORK ARCHITECT

Backend/Frontend/Storage Fabric Design in support of 2000 GPU for AI/ML Data Center

TABLE OF CONTENTS

EXECUTIVE SUMMARY

This document presents the end-to-end network architecture for a 2,000-GPU Dell XE9680 cluster using NVIDIA Spectrum Ethernet. The design separates four networks—Backend (RoCEv2) Dual-Rail Fabrics, Frontend (VXLAN/EVPN) Fabrics, Hot Storage (NVMe-oF RoCE) Fabrics, and Management/OOB network—to meet performance, reliability, and security objectives at hyperscale. Key principles: rail isolation, non-blocking in-pod, measured oversubscription elsewhere, strong observability, and operational safety.

Primary use cases include foundation model pre-training, multi-tenant inference at scale, HPC simulation, multi-modal R&D, and large-scale fine-tuning. Scale targets: 2000 GPUs (250 nodes), 2 pods (125 nodes/pod), 8 SUs (~32 nodes/SU).

1. ARCHITECTURE OVERVIEW

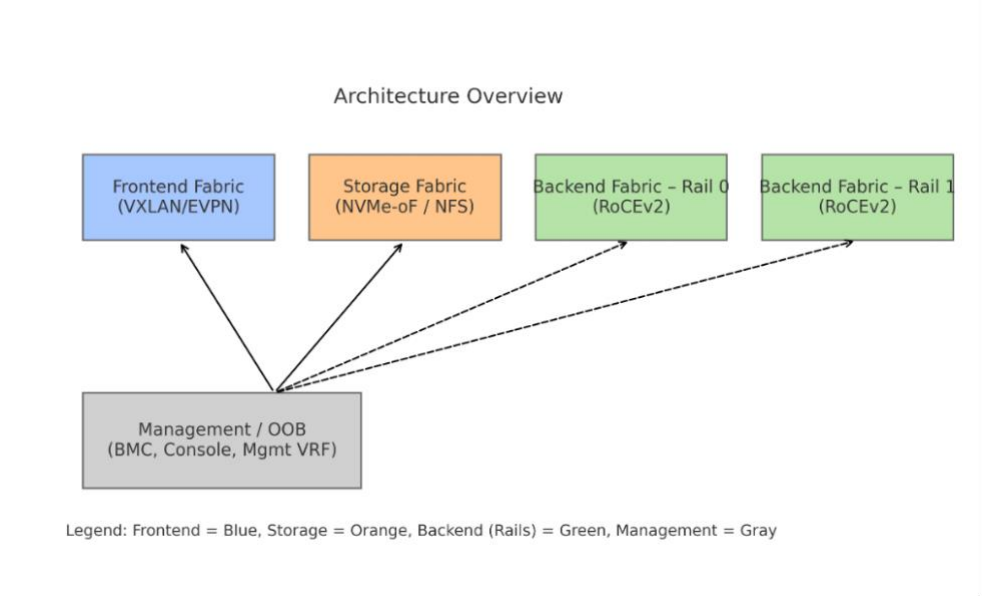


Figure 1.1: High-level, color-coded fabrics (Frontend=Blue, Storage=Orange, Backend=Green, Management=Gray)

As network architect, we need to address these unique challenges as it relates to building AI Fabric.

- Stringent latency
- Lossless fabric
- Very high bandwidth flows
- Oversubscriptions with busty traffic nature of IP network
- Security requirements

Data Center is broken down into Frontend Fabric, Backend Fabrics (Rail 0/Rail1) and Storage Fabric for various performance, traffic priority and security isolation. And in addition, Management network manage all these devices

- Backend Fabric (Rail 0 & Rail1); for Inter GPU communication (NCCL)- lossless GPU collectives
- Storage Fabric - provides hot storage (datasets and checkpoints) using GPU Direct Storage

  ** In alternate design, these two Fabrics (backend & Storage) can use same physical ports from the node (2 x400G) and hence same Fabric to keep Total Cost of Ownership (TCO) down. In that case, system relies on VRF isolation from Segmentation perspective and then QoS (ECN, PFC, DCQCN) for performance perspective. However, for AI/ML and HPC fabric it's highly recommended to keep separate Storage and Backend Fabric, and even that means saturating all high bandwidth ports for the nodes.

- Frontend provides user/API ingress, cold storage (NFS,S3, NVMo TCP) and orchestration.
- Management/OOB handles device administration and telemetry. Fabrics are physically and logically separated.

2. BACKEND FABRIC – AI/HPC CLUSTER

Dual-rail RoCEv2 CLOS. Each pod: 4 SUs × 32 nodes = 128 nodes; per rail: 4 leaves (64×400G) + 4 spines.

2.1 USE CASES

Foundation-model training, HPC collectives, distributed fine-tuning, and multi-tenant inference backplane.

2.2 ARCHITECTURE DETAILS- HIGH LEVEL BUILDING BLOCKS/DIAGRAMS

We are breaking down various connectivity blocks per Scalability Unit (SU), per Pod and for entire Fabric.

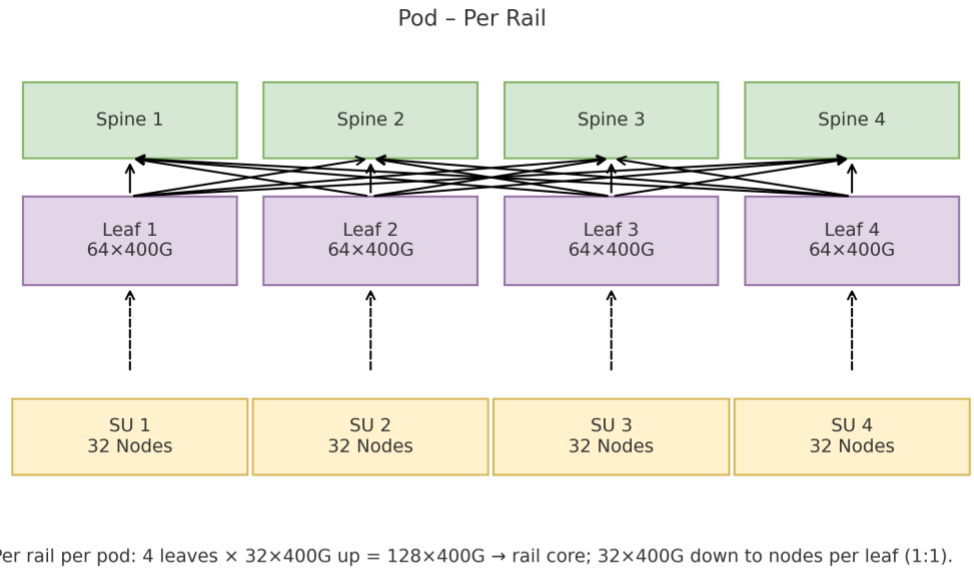


Figure 2.1: Intra-pod (per rail) — Nodes → Leaves → Spines (1:1 non-blocking to leaf, 32 up/32 down).

Even there are same # of Spines as Leaves- 4 per Pod/rail and hence total 8 per pod (16 total per BE Fabric), it makes sense to have Spine at each Pod, because

- Port efficiency at the core: With spines, you can aggregate east-west pod traffic locally, so not every flow needs to consume a core uplink.
- Failure blast radius: Spine failure impacts only that pod's external connectivity, not every leaf-core session.
- Cabling architecture: Structured trunks spine→core make for clean, repetitive cable runs vs. chaotic leaf→core distribution.
- Operational containment: You can work on leaves inside a pod without needing change control for the core row.

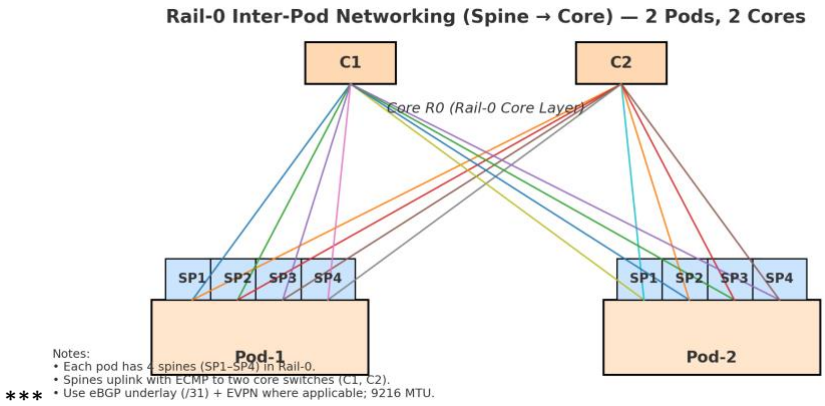


Figure 2.2: Inter-pod (per rail) — 128×400G uplinks/pod into dual-plane rail cores.

As described above, having Spine layer is highly recommended to avoid operational complexity and reduction of fault radius by having Leaf to Core connectivity. For Spine layer, we got two design options to choose from

- Option A: 4×64×400G spines
- Option B: 2×64×800G spines with 2×400 breakout (same 1:1, fewer spines)

In this design, we are sticking with Operational simplicity (less breakout cables), but Option B really provides Cost Optimization option.

2.3 TECHNICAL CONSIDERATIONS

2.3.1 ASSUMPTIONS & TUNABLES

• Dual-rail RoCEv2; rails are fully isolated (no cross-rail links).

• In-pod non-blocking (1:1); inter-pod bandwidth sized at rail core per uplink math.

• ECN/DCQCN parameters per NVIDIA guidance; validate per NIC firmware and driver.

• MTU 9216 end-to-end on RoCE class; overlay-free backend.

2.3.2 DESIGN NOTES

IPAM: /31 P2P underlay, /32 loopbacks; distinct RFC1918 blocks per rail; strict physical separation for BE0/BE1 (Rail0 and Rail1 of Backend Fabric)

• Underlay/Overlay: eBGP underlay; ECMP across spines RoCEv2 data-plane. No Overlay is needed or recommended for BE Fabric

• QoS/Isolation: RoCE class enabled with PFC; separate from storage/frontend classes; watch-dog PFC and headroom tuning.

• Buffers/PFC/ECN: Enable ECN marking at leaves/spines; DCQCN on NICs; size shared buffers for incast.

• Security: CoPP, infra ACLs, signed configs, RBAC; no cross-rail links.

• Observability: gNMI, PFC/ECN/DCQCN counters, flow telemetry; time sync with PTP

2.3.3 PROTOCOLS & BEST PRACTICES

.

Protocol / Component	Description & Function	Design Considerations & Best Practices
NVLink	NVLink: Direct copper links between GPUs in the same chassis.	In-chassis (within single Node), lossless GPU↔GPU fabric. Keep as much collective traffic here as possible.
RoCEv2	Routable RDMA over UDP/IP for low-latency GPU collectives.	Enable PFC on RoCE class only; ECN marking fabric-wide; DCQCN on NICs; ensure headroom buffers.
PFC (w/ Watchdog)	Per-priority pause to prevent drops in lossless class.	Enable watchdog PFC to break deadlocks; monitor pause counters; avoid multi-priority PFC if possible.
ECN	Marks congestion instead of dropping to signal senders.	Consistent thresholds on leaves/spines; verify NIC reaction with DCQCN.
DCQCN	End-host rate control driven by ECN marks.	Tune alpha/gain; validate ramp-up/ramp-down; align with NCCL job patterns.
eBGP**	Underlay routing protocol	This is how all RDMA endpoints (NIC) learn about each other's IP address throughout the Fabric

** Why no EVPN/VXLAN here, and just use eBGP as routing protocol?

- Backend must be overlay-free to minimize latency/jitter and avoid VXLAN encap overhead.
- Run pure L3 leaf–spine (eBGP underlay, ECMP) with L2 only at the Node edge.
- Keep PFC/ECN behavior predictable; don't carry VXLAN or mixed traffic classes on this fabric.

2.3.4 SOLUTION COMPONENTS

Component	Model / Spec	Function / Role	Notes / Best Practices
Leaf Switch	NVIDIA SN5400 (64×400G)	ToR for SU nodes; RoCE queues	Enable PFC/ECN; per-port headroom
Spine Switch	NVIDIA SN5400 (64×400G)	In-pod aggregation	Symmetric ECMP; telemetry
Rail Core Switch	NVIDIA SN5600 (64×800G → 2×400G)	Inter-pod per rail	Dual-plane; breakout planning
GPU Node	Dell XE9680 (8×GPU) + dual 400G NICs	Compute	RoCE NIC firmware tuned
NVLink Switch	Inside Dell XE9680	Internal switch	Keep GPU, NVSwitch, and NIC firmware versions in sync to avoid P2P anomalies..
Optics	400G OSFP/QSFP-DD DAC/AOC	Host↔Leaf / Leaf↔Spine / Spine↔Core	DAC in-rack; optics cross-row

2.4 INTER GPU COMMUNICATION: NVLINK & ROCEV2

PURPOSE

Enable high-throughput, low-latency GPU communication for large-scale AI/HPC training workloads by combining NVLink/NVSwitch (intra-node) and RoCEv2 Ethernet fabric (inter-node). The design ensures that most collective operations remain within ultra-fast NVLink domains, with RoCE handling only necessary inter-node transfers—both tuned for lossless operation and predictable performance.

NVLINK / NVSWITCH (INTRA-NODE)

Scope: Inside a single GPU server or chassis.
Function: Provides direct, lossless GPU-to-GPU links at hundreds of GB/s per GPU.
Topology: Fixed "islands" of 8, 16, or 32 GPUs depending on server architecture. Our solution is based on 8 GPUs per node
Design Goals:

- Maximize job placement within a single NVLink island.
- Zero packet drops/replays via vendor firmware and driver alignment.
- Maintain topology consistency across cluster.

Operational Notes for SRE/Platform team:

- Monitor link counters via DCGM.
- Align CPU NUMA to GPU mapping for locality.
- Validate with NCCL intra-node tests.

ROCEV2 BACKEND FABRIC (INTER-NODE)

Scope: Between GPU servers/chassis across the data center.
Function: Carries RDMA traffic (NCCL collectives) over Ethernet in a lossless class.
Topology: Dual-rail L3 Clos using leaf–spine–core; separate physical fabrics for compute and storage.
Design Goals:

- <300 ms failure detection/recovery using BFD with eBGP unnumbered.
- Lossless QoS: Single PFC-enabled traffic class (PCP 3), ECN tuned below PFC XOFF, DCQCN on NICs.
- Wide ECMP groups (64–128) with resilient hashing for stable NCCL flows.

Operational Notes:

- Use GSHUT for maintenance to drain traffic cleanly.
- Maintain *MTU 9216* end-to-end with appropriate FEC for optics.
- Validate with ib_write_bw, ib_read_lat, and nccl-tests across nodes.

INTEGRATION CONSIDERATIONS

Traffic Hierarchy: NCCL first uses NVLink for intra-island transfers, spilling to RoCE only when data must cross node boundaries.
Capacity Planning: RoCE bandwidth sized for spill traffic, not total NVLink capacity.
Scheduling Policy: Cluster scheduler should pack jobs within islands to reduce RoCE dependency.
Failure Domains: NVLink faults are contained to a node; RoCE faults isolated per rail.

Outcome

- A unified GPU communication strategy where:
- NVLink delivers extreme bandwidth for local collectives.
- RoCE provides scalable, lossless interconnect between nodes.
- Combined design supports predictable, high-efficiency AI training with clear operational boundaries between intra-node and inter-node networks

2.5 PERFORMANCE TARGETS & TEST CRITERIA

Metric	Target	Test Method	Pass/Fail	Notes
Intra-SU Latency	≤ 2 µs	NCCL microbench	P95 ≤ 2 µs	Leaf-local
Intra-Pod Latency	≤ 3 µs	NCCL ring/tree	P95 ≤ 3 µs	Across leaves via spines
Inter-Pod Latency	≤ 7 µs	NCCL multi-pod	P95 ≤ 7 µs	Via rail core
Loss (RoCE class)	0%	Ixia/RFC2544	No frame loss	PFC+ECN tuned
Throughput	Line-rate	Ixia/NCCL	≥ 98% of line	Steady-state

2.6 FAILURE DOMAINS

The following scenarios explain various failure within Backend Fabrics (Rail0 & Rail1)

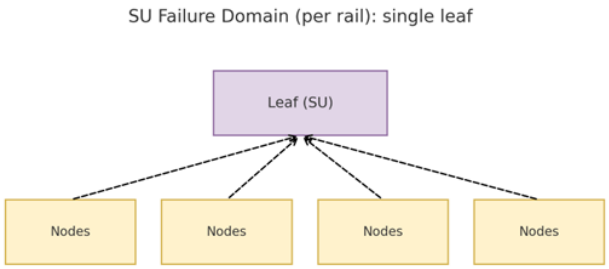


Figure 2.3: SU-level failure (leaf)

What Happens in Failures

- If a leaf switch fails, all nodes in that Scalable Unit (SU) lose connectivity on that rail.
  The SU is effectively isolated from the fabric on this rail.
- However, because each node is dual-homed, traffic automatically fails over to the second rail NIC, preserving connectivity and minimizing disruption.

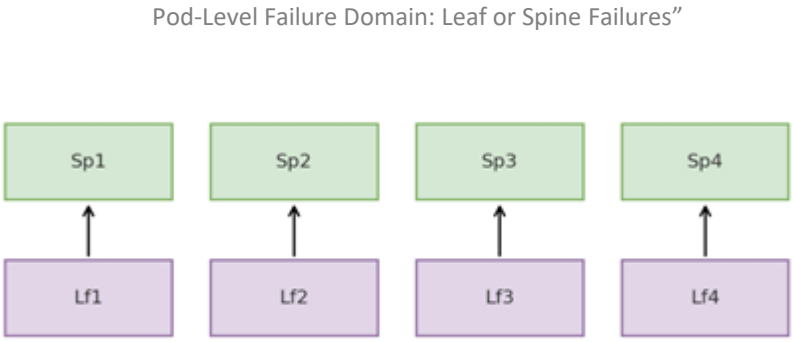


Figure 2.4: Pod-level failures (leaf/spine)

What Happens in Failures

1. **Single Spine Failure**
   - If one spine switch goes down:
   - The ECMP path count shrinks (e.g., from 4 paths → 3 paths).
   - But the fabric is still non-blocking since multiple spines remain.
   - Traffic automatically reroutes through the remaining spines.
   - Impact: Very minimal **— just slightly less path redundancy.**

2. **Multiple Leaf Failures**
    - The SU(s) connected to that leaf lose their uplinks to the fabric.
    - Those SUs are effectively cut off from the pod.
    - If more than one leaf fails, more SUs are isolated.
    - Impact: Only a subset of nodes in the pod are affected — not the whole pod.

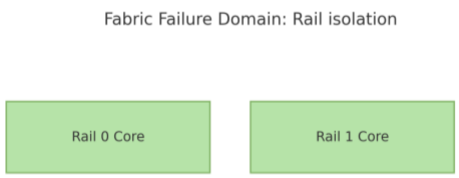


Figure 2.5: Rail isolation (fabric level)

What Happens in a Rail Isolation Failure

1. **Single Rail/Core Failure**
    - If an entire rail goes down (e.g., Core + Spine + Leaf path of Rail A):
        - All nodes lose connectivity on that rail.
        - However, each node still has its second NIC connected to Rail B.
        - Traffic fails over to the surviving rail.
    - Impact: Connectivity is preserved, but fabric capacity is cut in half.

2. **Performance Effect**

    - With only one rail, aggregate **bisection bandwidth** is reduced.
    - Large-scale collective operations (e.g., NCCL all-reduce across pods) may take longer.
    - But workloads can continue without a full outage.

2.7 NCCL COMMUNICATION PATTERNS

NCCL (NVIDIA Collective Communication Library) provides GPU-to-GPU collective communication (all-reduce, all-gather, reduce-scatter, broadcast).

- It is topology-aware → adapts to network design (single node, SU, Pod, Fabric).
- Efficient collectives are critical for distributed AI training where thousands of GPUs must exchange gradients every iteration.

Core Communication Patterns

1. Ring
   - GPUs (or nodes) form a logical ring.
   - Each node passes data to the next → full reduction or gather happens in multiple steps.
   - Efficient bandwidth use, but higher latency for large groups.
2. Tree / Hierarchical
   - GPUs/nodes arranged in a tree.
   - Reductions happen up the tree → results broadcast back down.
   - Lower latency, better scaling.
3. NCCL often hybridizes these patterns (ring inside groups, tree across groups).

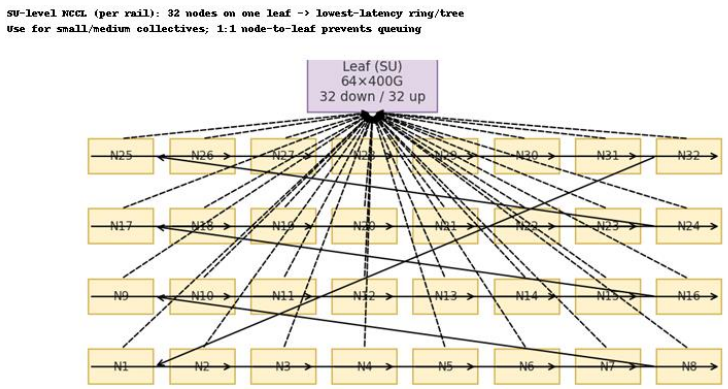


Figure 2.6: SU-level NCCL rings/trees

Pattern:

- All 32 nodes in a Scalable Unit (SU) are directly connected to one leaf.
- NCCL uses a single-ring or tree topology within the SU.
- Communication is 1 hop via the leaf switch → ensures lowest latency (≈2 µs P95) and no queuing (1:1 node-to-leaf links).

Use Case: Best for small/medium collectives (e.g., intra-SU all-reduce, all-gather).

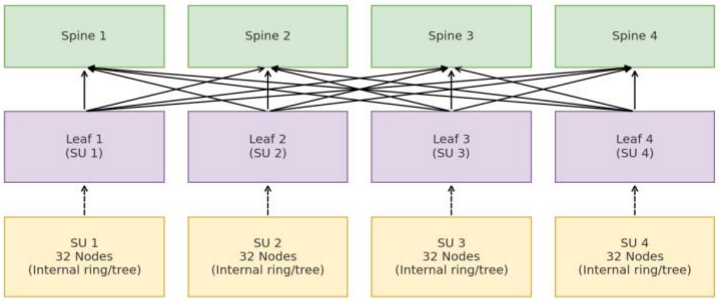


Figure 2.7: Pod-level hierarchical NCCL

Pattern:

- Each SU (32 nodes) first performs an intra-SU reduction using rings/trees.
- Partial results are then exchanged across leaves via the spine layer.
- NCCL combines these into a final all-reduce across the whole pod (e.g., 4 SUs × 32 nodes = 128 nodes).

Use Case: Hierarchical all-reduce → scales efficiently while minimizing congestion, ideal for pod-sized jobs.

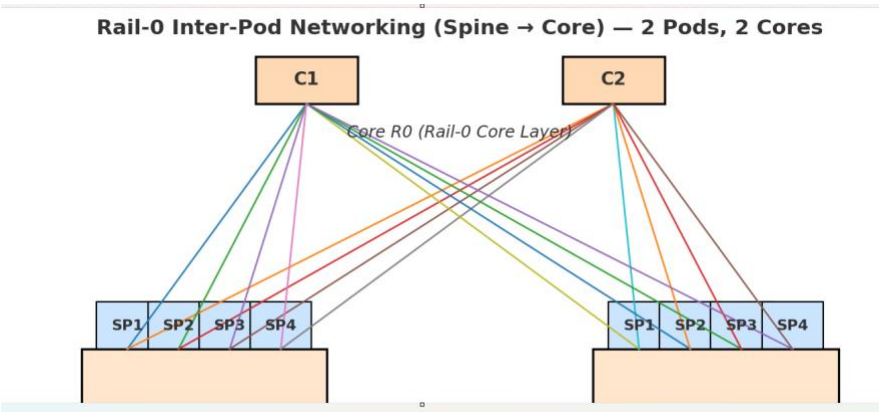


Figure 2.8: Inter-pod NCCL via rail core

Pattern:

- Each pod connects upward from its spines to the core layer.
- NCCL extends the hierarchical pattern:
    - Stage 1: Local reduce within each SU.
    - Stage 2: Pod-level reduce via spines.

- Stage 3: Inter-pod reduce via cores, combining results across pods.

- Final broadcast ensures **all GPUs across all pods** share the reduced result.

Use Case: Cluster-wide all-reduce across pods, supporting tens of thousands of GPUs.

2.8 SWITCHES & OPTICS BREAKOUT

Layer	***Per SU (per rail)***	***Per Pod (per rail)***	***Per Fabric/Rail (2 pods)***	***Total (both rails)***	***Switch Model***
Leaf	*1*	*4*	*8*	*16*	*SN5400*
Spine	*—*	*4*	*8*	*16*	*SN5400*
Core	*—*	*—*	*2*	*4*	*SN5600*

Optical-Cabling-Design in support of AI-ML Cluster.docx, has more details

2.9 SAMPLE CONFIGURATION FOR BGP

Example (Spectrum/Cumulus using NCLU; treat as reference):

```
# BGP core: eBGP unnumbered + BFD + ECMP + GR + GSHUT (ready)
net add bgp autonomous-system 65011
net add bgp router-id 10.0.11.1
net add bgp bestpath as-path multipath-relax
net add bgp graceful-restart                    # if control plan
crash/restart , the data plane continue to work (FIB)
net add bgp graceful-shutdown               # enable/disable during
drains , used during maintenance window


net add bgp neighbor SPINES peer-group
net add bgp neighbor SPINES remote-as external
net add bgp neighbor SPINES timers 60 180

for i in {33..64}; do
  net add bgp neighbor swp$i interface peer-group SPINES
  net add bgp neighbor swp$i bfd
done

# Address-family
net add bgp address-family ipv4 unicast maximum-paths 64
net add bgp
```

** *Backend (NCCL) RoCE Fabric- High Level Design.docx* has detail template for Leaf, Spine and Core switch showing detail BGP and QoS configuration.

3. FRONTEND FABRIC – API/USER/ORCHESTRATOR

3.1 USE CASES

User/API ingress, orchestration control-plane, service-to-service traffic.

3.2 ARCHITECTURE DETAILS

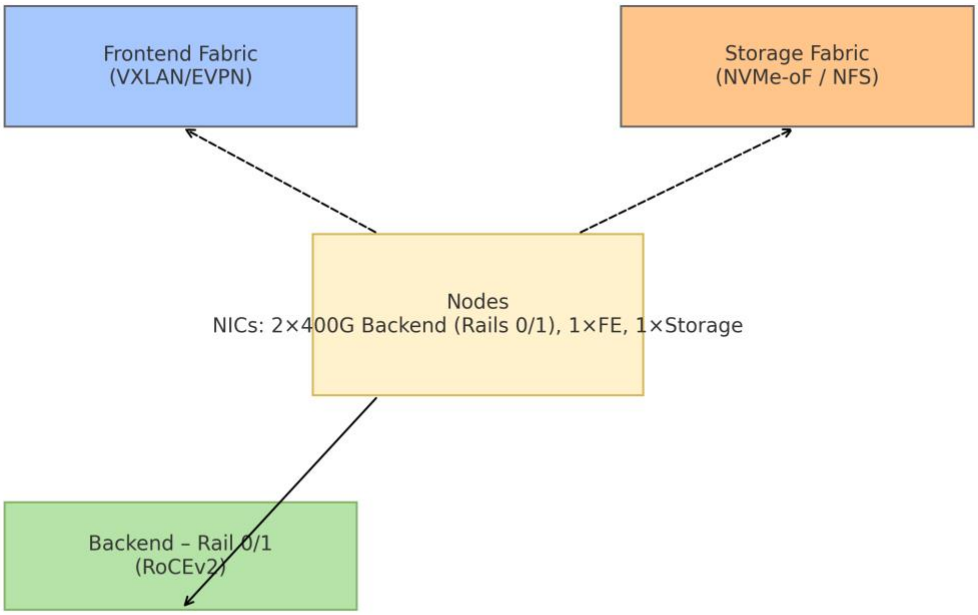
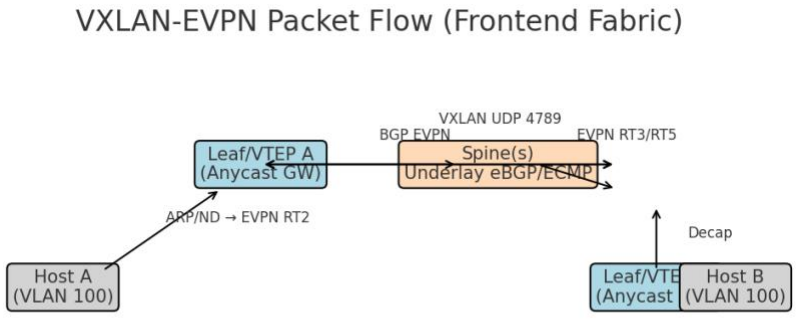


Figure 3.1: Frontend/Storage/Backend integration at node level (separate NICs)

3.3 TECHNICAL CONSIDERATIONS



3.3.1 ASSUMPTIONS & TUNABLES

- VXLAN overlay on eBGP underlay; symmetric IRB at ToR.

- Oversubscription target 3:1–4:1 at leaf↔spine; adjust per SLOs.
- Per-tenant VRFs and VNIs; VNI/RT/RD planning in IPAM.
- MTU 9216 to accommodate VXLAN overhead with headroom.

3.3.2 PROTOCOLS & BEST PRACTICES

- VXLAN/EVPN with symmetric IRB; eBGP underlay with ECMP; optional multicast for BUM if needed.
- IPAM: FE VRF with separate RFC1918 ranges; VNI allocation policy per tenant; route-target design.
- Kubernetes: Cilium (eBPF) for pod networking, Hubble for visibility; Ingress/Service LB via BGP or GWLB.
- QoS: Separate from RoCE; TCP-friendly; 3:1–4:1 oversub typical; no PFC for data.
- Security/Policy: VRF separation, micro-segmentation with NetworkPolicies; mTLS between services;
- Observability: Flow logs, eBPF metrics, gNMI from switches, Golden signals on control plane.

3.3.3 SOLUTION COMPONENTS

Component	Model / Spec	Function / Role	Notes / Best Practices
Leaf/VTEP	NVIDIA SN4600C/SN5400	Tenant L2/L3 services	Symmetric IRB; EVPN type-5 if needed
Spine	NVIDIA SN5600	Aggregation & ECMP	Underlay only
Optics	200/400G QSFP-DD	Leaf↔Spine	Plan breakouts as needed
LB	L4/7 or BGP-based	Ingress/egress load balancing	Per-tenant VIPs; DSR where applicable

3.4 PERFORMANCE TARGETS & TEST CRITERIA

Metric	Target	Test Method	Pass/Fail	Notes
Latency (P95)	≤ 500 µs DC-wide	Ixia/TCP RTT	≤ 500 µs	Under 50% link load
Loss	< 0.01%	Ixia RFC 6349	≤ 0.01%	With bursts
Throughput	As designed (3–4:1)	HTTP/gRPC bench	Meets SLO	Ingress mix

3.5 SWITCHES & OPTICS BREAKOUT

SN5400/SN4600 VTEPs; 400G→2×200G breakouts to FE core.

3.6 SAMPLE VXLAN-EVPN CONFIGURATION

```
feature nv overlay
feature bgp
```

```
interface nve1
  no shutdown
  source-interface loopback0
  member vni 10100
    ingress-replication protocol bgp

router bgp 65100
  router-id 10.255.1.1
  address-family l2vpn evpn
    neighbor 10.255.1.2 activate

interface ethernet1/1
  description TO-SPINE
  mtu 9216
  no switchport
  ip address 10.10.0.1/31

vlan 100
  name TENANT-A
vrf context TENANT-A
interface vlan100
  vrf member TENANT-A
  ip address 172.16.100.1/24
```

4. STORAGE FABRIC – NVME-OF / NFS

4.1 USE CASES

Dataset ingress/egress, checkpoints, artifact repositories, evaluation outputs.

4.2 ARCHITECTURE DETAILS

Dedicated Ethernet fabric with storage ToRs and arrays; RoCE or TCP depending on stack; size per GB/s per GPU. NVMe-oF over RoCEv2 as primary for low-latency block; This fabric will have very similar design and architecture like Backend Fabric, except the IP addresses subnets (for loopback & RDMA NICs), as well the actual application; NCCL over RoCE vs NMV-oF RoCE for this fabric.

** NFSv4.1 TCP and other warm storage traffic (NVMe-oF TCP etc,.) is going to be consolidated Frontend Fabric.

4.3 TECHNICAL CONSIDERATIONS

4.3.1 ASSUMPTIONS & TUNABLES

- Primary: NVMe-oF over RoCEv2;
- Target oversubscription ≈1:1 at leaf↔spine; adjust for GB/s per GPU.
- Separate PFC priority from backend's RoCE class to prevent coupling.
- Jumbo MTU end-to-end; validate array and NIC settings.

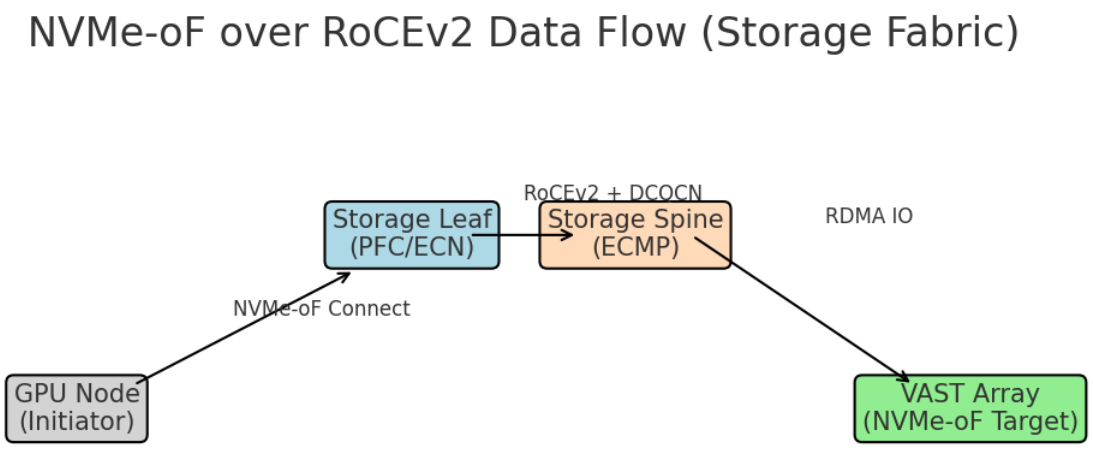
4.3.2 PROTOCOLS & BEST PRACTICES

- IPAM: Separate subnets per storage tier; jumbo MTU where supported.
- Routing: eBGP underlay; storage paths prefer shortest ECMP; no mixing with backend PFC class unless converged by policy.
- QoS: Traffic class isolated; optional PFC for NVMe-oF RoCE; ECN tuned for bursty writes.
- Security: ACLs between storage tenants; encryption in transit; export controls (NFS).
- Observability: IOps/throughput/latency exports; switch queue depth; end-to-end tracing for IO paths.

4.3.3 SOLUTION COMPONENTS

Component	Model / Spec	Function / Role	Notes / Best Practices
Storage Leaf	NVIDIA SN5400/SN3700	Array & node storage NIC ToR	Separate PFC class
Storage Array	VAST Universal Storage (NVMe-oF)	Per-backend pod scratch & dataset	1 array per backend pod (≈98 total)

Optics	400G QSFP	Node/Array↔Leaf and Leaf↔Spine	Prefer DAC in-rack



4.4 PERFORMANCE TARGETS & TEST CRITERIA

Metric	Target	Test Method	Pass/Fail	Notes
Latency (Read/Write)	≤ 200/400 µs	fio (NVMe-oF)	P95 within target	Queue-depth tuned
Loss (storage class)	0% (RoCE) / <0.01% (TCP)	Ixia/fio	Meets target	Depends on PFC mode
Bandwidth	Per-GPU GB/s target	fio/iperf3	Meets target	Parallel streams

4.5 SWITCHES & OPTICS BREAKOUT

SN5400 FE/storage cores with 400G→2×200G breakouts; host-ToR links as DAC/AOC where feasible.

4.6 SAMPLE STORAGE CONFIG (LEAF)

* this would be identical to Backend Fabric with different RDMA (access vlan at Leaf) and Loopback IP of each switches

5. NETWORK MANAGEMENT – BMC & OOB

Separate L3 OOB fabric for switch mgmt, BMCs (iDRAC), consoles. Mgmt VRF on switches; strict ACLs and MFA via TACACS+/RADIUS. Streaming telemetry via gNMI; configuration via Ansible/NEO; logs to SIEM.

6. APPENDIX

6.1 BOM & OPTICS SUMMARY

Item	Model	Qty	Notes
GPU Nodes	Dell XE9680	2,50	8 GPUs/node
Backend Leafs	NVIDIA SN5400	16	64×400G
Backend Spines	NVIDIA SN5400	16	64×400G
Backend Core	NVIDIA SN5600	4	64×800G, dual-plane both rails
Frontend Core	NVIDIA SN5400	4	200G lanes, dual-plane (depends on oversub)
Storage Core	NVIDIA SN5400	4	200G lanes, dual-plane
400G DAC/Optics	QSFP112/OSFP	~4k+	Host-leaf + leaf-spine + spine-core

6.2 CAPACITY PLANNING SNAPSHOT (BACKEND FABRIC)

Metric	Value
Uplinks per leaf (400G lanes)	32
Uplinks per pod per rail (400G lanes)	128
Total lanes per rail (400G)	256
Core capacity per switch (400G lanes)	128
Single-plane core switches per rail	2