AI Infrastructure / GPU Systems Notes

Live Lecture Notes Template

1 Module 2 — AI-Centric Data Center

1.1 Inside an AI Data Center

There is compute so that whatever request comes in, we can process it. One server may not be enough, so we need multiple compute nodes and a network to communicate between them. Then we need storage for the data.

Support infrastructure includes power, cooling, security, etc.

Key Idea

AI data center = compute + network + storage + support infra (power/cooling/security).

Key constraints when deploying high-density GPU workloads:

- **Power:** limited power capacity per rack; GPUs need a high, consistent power supply.
- **Heat:** cooling capacity of racks and rooms is a bottleneck.
- **Space:** GPUs at scale require a large physical footprint (and good airflow/layout).

1.2 New Paradigms

- **GPUs and parallelism:** scale performance by running many operations at once.
- **Chiplets:** multiple smaller chips combined for scalability (yield/cost/perf benefits).
- **3D stacking:** vertical layers to boost density (closer memory/compute).
- **AI accelerators:** custom silicon for massive speedups on specific workloads.

1.3 Data Processing Unit (DPU)

Handles **data-centric** tasks:

Networking

- packet processing
- load balancing

- overlay/underlay networking
- RDMA (remote direct memory access)

Storage

- compression/decompression
- encryption for data at rest
- RAID/erasure coding
- data dedupe

Security

- firewalling & packet inspection
- IPsec/TLS offload
- multi-tenant isolation
- zero trust enforcement

Key Idea

Why DPUs exist: offload data movement + infrastructure work from the CPU so the CPU/GPU can focus on application logic and training/inference.

CPU vs GPU vs DPU (cleaned + clarified)

CPUs are best at decision-making and control flow, but they don't scale well to thousands of data-movement tasks.

GPUs are best for AI training and parallel computation, but are not ideal for OS-level operations and general-purpose control logic.

DPUs take care of data movement and infrastructure offload. They are best at the networking/storage/security tasks above, but not good for running heavy application compute.

Unit	Best at	Not great at
CPU	control flow, OS, scheduling, general logic	massive parallel math throughput
GPU	matrix math, parallel compute, training/inference kernels	branchy logic, OS responsibilities
DPU	networking/storage/security offload, RDMA pipelines	heavy application compute

1.4 Networking

We need separate networks for different tasks. Separation helps with performance isolation, latency sensitivity, security, scaling, and failure robustness.

1.5 Network Fabric

There is **logical** and **physical** networking.

Compute fabric:

- Designed primarily for GPU-to-GPU communication.
- Implemented through InfiniBand, RoCE, or NVLink (depending on scope).
- Requirements: extremely high throughput, low latency, and scalability.

In-band management network (configuration/control within infra):

- Handles control-plane management: SSH, job scheduling, access to code repos.
- Typically Ethernet-based leaf/spine, with logical separation (VLANs).
- Requirements: reliability, security/traffic isolation, moderate bandwidth.

Out-of-band (OOB) management network:

- Provides management even if the OS is down.
- Uses a **Baseboard Management Controller (BMC)** to remotely manage the server (monitoring/logging/power cycling).
- Requirements: always available, strong security/access control, separate physical ports, low-speed switches.

Network	Purpose	Key requirements
Compute	GPU↔GPU traffic (training/inference)	**low latency**, **high throughput**, scalable
In-band mgmt	SSH, scheduling, repos, config	reliable, secure isolation, moderate BW
OOB mgmt	manage server if OS down (BMC)	**always on**, strong access control

---

1.6 Ethernet vs InfiniBand

Ethernet:

- General-purpose networking; uses TCP/IP stack.
- Universally supported.
- Typical speeds: 1–400 Gbps (depends on generation and deployment).

InfiniBand:

- Ultra-fast with specialized design for HPC.
- Lower latency; supports RDMA (bypass CPU with low overhead).
- Typical speeds: 10–400+ Gbps (depends on generation).
- Requires specialized drivers; often more expensive.

Key Idea

Simple mental model: Ethernet = universal + flexible. InfiniBand = specialized + low-latency HPC.

1.7 Converged Ethernet

Converged LAN + SAN + HPC.

- Can use RDMA over Ethernet (e.g., RoCE) to bypass CPU.

1.8 Storage

A data center uses **tiered storage**:

- **(1) NVMe SSD (local):** non-volatile local storage for fast I/O during training/inference; limited capacity but very fast.
- **(2) Clustered storage / parallel file system:** shared high-speed access across many GPU nodes in a cluster.
- **(3) Network file system (NFS):** distribute small datasets, configs, and scripts across nodes.
- **(4) Object storage:** long-term storage for massive datasets, model checkpoints, and logs.

Tier	Used for	Tradeoff
Local NVMe	hot data + fast scratch I/O	fastest, limited capacity
Parallel FS	shared high-speed access	scalable, more complex
NFS	configs/scripts/small data	simple, not for heavy throughput
Object store	checkpoints/logs/big datasets	durable + cheap, higher latency

---

1.9 Cloud vs On-Prem

Cloud:

- Accessibility: no need to buy infrastructure.
- Pay-as-you-go.
- Flexible scaling.
- Control depends on the provider.

On-prem:

- Data security and sovereignty.
- High upfront cost.
- Limited by owned hardware.
- Full control over data and systems.

---

2 Module 3 — NVIDIA Technology Stack

2.1 NVIDIA AI Leadership

2.2 Technology Stack Overview

2.3 Layer 1 — Physical Infrastructure

Here are the layers in NVIDIA infrastructure:

1. **Physical layer:** GPU, data center, NICs, DPU
2. **Data movement & I/O acceleration:** NVLink, RDMA, storage, HPC fabrics (InfiniBand)
3. **OS / drivers / virtualization:** GPU drivers
4. **Core libraries:** CUDA, NCCL
5. **Monitoring & management:** NVIDIA-SMI, etc.
6. **Applications / vertical solutions:** NVIDIA NIMs, enterprise stacks

---

2.3.1 SuperPOD

An architecture implementation of multiple DGX nodes: a scalable AI system.

Rough flow: jump box → management nodes → InfiniBand compute fabric → compute nodes → InfiniBand storage fabric → high-speed storage

---

2.3.2 ConnectX Networking

Think of NICs as high-performance interconnect adapters for InfiniBand/Ethernet, used for reliable high-speed networking.

2.4 GPU Architecture Details

2.4.1 GPU Cores

- **CUDA cores:** general compute + graphics rendering; flexible for general parallel tasks.
- **Tensor cores:** deep learning acceleration; major throughput for training/inference (often large speedups for dense math).
- **Ray tracing cores:** real-time ray tracing/light simulation (graphics realism: shadows/reflections).

Core type	What it does	Where used
CUDA core	general-purpose parallel compute	HPC, graphics, kernels
Tensor core	dense matrix math acceleration	training, inference
RT core	ray/triangle intersection + lighting	graphics

2.5 Layer 2 — Data Movement / IO

2.5.1 NVLink

Connection between multiple GPUs or (in some systems) CPU↔GPU. PCIe is slower; NVLink is a higher-bandwidth interconnect with both hardware and software components.

- NVLink bridges: consumer and professional graphics cards (some generations).
- Integrated NVLink: NVLink inside the silicon.
- NVSwitch: for large-scale multi-GPU systems.

2.5.2 InfiniBand

High-performance connection used for distributed GPU clusters (fast inter-node communication). Often paired with RDMA for low overhead and high throughput.

2.5.3 RDMA

Start with DMA: a hardware feature that lets supported devices transfer data to/from memory without the CPU acting as the middleman.

Key Idea

DMA: device ↔ local host memory without CPU copies.
RDMA: device ↔ *remote* host memory (across the network) with low CPU overhead.

RDMA is an extension of DMA: devices can bypass the CPU and access not only local host memory but also memory on another host.

Example mental model: host1 GPU → host2 system memory (via RDMA).

2.5.4 GPUDirect

NVIDIA uses RDMA to implement GPUDirect features.

GPUDirect RDMA:

- Across hosts: host1 GPU memory → RDMA NIC → network → RDMA NIC → host2 GPU memory.
- Bypasses OS, system memory, and CPU copies.
- Ultra-low-latency networking between GPU nodes.
- Needs an NVIDIA GPU + RDMA-capable NIC.

GPUDirect Storage:

- Within a host (and sometimes via storage fabric): GPU ↔ storage, bypassing CPU bottlenecks.
- High-throughput loading of large datasets into GPUs (e.g., NVMe RAID, parallel storage).
- Needs an NVIDIA GPU + NVMe/parallel storage setup.

Feature	Data path (simplified)	Goal
GPUDirect RDMA	GPU mem ↔ NIC ↔ network ↔ NIC ↔ GPU mem	low latency, avoid CPU copies
GPUDirect Storage	GPU mem ↔ NVMe / storage fabric	feed GPUs faster (I/O)

2.6 Layer 3 — OS / Drivers / Virtualization

2.6.1 GPU Drivers

Software that connects NVIDIA GPUs with the operating system and applications.

- Enables GPU acceleration for AI/HPC/graphics.
- Manages GPU resources, memory, and performance.

- Supports containers, virtualization, and multi-GPU setups.

2.6.2 Virtualization

If a physical server has a GPU and CPU, we can add a virtualization layer and create a virtual CPU/VM and a virtual GPU.

Virtualization matters because multiple users/workloads can share the same hardware efficiently. It enables isolation, flexible management, better utilization, and reproducibility.

CPU virtualization: divides CPU resources into multiple VMs, each running its own OS and applications with minimal interference.

GPU virtualization: more complex; requires techniques like GPU pass-through, vGPU, or API interception to safely share a GPU across workloads (common in cloud environments).

2.6.3 MIG vs vGPU

vGPU (software virtualization):

- Physical GPU → hypervisor (NVIDIA virtualization software) → virtual GPUs (up to many) → VMs.
- Hypervisor-based (e.g., VMware); uses guest drivers + a vGPU manager.

MIG (Multi-Instance GPU):

- Physical GPU → Linux + NVIDIA tools (e.g., `nvidia-smi`) + container runtime (Docker) → MIG instances (e.g., up to 7 on some GPUs) → containers.
- Hardware-enforced strong isolation; common for data centers, HPC, AI/ML containers.

Tech	**Isolation**	**Typical usage**
vGPU	software-enforced (hypervisor)	VM environments, cloud multi-tenant VMs
MIG	hardware-enforced	containers, HPC/AI clusters, strong partitioning

---

2.7 Layer 4 — Libraries

2.7.1 cuDNN / NCCL

NCCL: multi-GPU communication library that provides abstractions and optimizes patterns of communication between many GPUs.

Think of it like a traffic management system: it coordinates transfers and collective operations (e.g., gradient synchronization) across many GPUs.

2.8 Layer 5 — Monitoring

2.8.1 NVIDIA-SMI

Used for quick GPU monitoring and basic management on a single node. It reports GPU utilization and memory usage.

2.8.2 BCM (Base Command Manager)

Manage, monitor, and orchestrate GPU resources and workloads across an entire AI data center:

- Cluster resource utilization, infra health, job status, user quotas, workload performance.
- Management: firmware configs, power policies, software updates, provisioning, scheduling, allocation, deployments.
- Interfaces: CLI, web UI, REST API.

Installation/deployment is typically an enterprise multi-component system requiring dedicated management nodes.

Monitoring capabilities (cleaned)

- CPU, NVIDIA GPUs, NVIDIA networking, storage (NFS, clusters, NVMe), workloads (AI/HPC).

Management/config capabilities (cleaned)

- GPU firmware configs, power policies, allocation.
- Workload submission/orchestration through Slurm/Kubernetes.