GPU Clusters

Hardware, Software, Runtime

2025

Обо мне

- Бакалавриат ММП
- **YandexGPT** Pretrain Team 2+ years
- **Together AI** Model Shaping (finetuning) team

https://www.linkedin.com/in/vorobyov01/

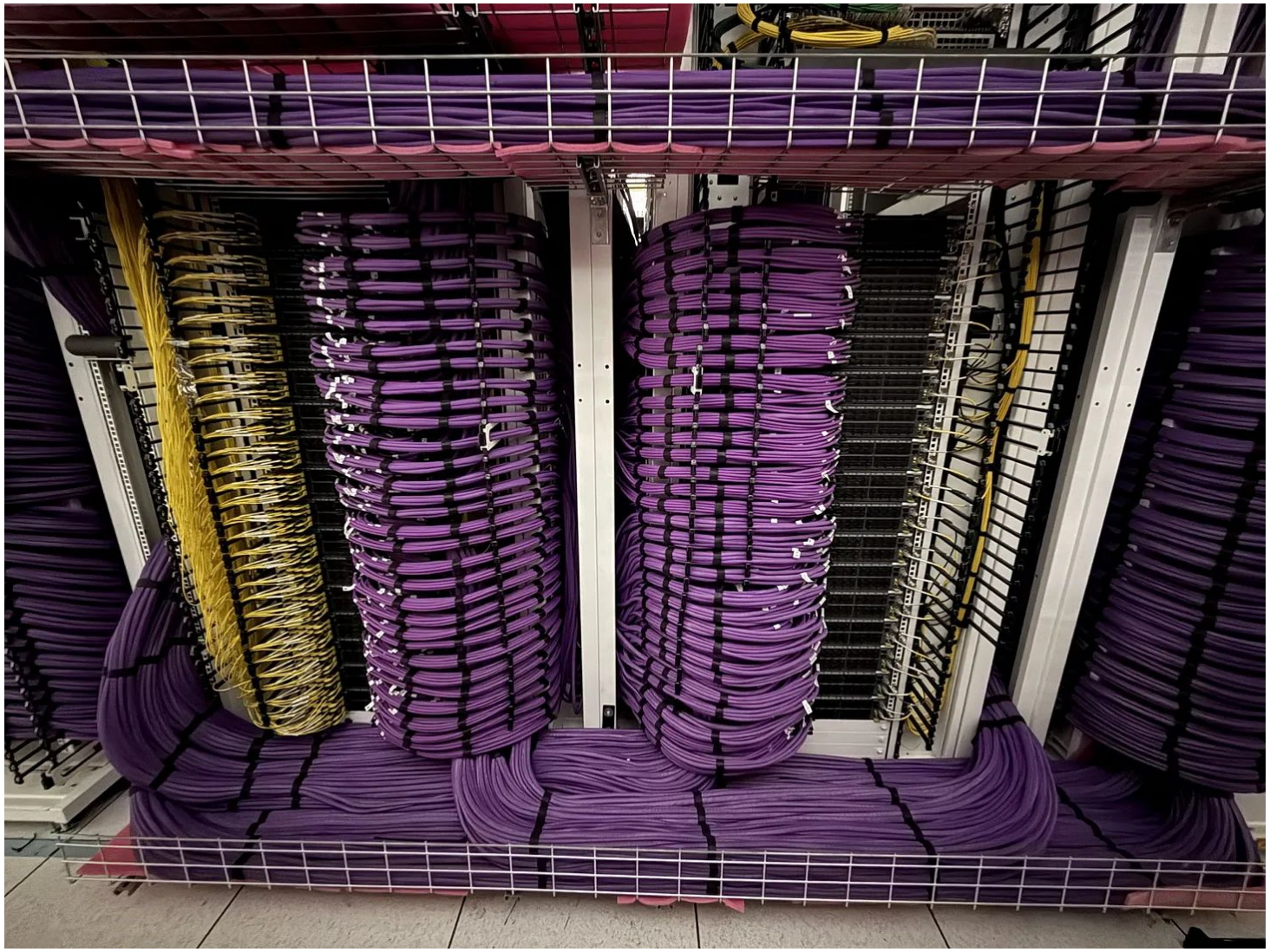
t.me/serv01

Распределенный блок

- **GPU clusters <——— you are here**
- DDP & AllReduce
- FSDP
- TP & CP & EP & PP …

Сегодняшняя лекция

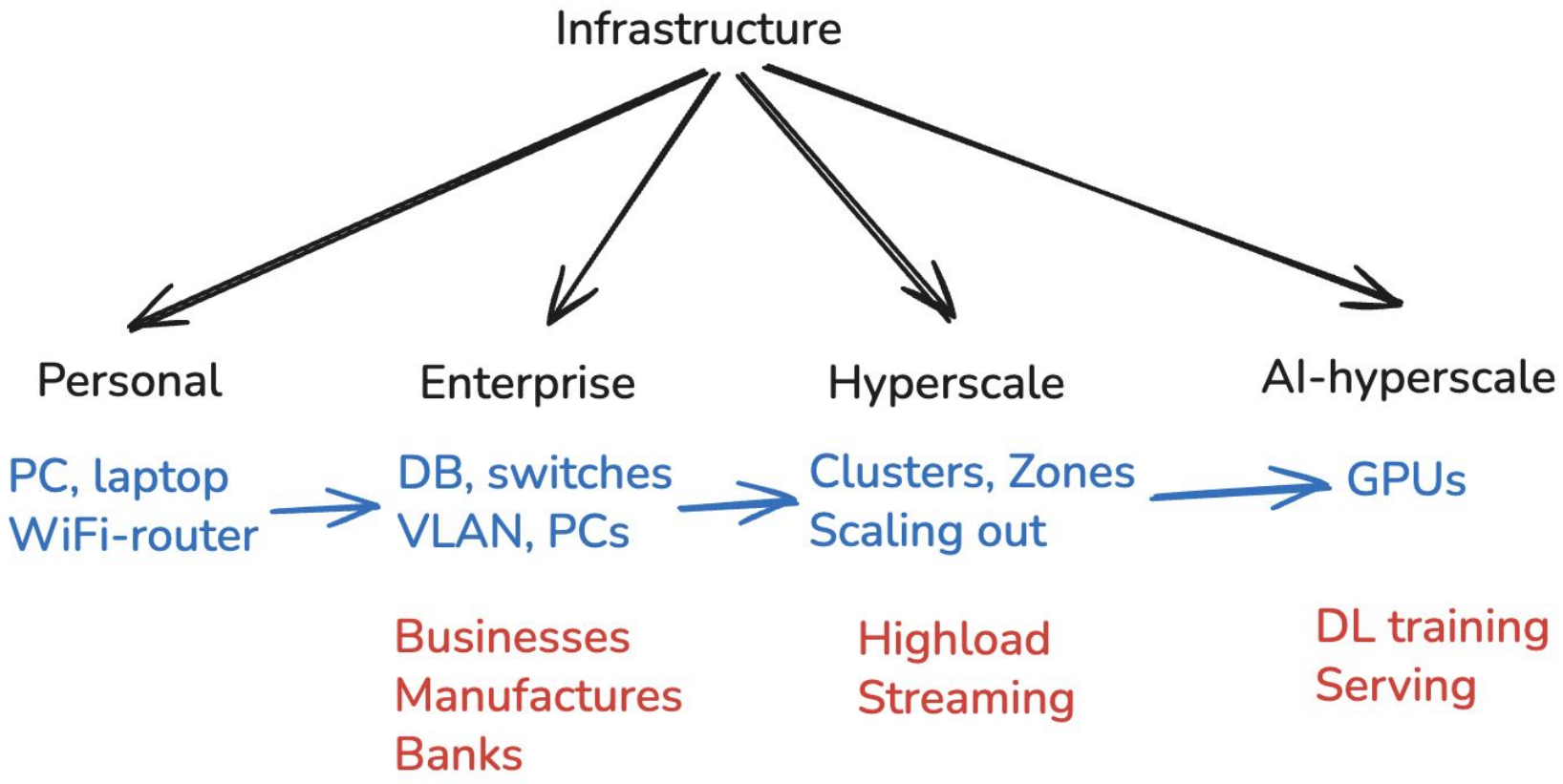
- Infrastructure: scaling, why
- GPU Clusters: compute, network fabrics, topology
- Software: K8s/slurm, MPI / NCCL
- Runtime: fault tolerant



A shot from Elon's Colossus 2

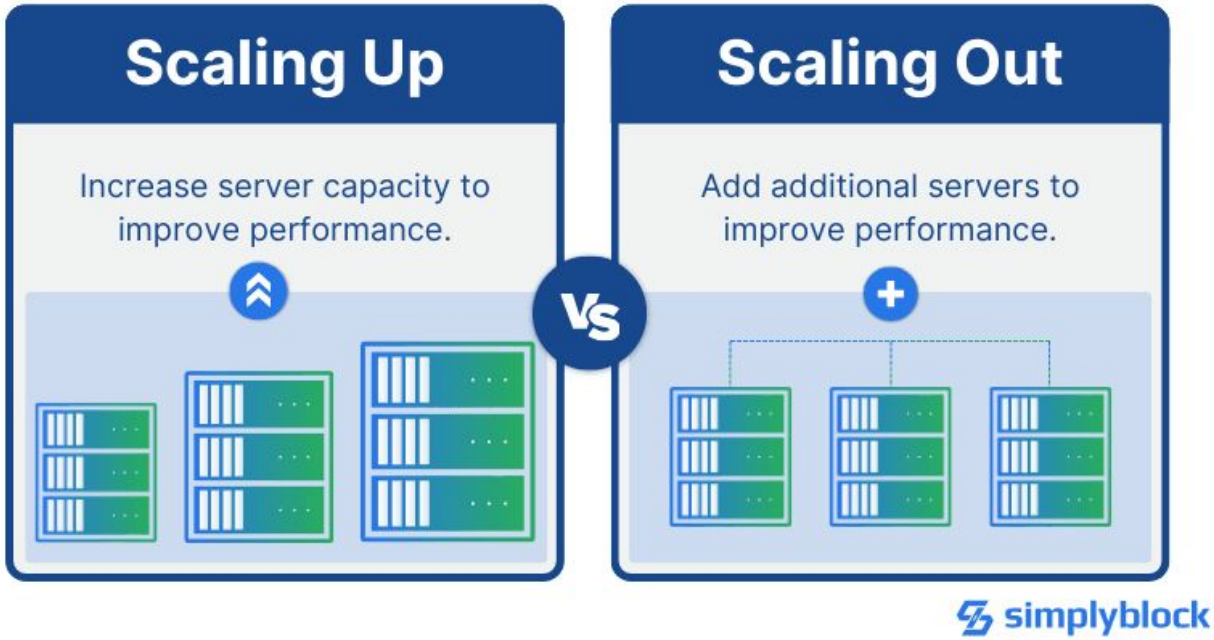
Infrastructure

Компьютерная инфраструктура



Масштабирование (scaling)

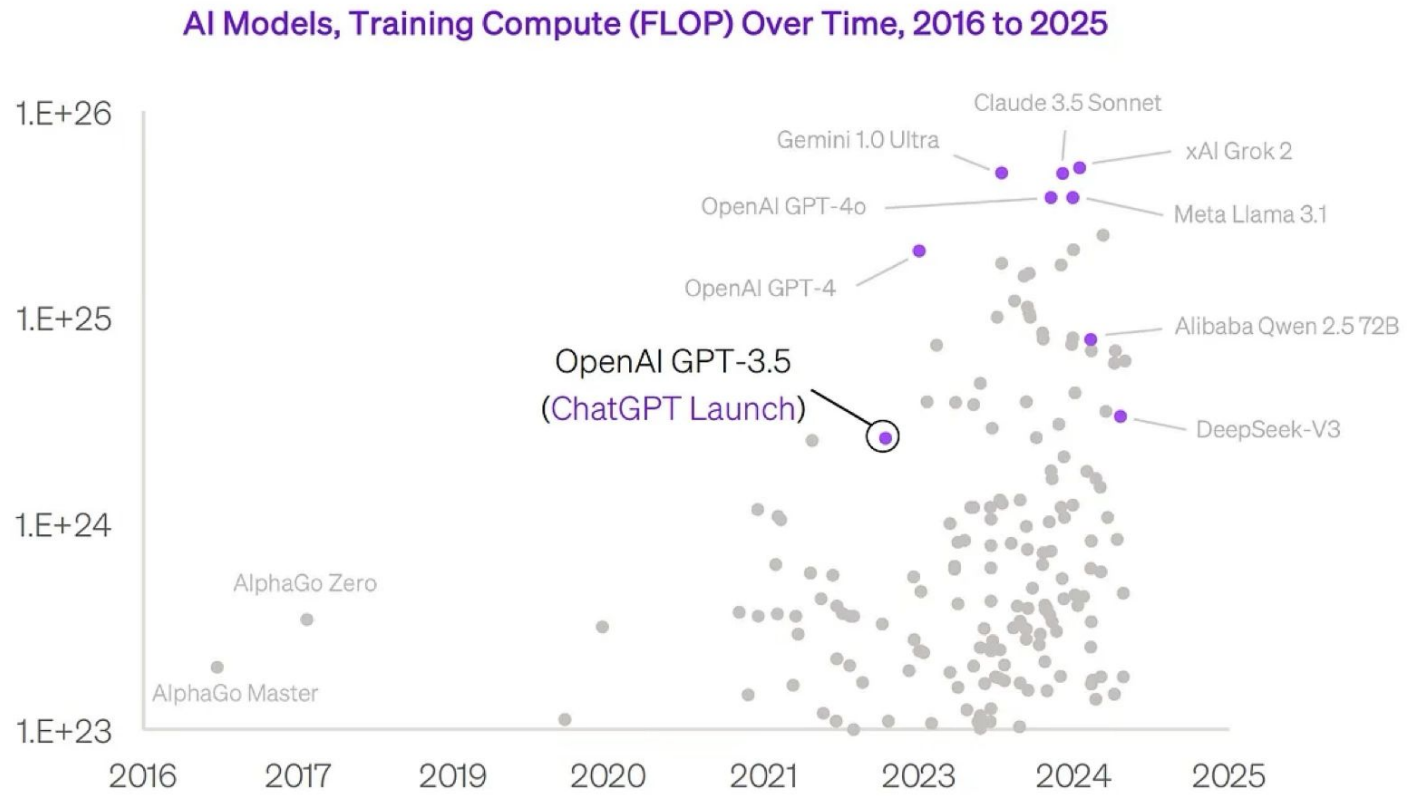
- Personal — vertical
- Enterprise — mostly vertical
- Hyperscale — mostly horizontal
- AI-hyperscale — mostly horizontal



Source: https://www.simplyblock.io/blog/scale-up-vs-scale-out/

Why scaling?

Over the course of two years, many companies have entered the race with high-quality language models, which puts each model at risk of commoditization.



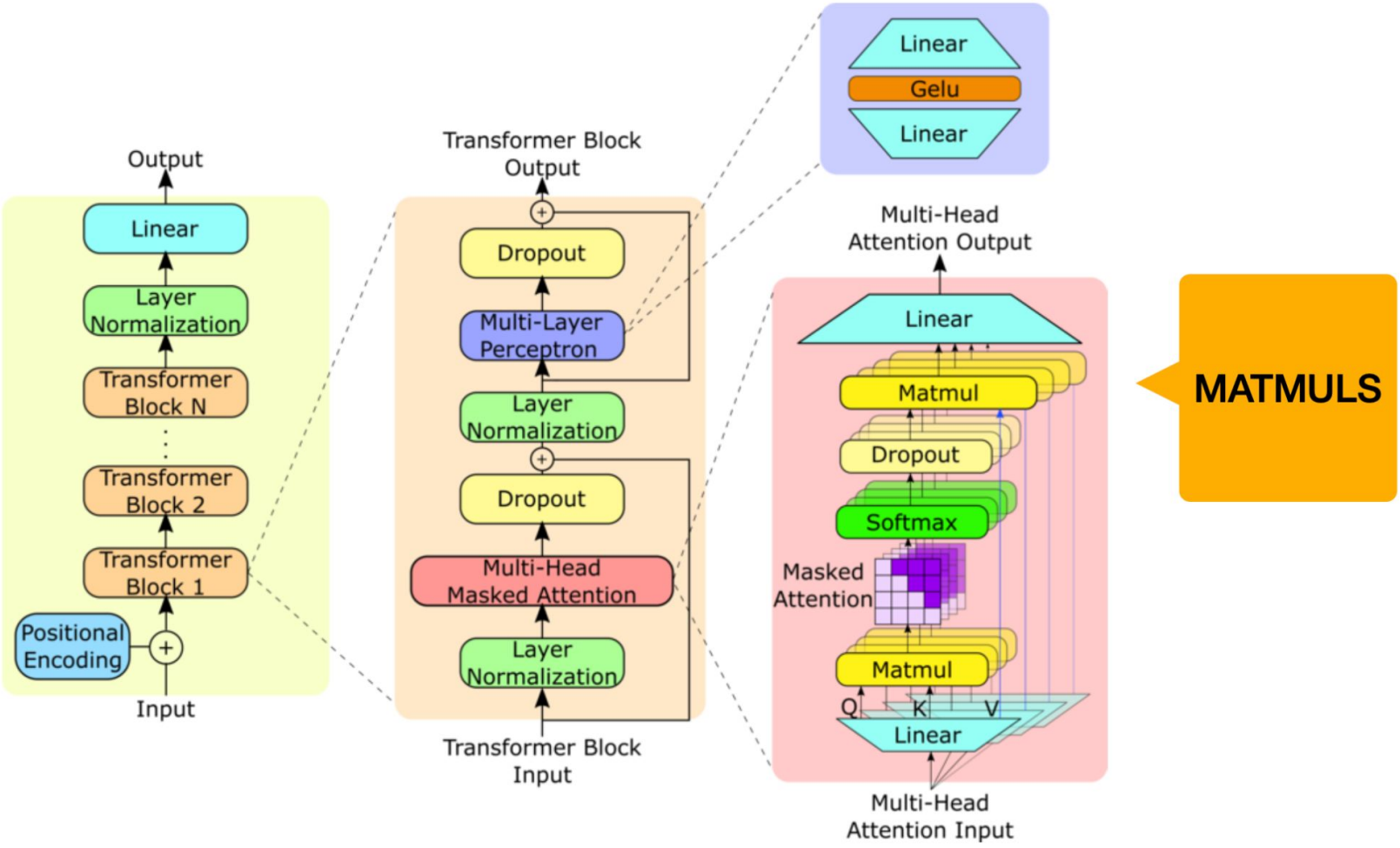
Source: https://chamath.substack.com/p/deep-dive-the-current-state-of-ai

FLOPs

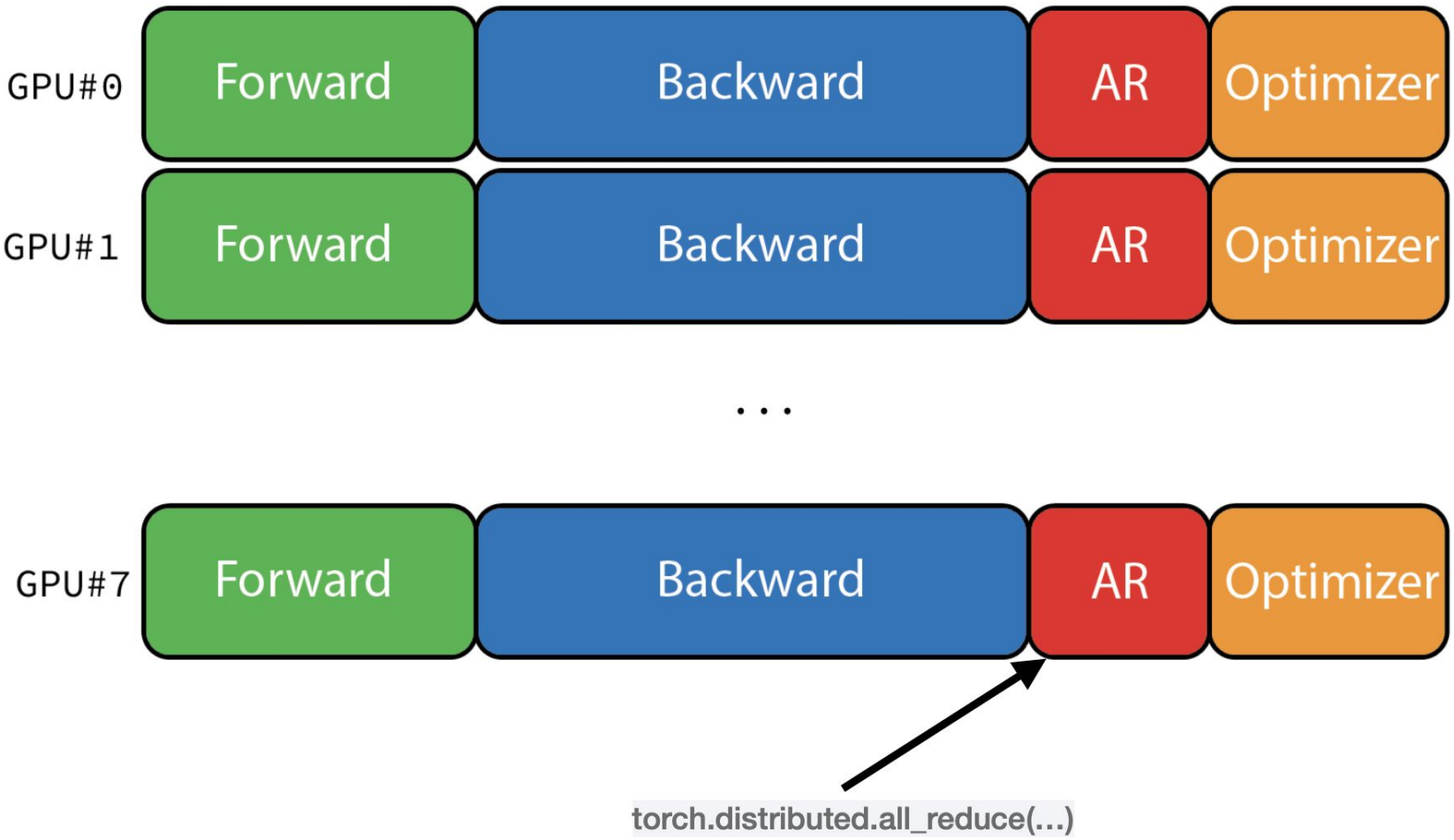
“If a single GPU can produce about **2 PetaFLOP/s** (2 * 10^15 floating point operations per second) and there are 86,400 seconds in a day, that equates to roughly 1.7 x 10^20 FLOPS. In the most ideal scenario, using **a single GPU**, you would need to train for approximately **16 years** to reach **10^24 FLOPs**.”

Source: https://www.chipstrat.com/p/gpu-networking-basics-part-1

Model training task



Model training task

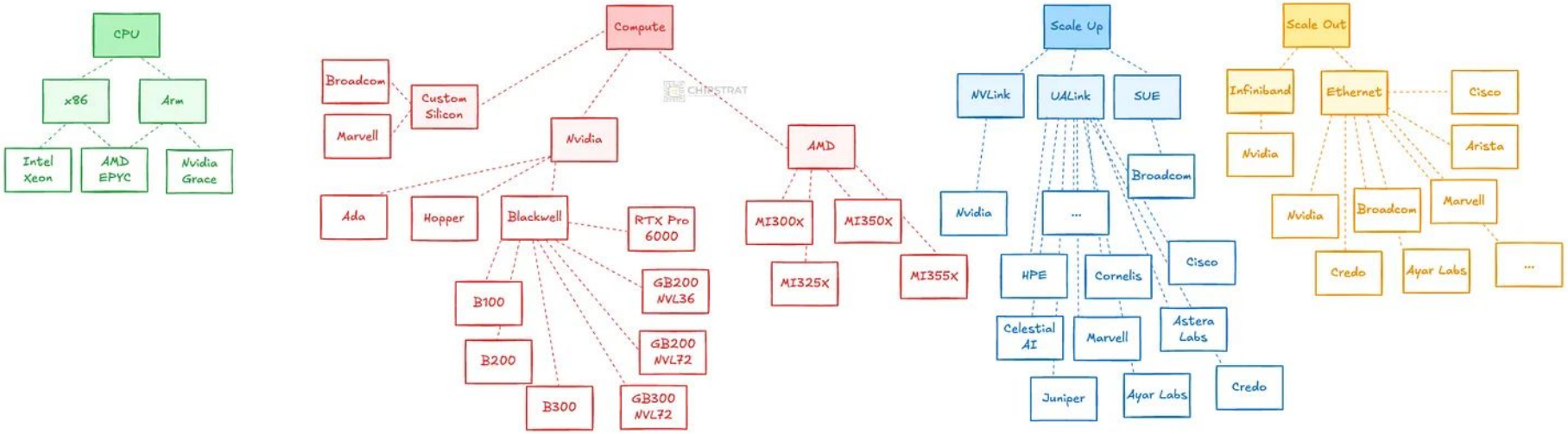


GPU Clusters

GPU Cluster resources

- Compute chips: GPU(NPU), CPU
- Network: NICs, switches, cords, chips
- Storage: RAM, SSD, HDD
- PDUs, cooling and much more…

GPU Cluster resources



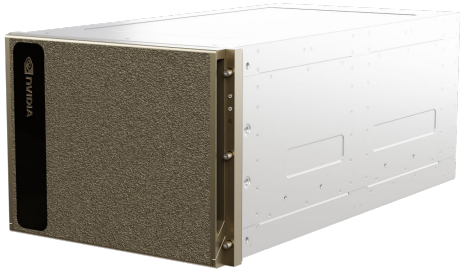
Source: https://www.chipstrat.com/p/right-sized-ai-infrastructure-marvell

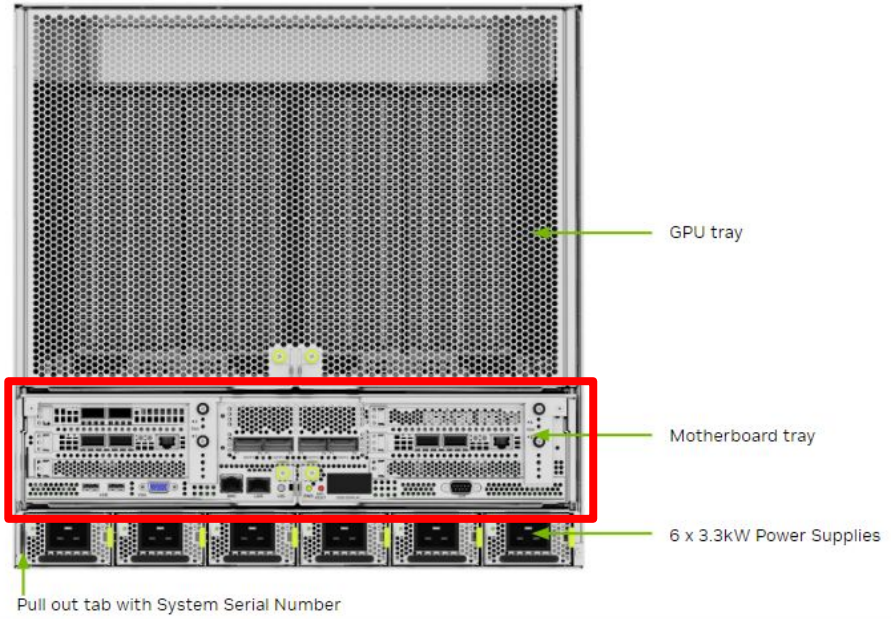
GPU Cluster

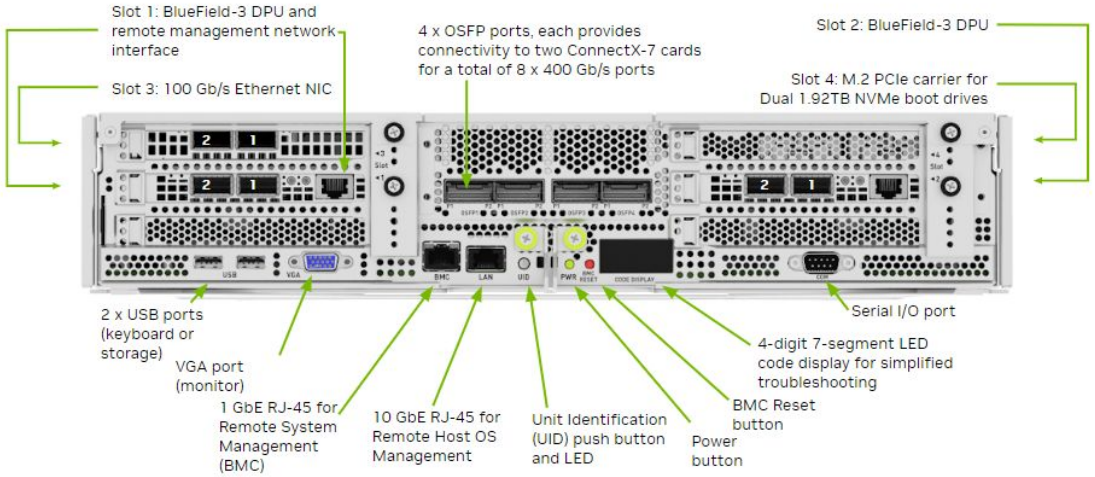
=

Source: https://docs.nvidia.com/dgx/dgxb200-user-guide/introduction-to-dgxb200.html

GPU Cluster

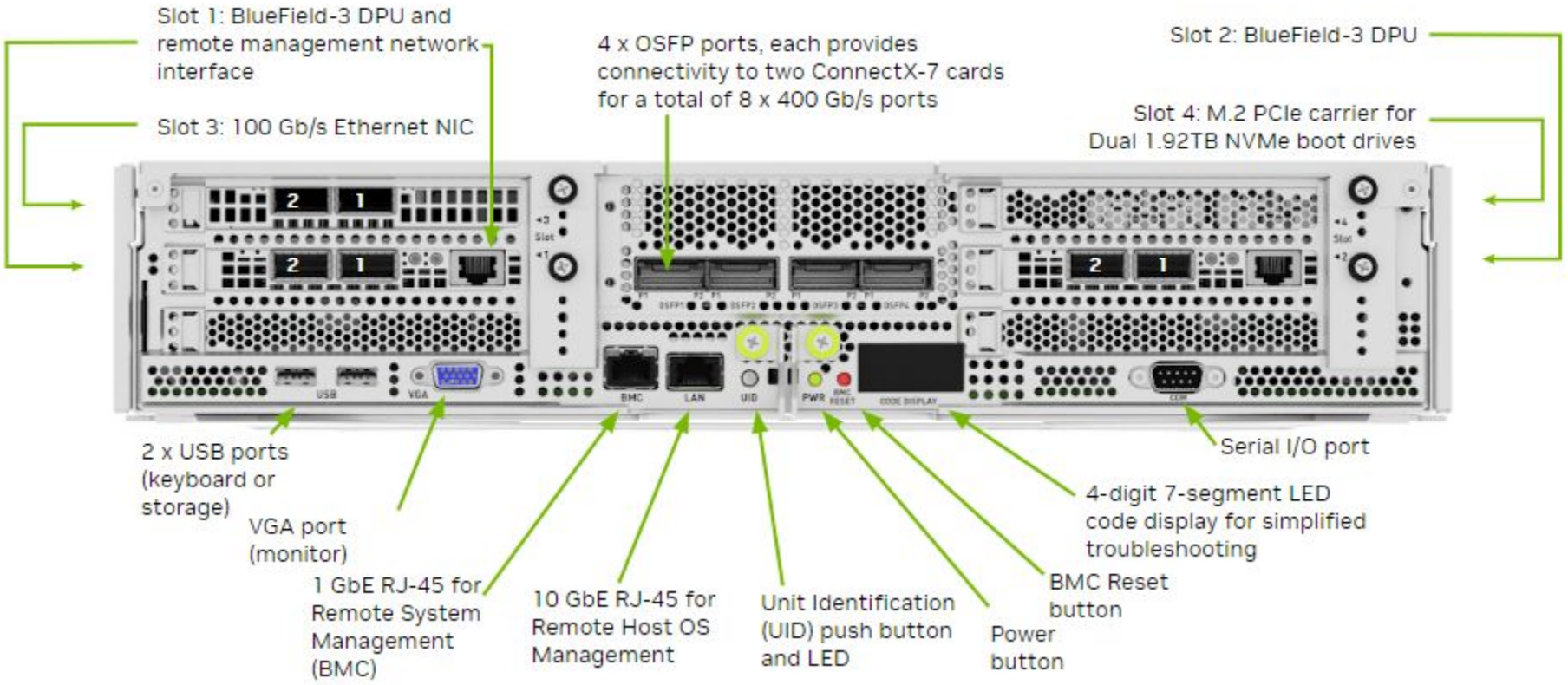






Source: https://docs.nvidia.com/dgx/dgxb200-user-guide/introduction-to-dgxb200.html

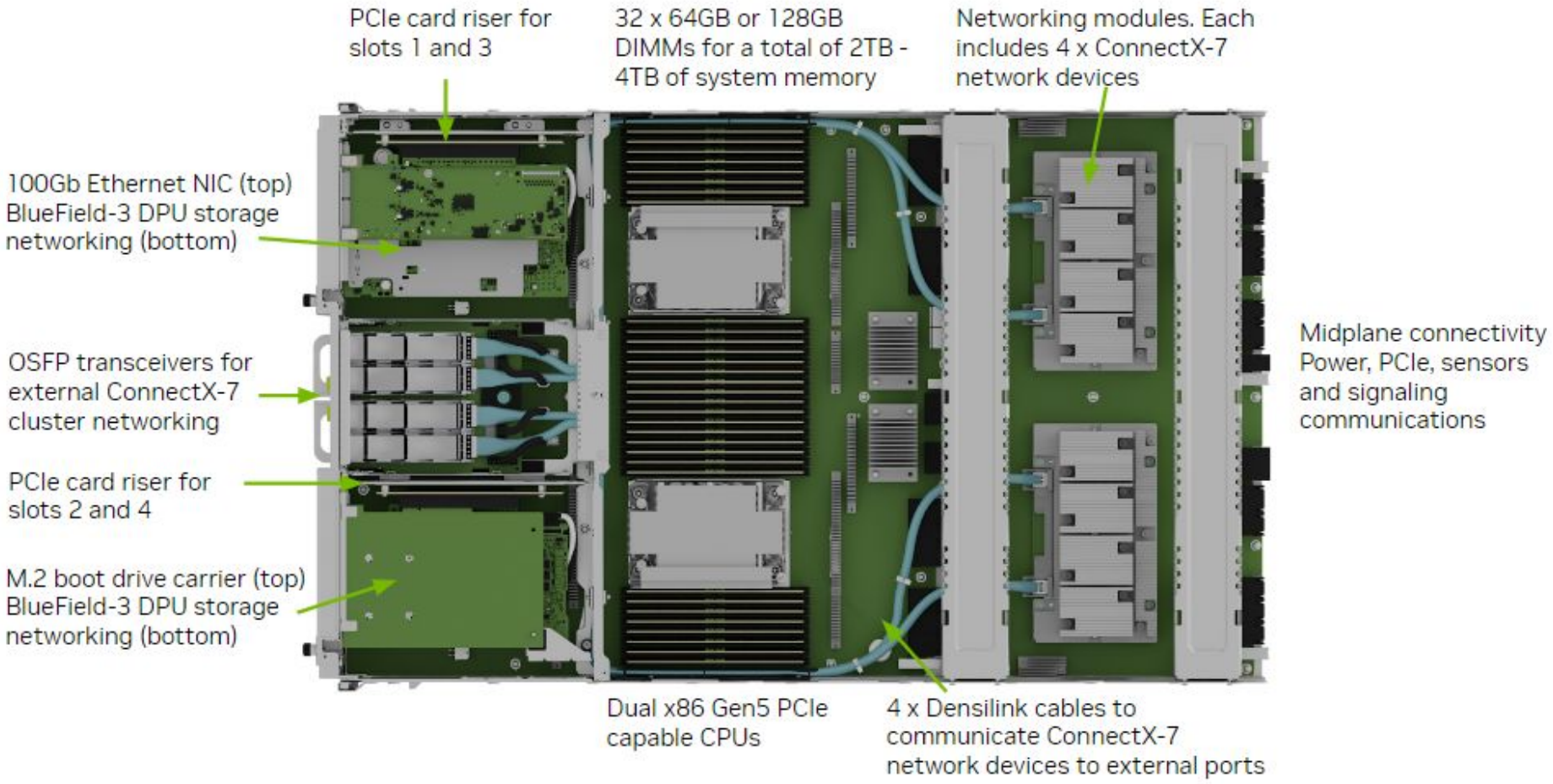
Motherboard tray



NIC = Network interface controller
OSFP = Octal Small Form-factor Pluggable
DPU = Data Processing Unit
BCM = Baseboard Management Controller

Source: https://docs.nvidia.com/dgx/dgxb200-user-guide/introduction-to-dgxb200.html

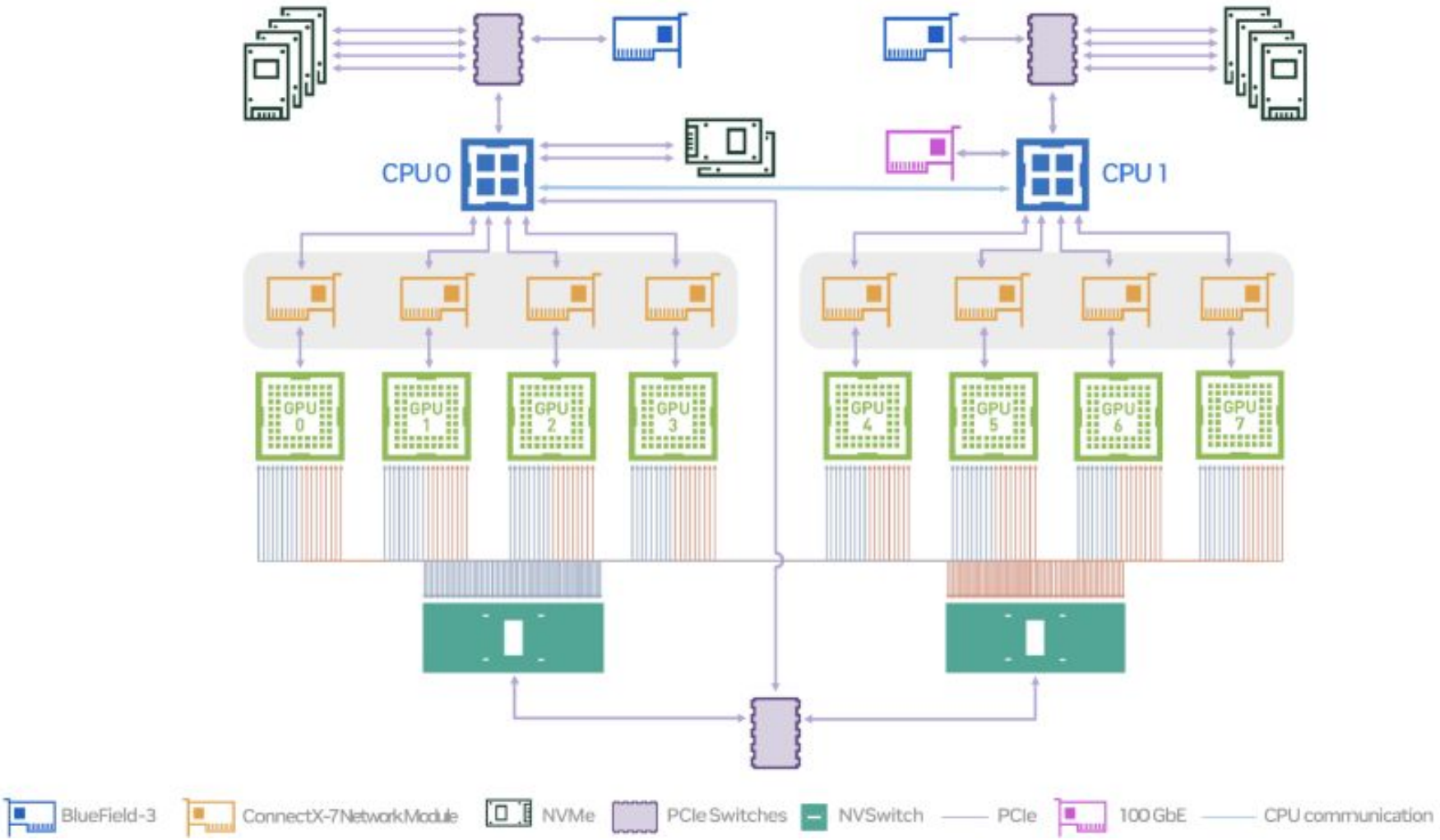
Motherboard tray



DIMM = Dual In-line Memory Module

Source: https://docs.nvidia.com/dgx/dgxb200-user-guide/introduction-to-dgxb200.html

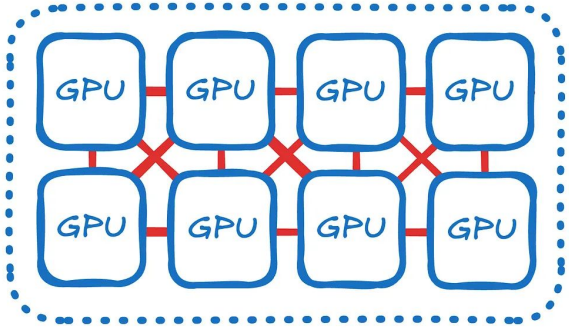
DGX B200 System Topology

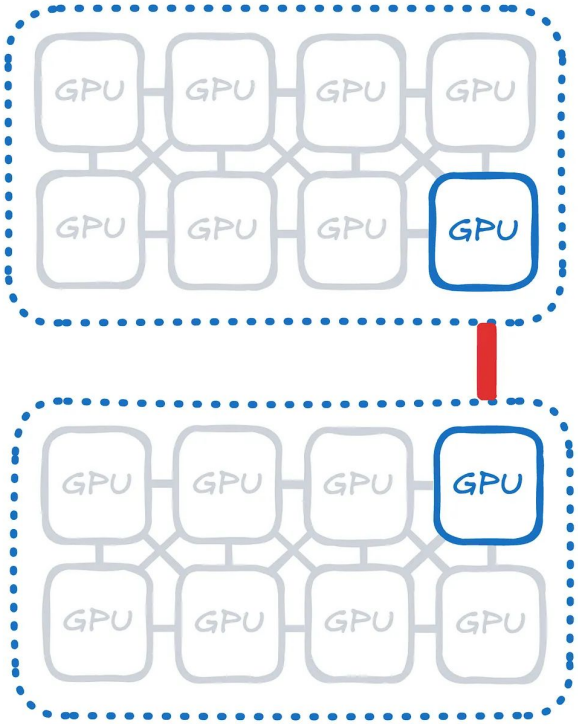


Source: https://docs.nvidia.com/dgx/dgxb200-user-guide/introduction-to-dgxb200.html

Intra vs Inter

Intra-node: Fast NVLink connects all GPUs in a node



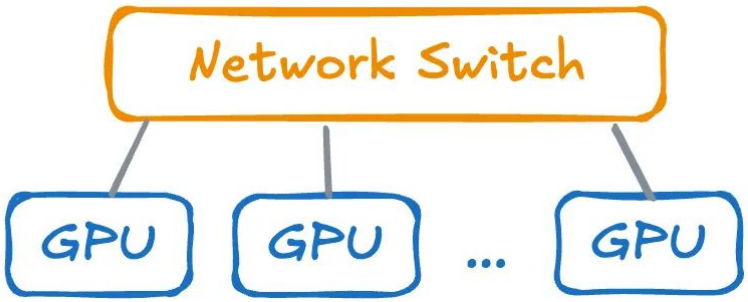


Slower InfiniBand connects GPUs in different nodes

Networking

Как мы можем соединить все GPU?

- Full Mesh (Каждый с каждым)
- Через свитчи



NVIDIA Q3400-RA Quantum-X800 XDR InfiniBand Switch

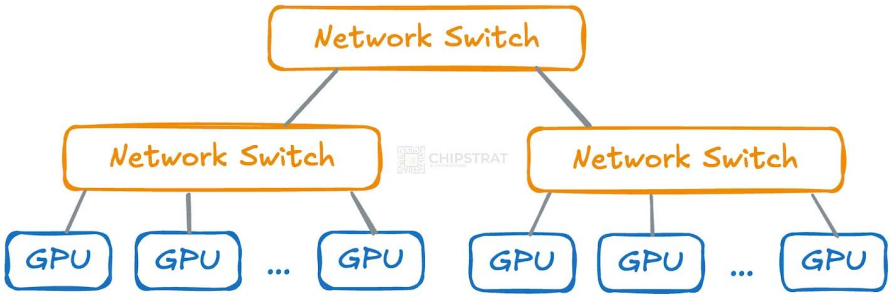
Leaf-Spine Topology

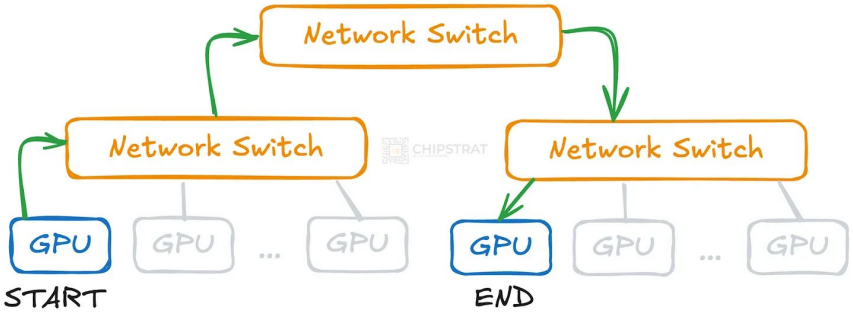
Плюсы:

- Экономим на проводах
- Просто скейлить

Минусы:

- Latency

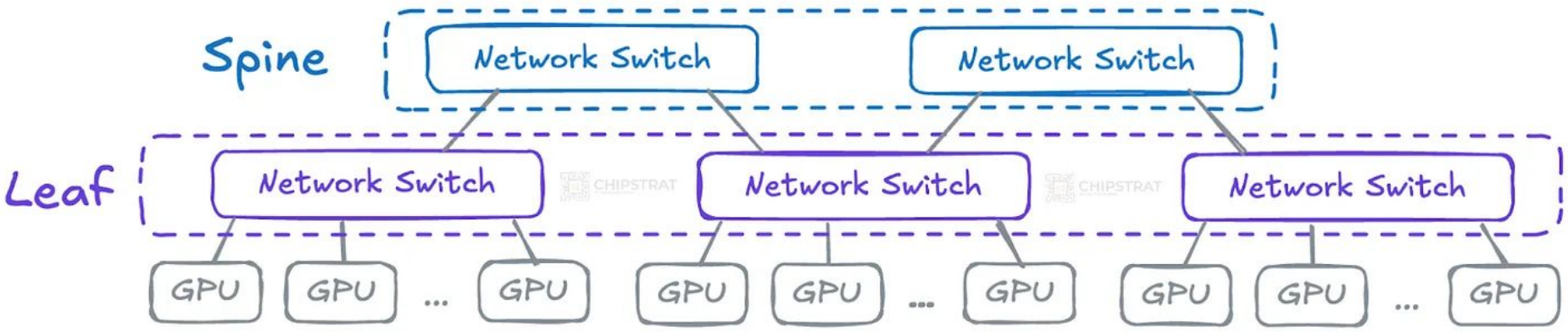




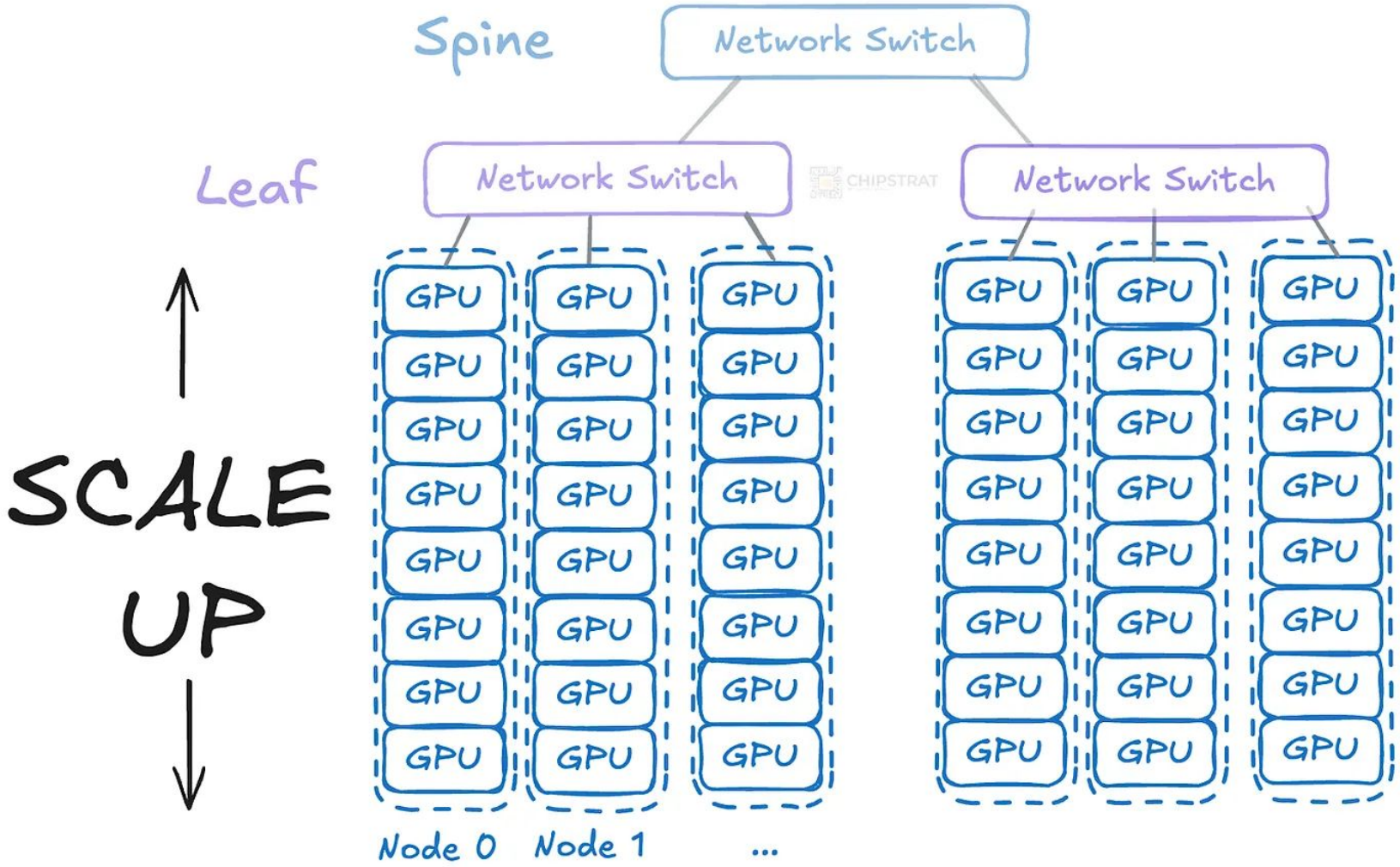
Network hops increase the time it takes for data to reach its destination.

Leaf-Spine Topology

- Leaf switches соединяем напрямую с compute
- Spine switches соединяем с leaf switches
- Просто масштабировать!



GPU Nodes


Spine
Network Switch
Leaf
Network Switch
Network Switch
SCALE UP
GPU
Node 0
Node 1
...

Intra

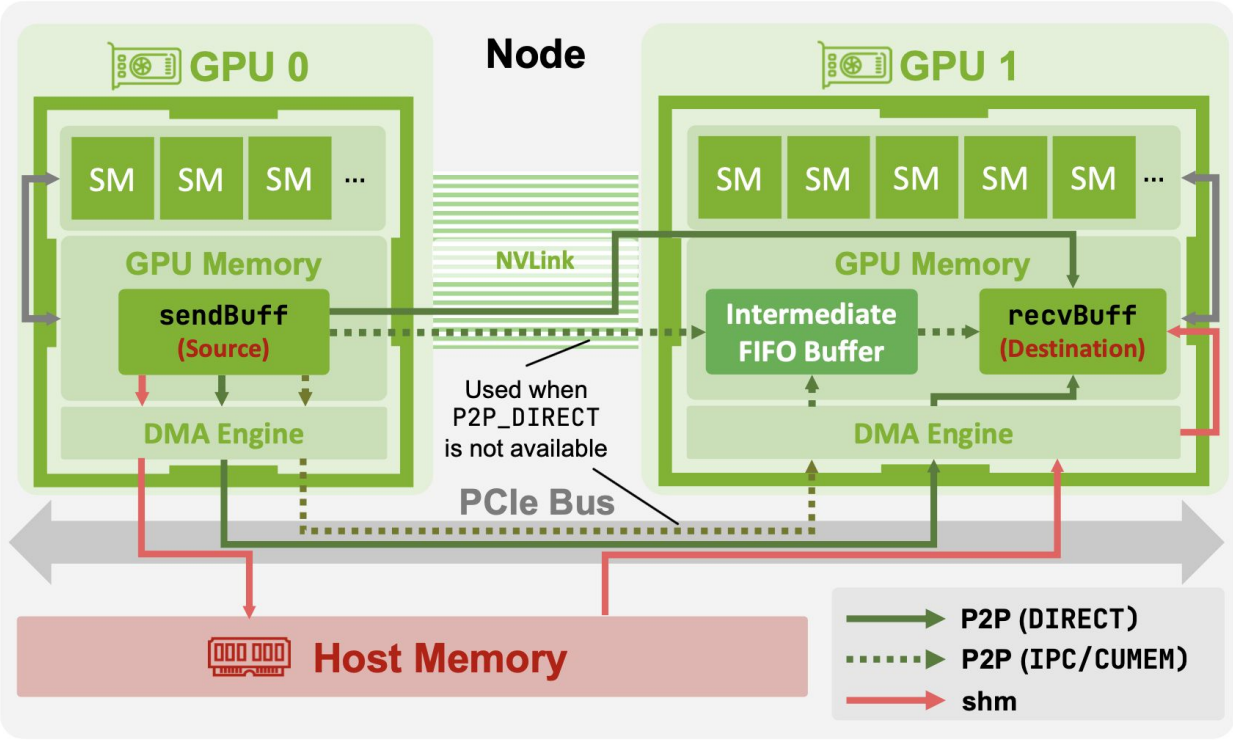


Fig. 1. Illustration of intra-node data transfer paths in NCCL. Each path is color-coded to indicate the selected transport and hardware support.

Source: https://arxiv.org/pdf/2507.04786

Inter

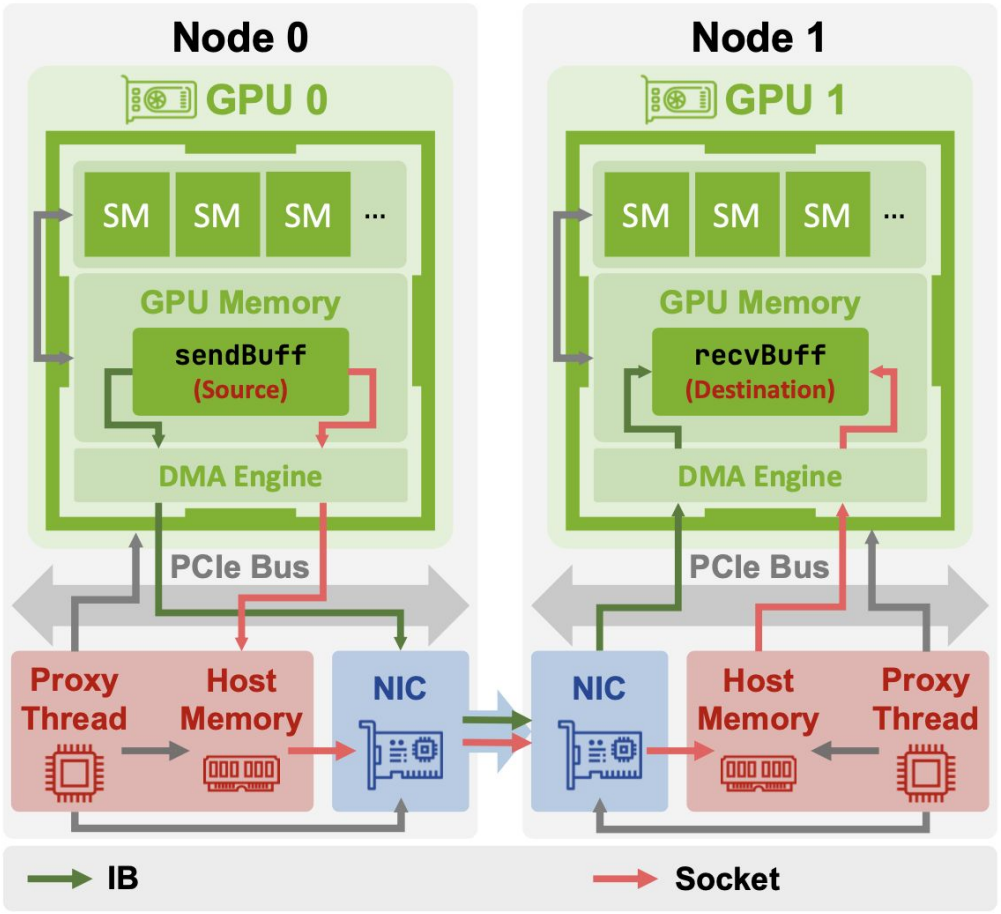


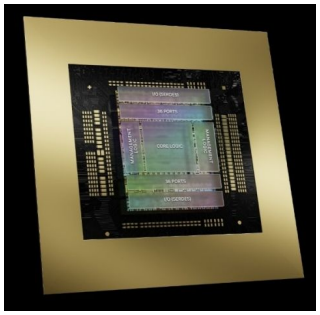
Fig. 2. Illustration of intra-node data transfer paths in NCCL. Each path is color-coded to indicate the selected transport and hardware support.

Source: https://arxiv.org/pdf/2507.04786

NVLink & Infiniband

NVLink

OSFP cord



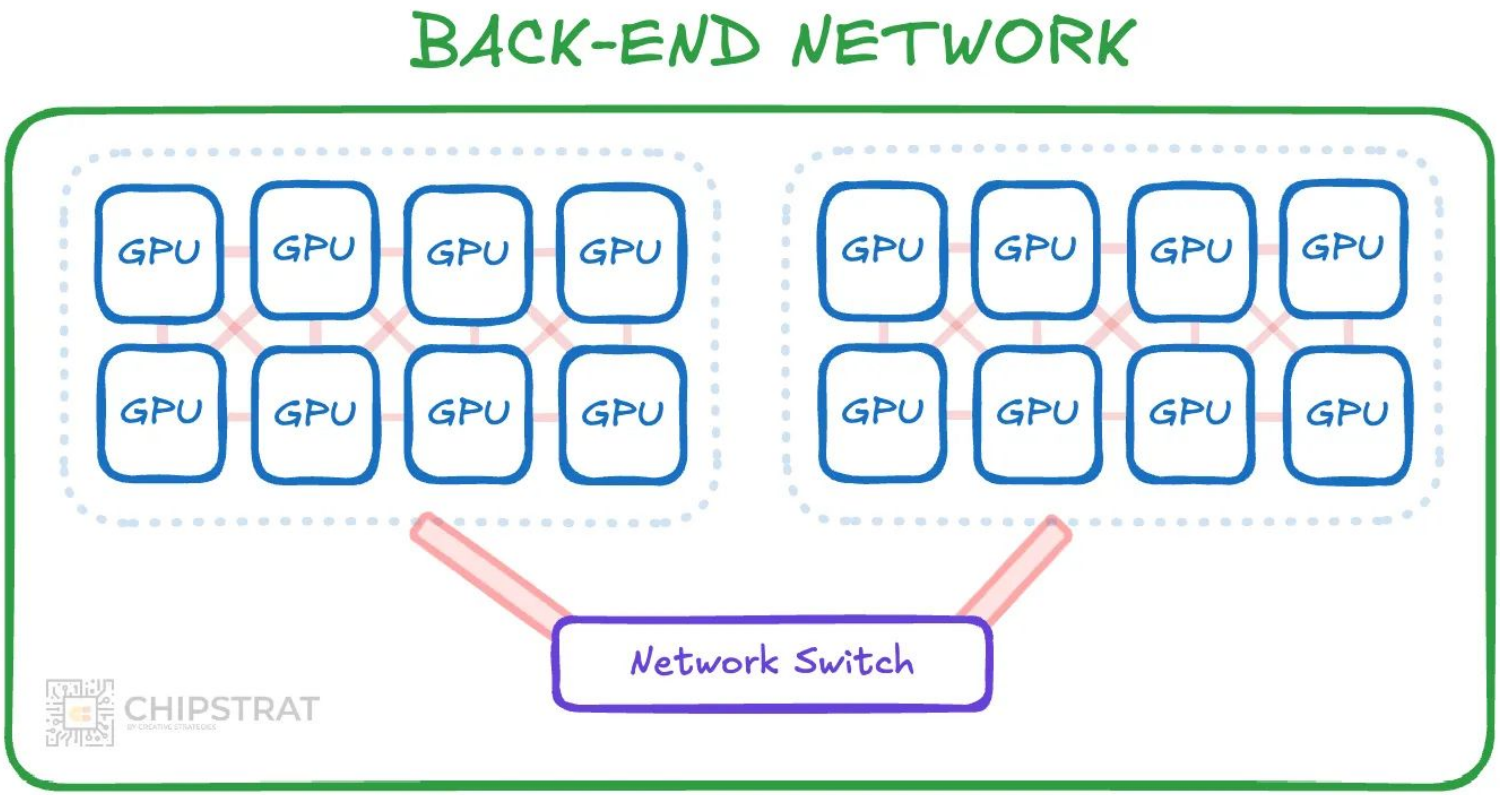
NVSwitch chip

NVIDIA Q3400-RA Quantum-X800 XDR InfiniBand Switch

Source: https://www.fibermall.com/blog/nvidia-nvlink-and-nvswitch-evolution.htm#

Backend Network

“GPU-to-GPU communication network is called the back-end network”

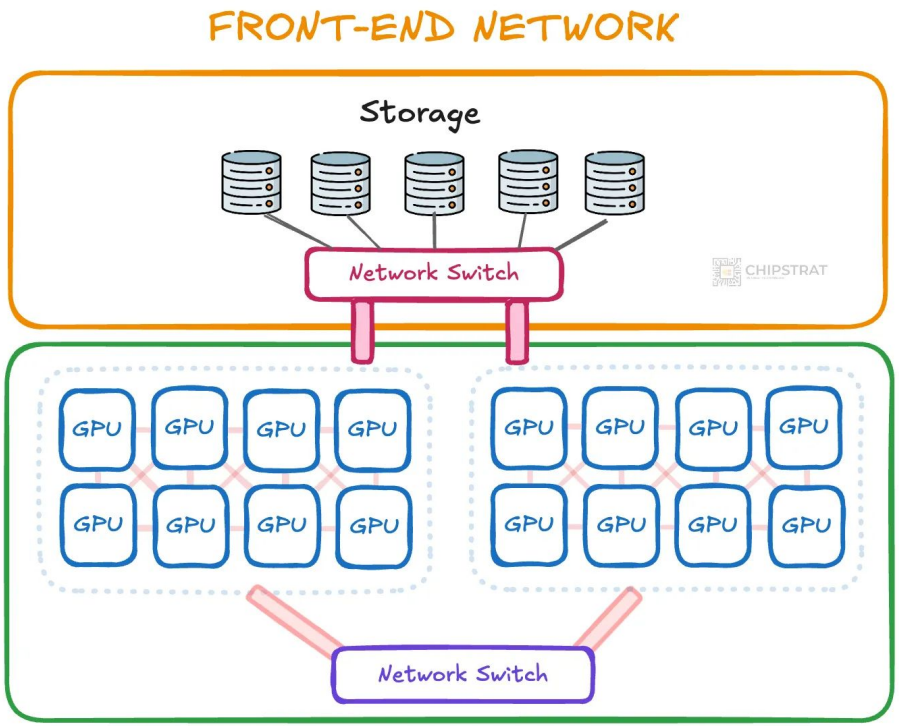


Source: https://www.chipstrat.com/p/gpu-networking-basics-part-2

Frontend Network

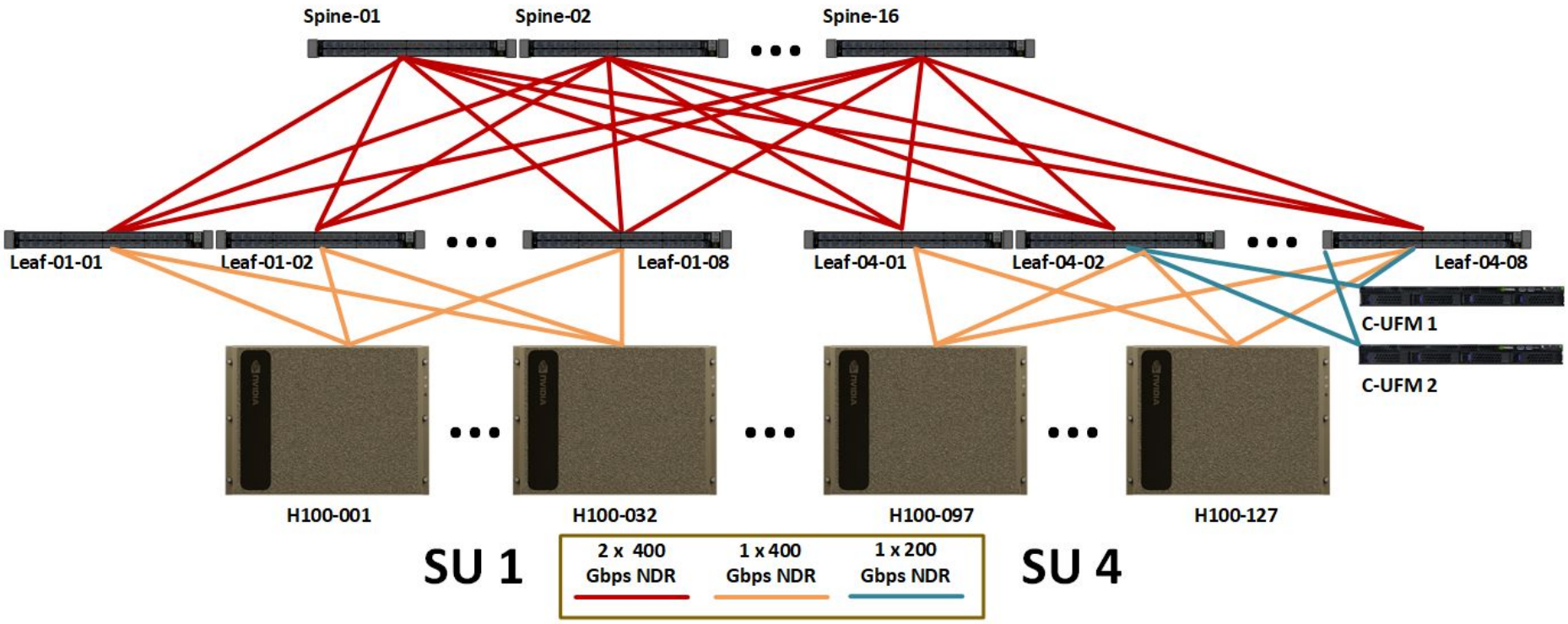
Все остальное это frontend network:

- Storage (датасеты)
- Удаленный доступ
- Management software
- Чекпоинты (*mb backend*)



Source: https://www.chipstrat.com/p/gpu-networking-basics-part-2

SuperPod Compute Network Fabric



Compute InfiniBand fabric for full 127 node DGX SuperPOD

Source: https://docs.nvidia.com/dgx-superpod/reference-architecture-scalable-infrastructure-h100/latest/network-fabrics.html

AI Hyperscale Clusters

Чтобы собрать кластер нужно учесть:

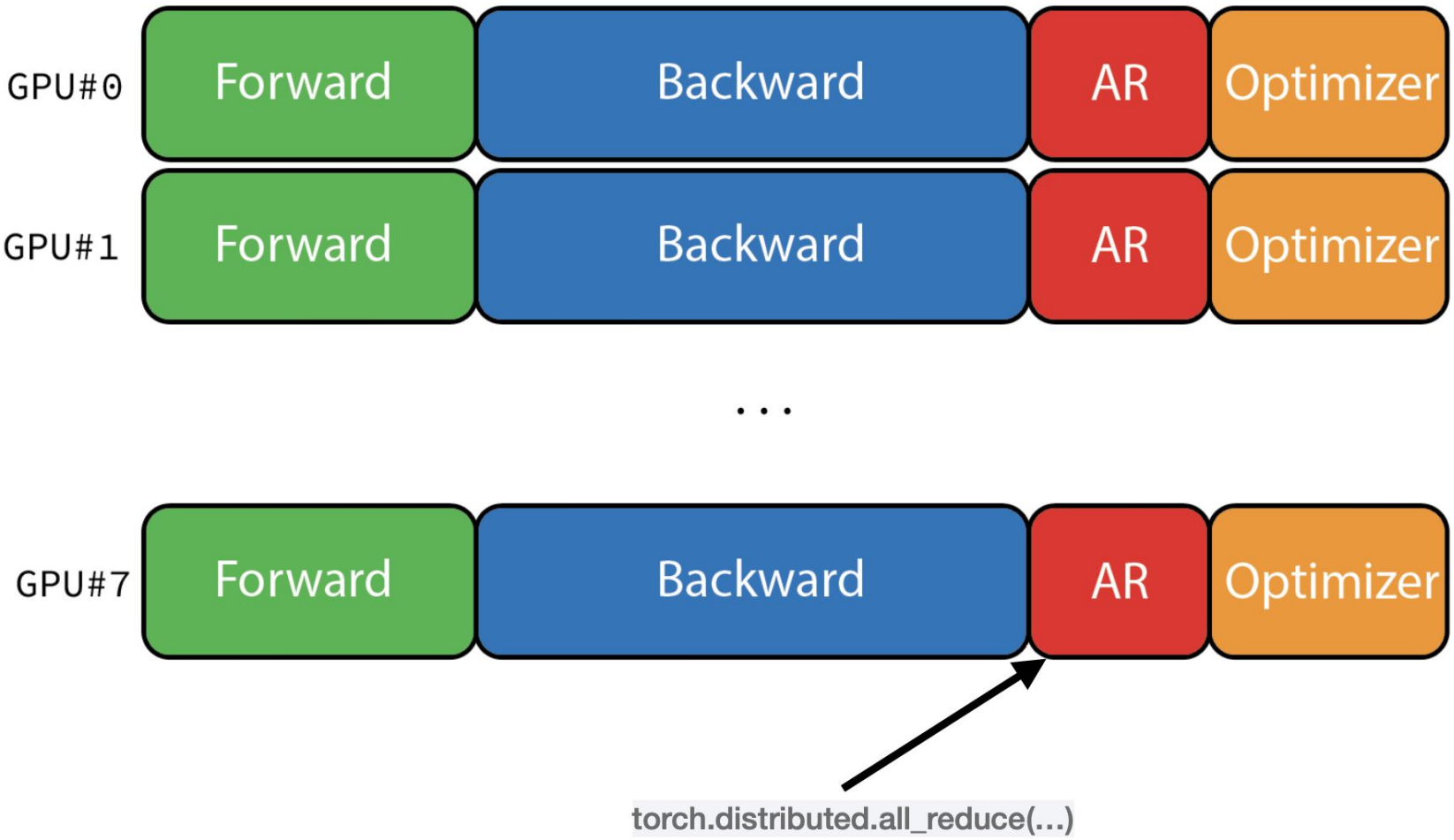
1. **GPU subsystem** — тип (SXM/PCIe), поколение (H100/B200), NVLink/NVSwitch топология, GPU-to-GPU пропускная способность.
2. **CPU platform** — количество сокетов, ядра, PCIe-линии, NUMA-архитектура, совместимость с GPU-фабрикой.
3. **Memory subsystem** — объём и скорость RAM, каналы на сокет, баланс CPU↔GPU, поддержка CXL/HBM-кеша.
4. **Interconnect / network fabric** — тип (InfiniBand, RoCEv2), скорость (400/800 Gb/s), топология (fat-tree, dragonfly), oversubscription.
5. **Storage hardware** — NVMe SSD tiers, HDD-пулы, JBOD/expander-нод, NVMe-oF, RAID/Erasure-coding схемы.
6. **Power infrastructure** — трёхфазное питание, PDU типы (C19/C21), N+1/N+N резервы, UPS/DC bus, power budgeting per rack.
7. **Cooling system** — воздушное vs жидкостное (CDU, rear-door HX), потоки холодного/горячего воздуха, температурные сенсоры.
8. **Rack & chassis design** — форм-фактор (U-высота, глубина), плотность GPU/CPU/дисков, кабель-менеджмент, монтаж оптики.
9. **Cabling & optics** — OSFP/QSFP112, DAC/AOC/SMF, длины, радиусы изгиба, маркировка и трассировка.
10. **Out-of-band management** — BMC/IPMI/Redfish, консольный доступ, сеть управления, питание и перезапуск удалённо.

Source: ChatGPT

Software

А какую задачу мы вообще решаем?

Model training task



Минимальный стек

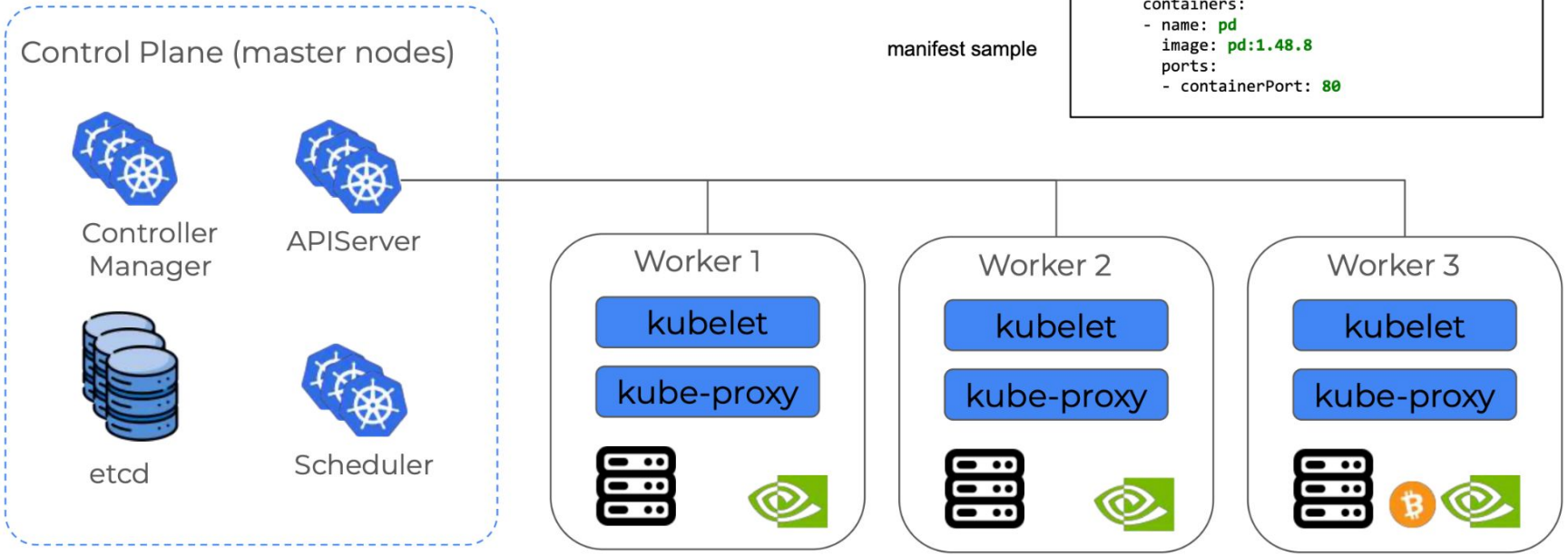
- OS (Ubuntu) + drivers + fabric managers
- SLURM / Kuber
- CUDA + NCCL + UCX
- MPI, torchrun
- torch.distributed
- PyTorch, FSDP, Apex/xFormers, Flash Attention
- VAST / WEKA
- Ceph
- DCGM + Prometheus + Grafana
- Nsight Systems / PyTorch Profiler
- NCCL tests, nvidia-smi topo

Kubernetes

The K8s

manifest sample

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: private-detector
  labels:
    app: pd
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: pd
    spec:
      containers:
      - name: pd
        image: pd:1.48.8
        ports:
        - containerPort: 80
```



Source: https://github.com/mryab/efficient-dl-systems/blob/2024/week10_invited/lecture.pdf

Why K8s

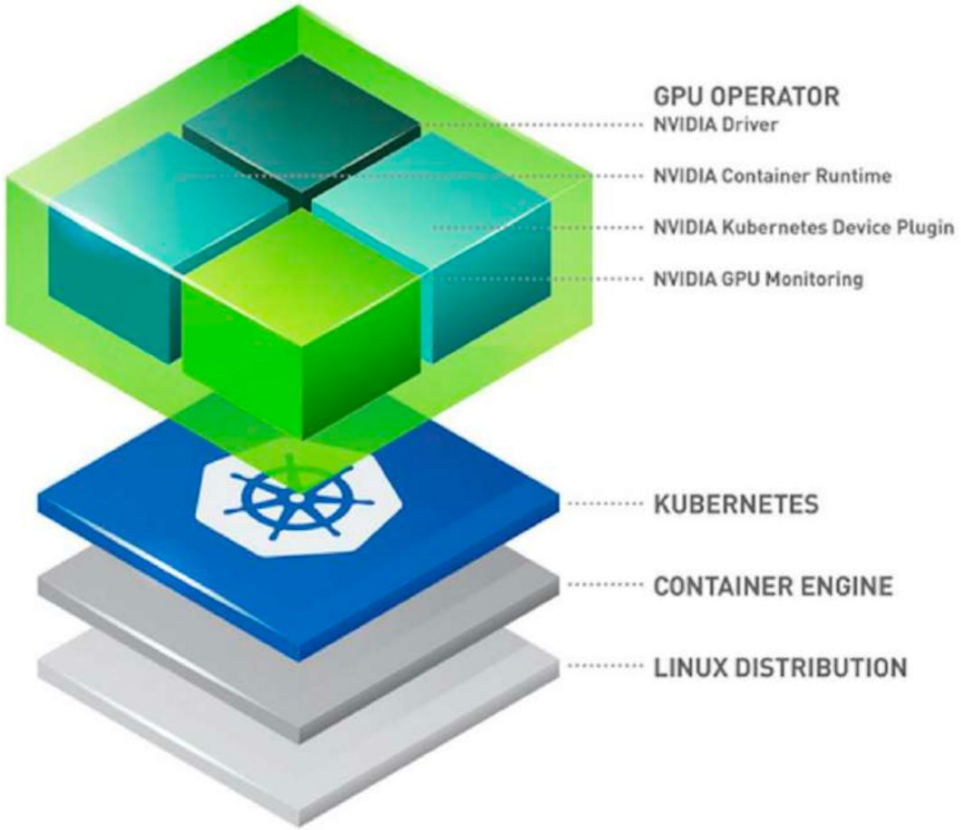
- Scalability
- Fault Tolerance and High Availability
- Declarative Configuration and Automation
- Portability
- Ecosystem and Community
- Containers
- Rolling upgrades and rollbacks
- Service discovery and load balancing

Source: https://github.com/mryab/efficient-dl-systems/blob/2024/week10_invited/lecture.pdf

NVIDIA GPU Operator

Components:

- GPU Feature discovery
- Nvidia Container Runtime
- K8s Device Plugin
- DCGM Exporter
- Driver Manager
- MIG Manager

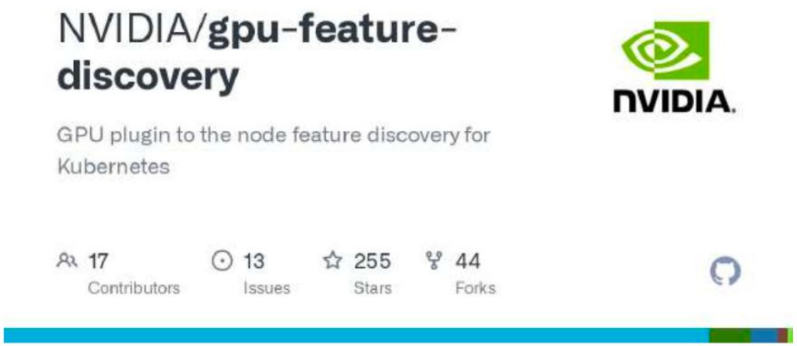


Source: https://github.com/mryab/efficient-dl-systems/blob/2024/week10_invited/lecture.pdf

NVIDIA GPU Operator

Components:

- **GPU Feature discovery**
- Nvidia Container Runtime
- K8s Device Plugin
- DCGM Exporter
- Driver Manager
- MIG Manager



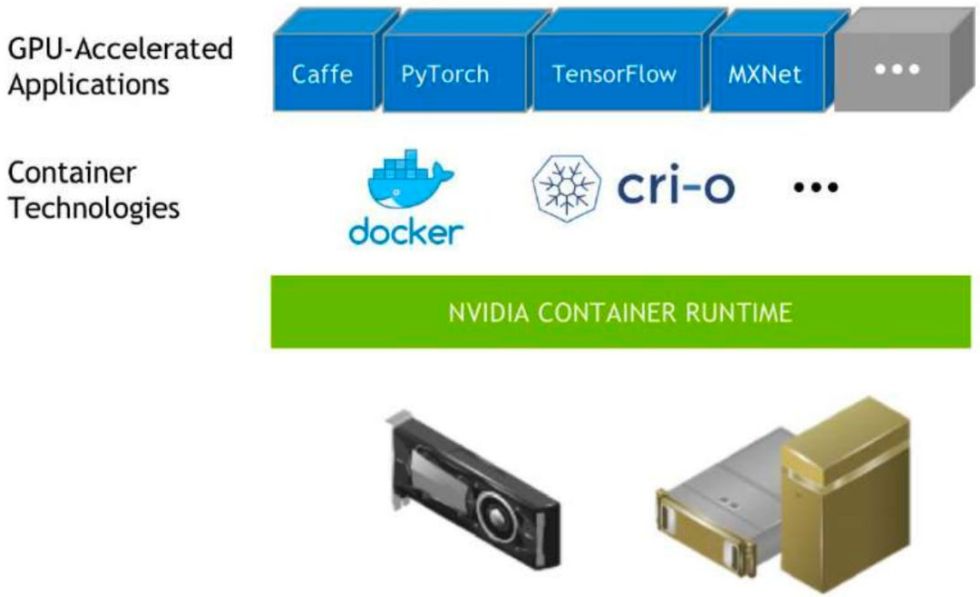

```
$ kubectl get nodes --show-labels

NAME       STATUS   ROLES        AGE    VERSION    LABELS
ds-node1   Ready    worker       210d   v1.24.3    ...,nvidia.com/gpu.product=NVIDIA-A100,nvidia.com/gpu.replicas=2
ds-node2   Ready    worker       210d   v1.24.3    ...,nvidia.com/gpu.product=Tesla-T4,nvidia.com/gpu.replicas=4
ds-node3   Ready    worker       210d   v1.24.3    ...,nvidia.com/gpu.product=Tesla-T4,nvidia.com/gpu.replicas=4
```

Source: https://github.com/mryab/efficient-dl-systems/blob/2024/week10_invited/lecture.pdf

NVIDIA GPU Operator

Components:

- GPU Feature discovery
- **Nvidia Container Runtime**
- K8s Device Plugin
- DCGM Exporter
- Driver Manager
- MIG Manager

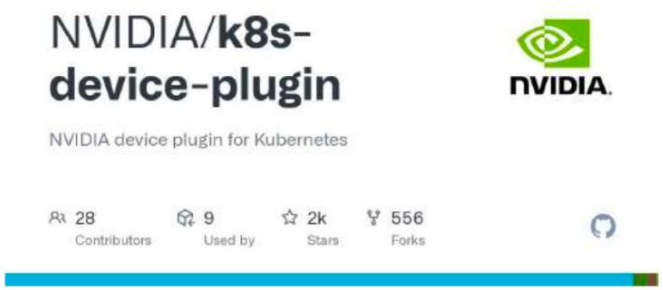


Source: https://github.com/mryab/efficient-dl-systems/blob/2024/week10_invited/lecture.pdf

NVIDIA GPU Operator

Components:

- GPU Feature discovery
- Nvidia Container Runtime
- **K8s Device Plugin**
- DCGM Exporter
- Driver Manager
- MIG Manager




```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: private-detector
  labels:
    app: pd
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: pd
    spec:
      containers:
      - name: pd
        image: pd:1.48.8
        ports:
        - containerPort: 80
         resources:
          limits:
            cpu: 2
            memory: 16Gi
            nvidia.com/gpu: 2
```

Source: https://github.com/mryab/efficient-dl-systems/blob/2024/week10_invited/lecture.pdf

NVIDIA GPU Operator

Components:

- GPU Feature discovery
- Nvidia Container Runtime
- K8s Device Plugin
- **DCGM Exporter**
- Driver Manager
- MIG Manager

DCGM = Data Center GPU Manager

Source: https://github.com/mryab/efficient-dl-systems/blob/2024/week10_invited/lecture.pdf

NVIDIA GPU Operator

Components:

- GPU Feature discovery
- Nvidia Container Runtime
- K8s Device Plugin
- DCGM Exporter
- **Driver Manager**
- MIG Manager

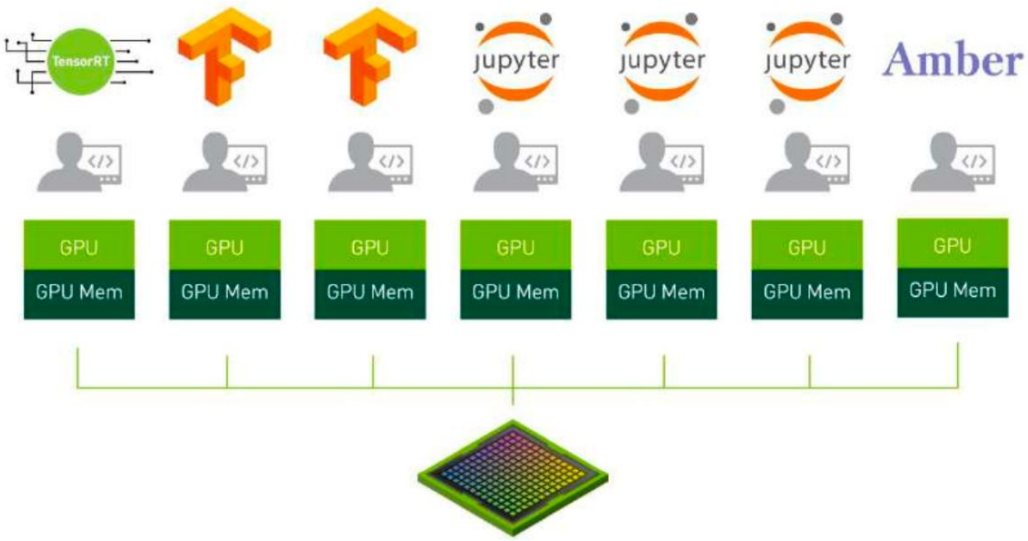
```
Fri Dec 30 10:57:52 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 525.60       Driver Version: 525.60       CUDA Version: 12.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  NVIDIA A100-PCI...  Off  | 00000000:86:00.0 Off |                    0 |
| N/A   30C    P0    36W / 250W |      0MiB / 40960MiB |      0%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Source: https://github.com/mryab/efficient-dl-systems/blob/2024/week10_invited/lecture.pdf

NVIDIA GPU Operator

Components:

- GPU Feature discovery
- Nvidia Container Runtime
- K8s Device Plugin
- DCGM Exporter
- Driver Manager
- **MIG Manager**



MIG = Multi-Instance GPU

Source: https://github.com/mryab/efficient-dl-systems/blob/2024/week10_invited/lecture.pdf

NVIDIA GPU Operator

Components:

- GPU Feature discovery
- Nvidia Container Runtime
- K8s Device Plugin
- DCGM Exporter
- Driver Manager
- **MIG Manager**

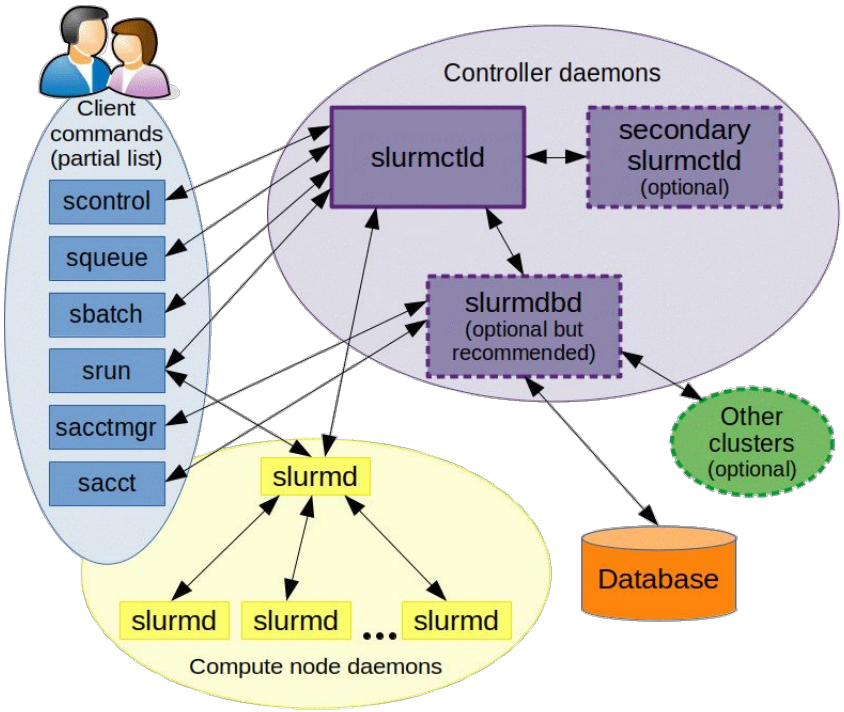
```
kubectl label nodes ds-node1 nvidia.com/mig.config=all-1g.10gb

kubectl label nodes ds-node1 nvidia.com/mig.config=all-1g.5gb

kubectl label nodes ds-node1 nvidia.com/mig.config=all-3g.40gb
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: private-detector
  labels:
    app: pd
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: pd
    spec:
      containers:
      - name: pd
        image: pd:1.76.9
        ports:
        - containerPort: 80
         resources:
          limits:
            cpu: 1
            memory: 2Gi
            nvidia.com/mig-1g.5gb: 1
```

Source: https://github.com/mryab/efficient-dl-systems/blob/2024/week10_invited/lecture.pdf

Slurm

- Планировщик задач (job scheduler) и система управления ресурсами
- Проще операционно; пользователи работают через sbatch/srun
- По сути “терминал на стероидах”

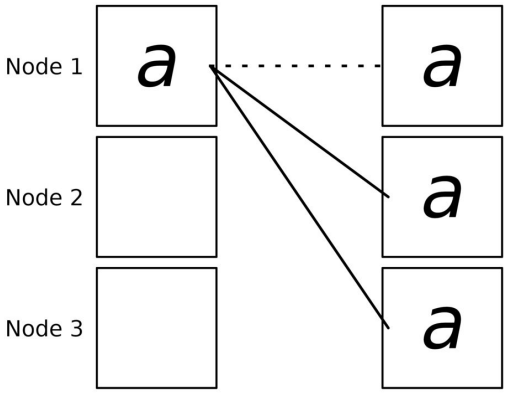


MPI

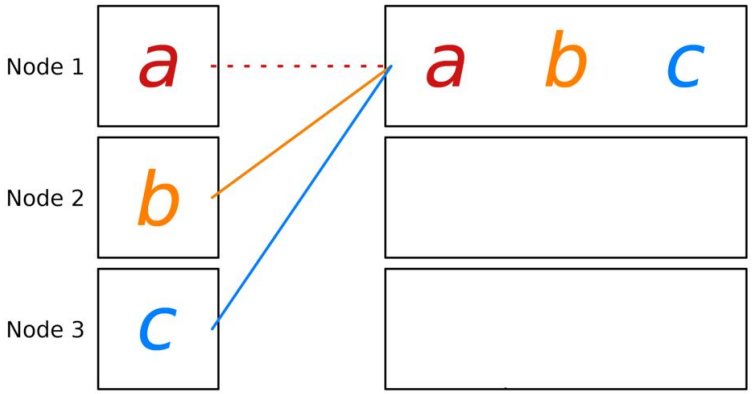
The **Message Passing Interface** (**MPI**) is a portable message-passing standard designed to function on parallel computing architectures.

- Стандарт (спецификация) API для обмена сообщениями между процессами в распределённых приложениях
- Модель: процессы/ранги, коммуникаторы, p2p (send/recv) и коллективы (all-reduce, bcast и т.д.) нестандартные типы данных
- Реализации: Open MPI, MPICH, MVAPICH2, Intel MPI, IBM Spectrum MPI и др
- Работает поверх разных транспортов: shared memory, TCP, InfiniBand/ROCE, иногда через UCX
- Обычно в LLM Training MPI используют для запуска/bootstrapping

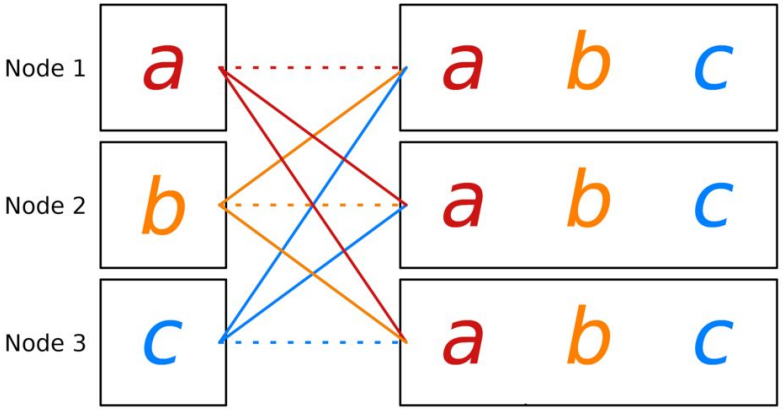
Коллективные операции MPI



MPI_Bcast

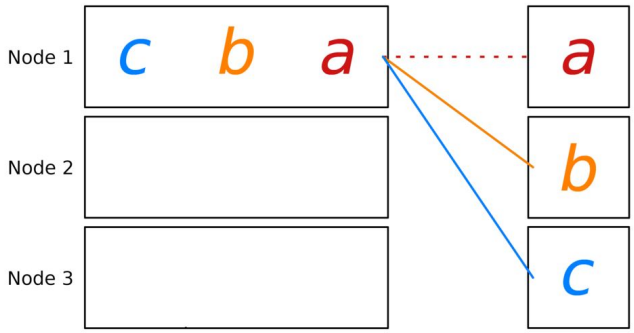


MPI_Gather

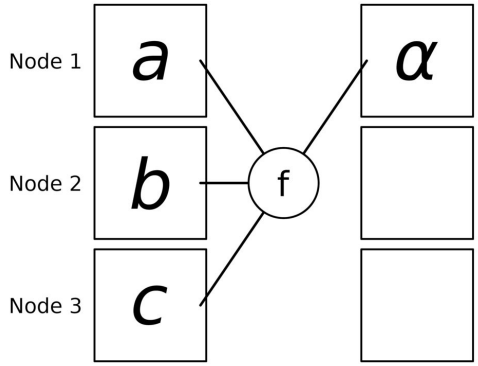


MPI_AllGather

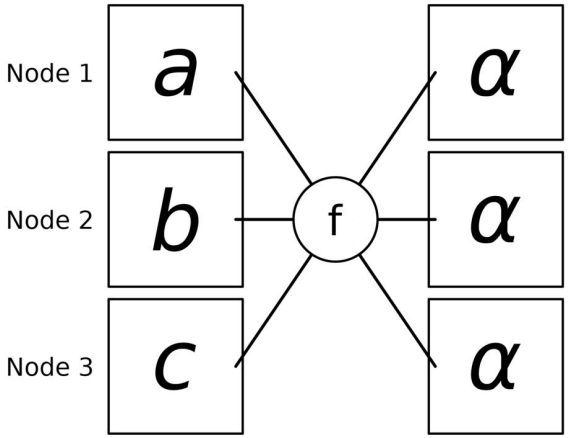
Коллективные операции MPI



MPI_Scatter

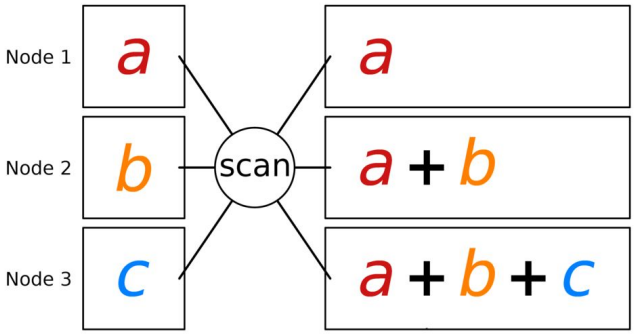


MPI_Reduce



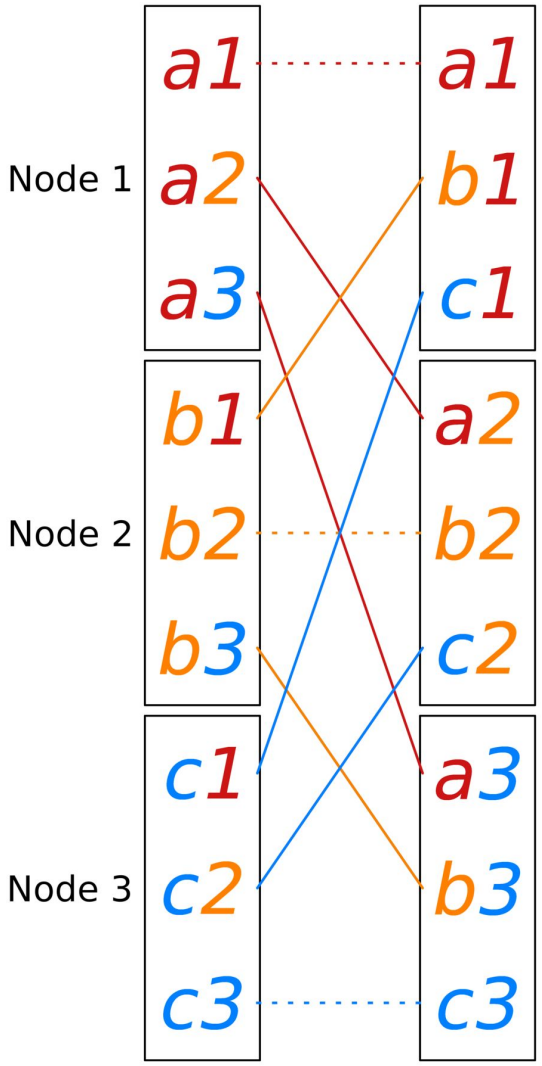
MPI_AllReduce

Коллективные операции MPI



MPI_Scan

+MPI_Barrier
+MPI_Send+MPI_Recv
+MPI_Isend+MPI_Irecv



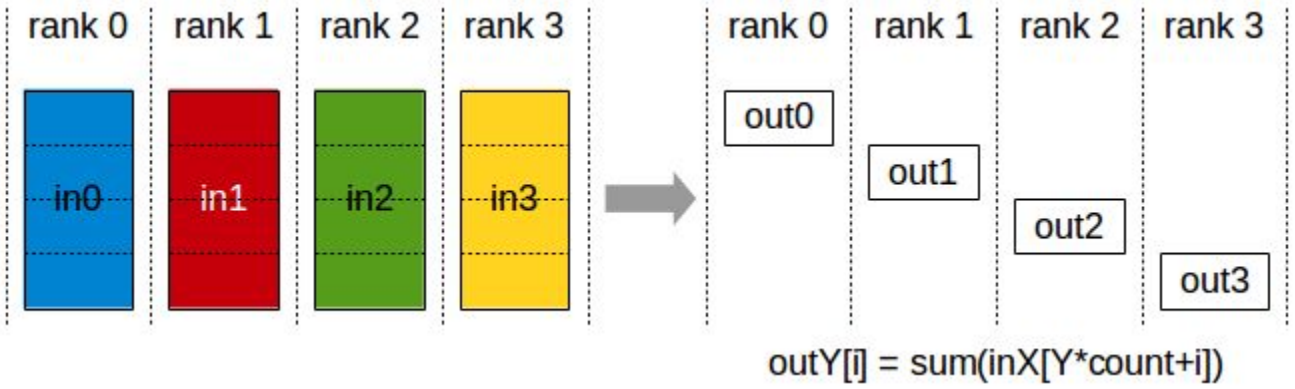
MPI_AllToAll

NCCL

- Высокопроизводительные коллективные и p2p-коммуникации для ускорения обучения на нескольких GPU и нодах
- API уровня устройства для прямого обмена данными, что снижает задержки и улучшает overlap вычислений и коммуникаций
- Автоматическое определение топологии (PCIe, NVLink™, NVSwitch™, InfiniBand, RoCE и др.) для максимальной производительности
- Оптимизация маршрутов с помощью графовых алгоритмов для достижения пиковой пропускной способности и минимальной задержки
- Полная совместимость с многопоточными, многопроцессными и MPI-приложениями

Source: https://developer.nvidia.com/nccl

NCCL Collective Operations



ncclReduceScatter

torchrun

- CLI-лаунчер PyTorch для распределённого запуска: спаунит процессы (обычно 1 процесс на GPU), выставляет RANK/WORLD_SIZE, делает rendezvous
- Поддерживает elastic / fault-tolerant тренинг (перезапуски, snapshotting)
- Используется вместе с torch.distributed (часто backend=nccl) на single- и multi-node

torch.distributed

- **Инициализация процесса** и **процесс-группы**: init_process_group, ранги, WORLD_SIZE.
- **Коллективы и p2p**: all_reduce, all_gather, broadcast, reduce_scatter, send/recv.
- **Бэкэнды**: nccl (GPU), gloo (CPU/отладка), mpi, (есть ucc у некоторых сборок).
- **Хранилища (Store) для rendezvous**: TCPStore, FileStore, EtcdStore.
- **Высокоуровневые обёртки**: **DDP** (torch.nn.parallel.DistributedDataParallel), **FSDP** (torch.distributed.fsdp), checkpointing.
- **Elastic/fault-tolerant запуск** через torch.distributed.run (**torchrun**) и TorchElastic.

Как собрать это в кучу?

- OS (Ubuntu)
- SLURM / Kuber (+ nvidia gpu operator)
- MPI, torchrun
- torch.distributed

Runtime

Почему важна стабильность?

“If a single node fails, it can interrupt the entire job, resetting the training progress to the last checkpoint and wasting precious compute time. In a 1,024-GPU cluster, this means 1,023 healthy GPUs remain idle while the failed node is restored or replaced.”

The data from the Revisiting Reliability in Large-Scale Machine Learning Research Clusters paper illustrates this fact clearly. Their Mean Time To Failure (MTTF) metric on different cluster scales is given below:

- On 1024 GPUs, **MTTF is 7.9 hours**
- On 16,384 GPUs, **MTTF is 1.8 hours**
- On 131,072 GPUs, **MTTF is 14 minutes**

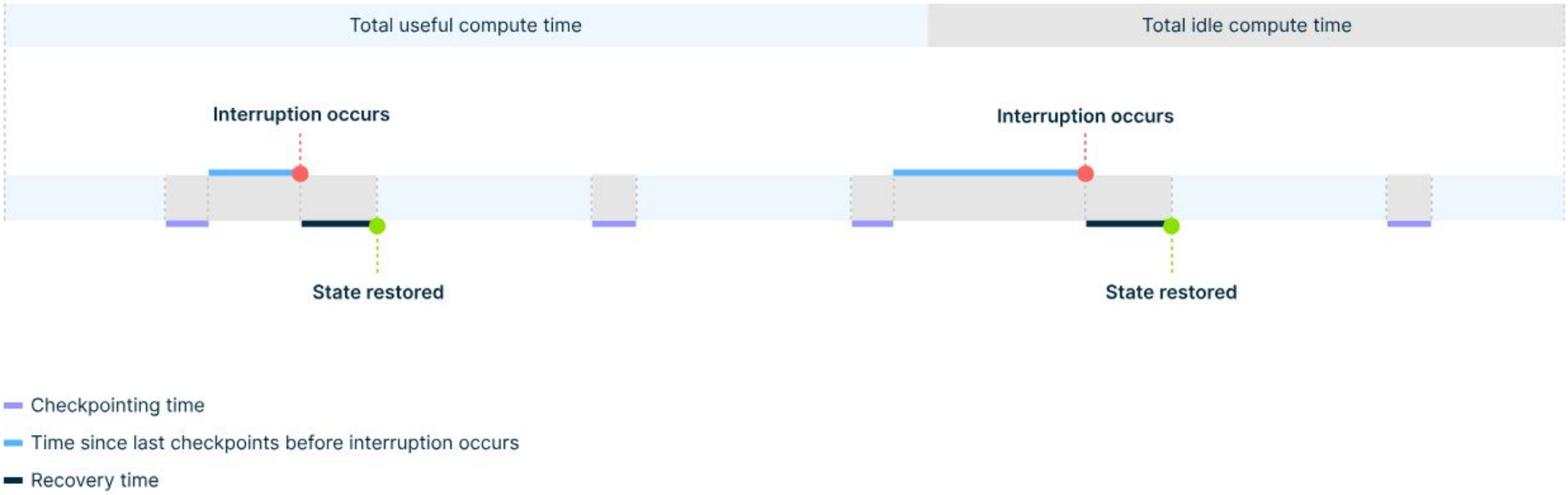
Source: https://nebius.com/blog/posts/how-we-build-reliable-clusters

Какие бывают ошибки?

Failure symptoms	Failure domain			Likely failure cause
	User program	System software	Hardware infra	
OOM	✓	x	x	User bug
GPU unavailable	x	✓	✓	PCIe error, driver/BIOS, thermals
GPU memory errors	x	x	✓	Thermal noise, cosmic rays, HBM defect or wear
GPU driver/firmware error	x	✓	x	Outdated software, high load
GPU NVLink error	x	x	✓	Electro/material failure, switch
Infiniband link	x	x	✓	Electro/material failure, switch
Filesystem mounts	x	✓	x	Failed frontend network, drivers in D state, storage backend
Main memory errors	x	x	✓	Circuit wear, thermal noise, cosmic rays
Ethlink errors	x	x	✓	Electro/material failure, switch
PCIe errors	x	x	✓	GPU failure, poor electrical contacts
NCCL timeout	✓	✓	✓	Userspace crash, deadlock, Failed hardware
System services	✓	✓	✓	Userspace interference, software bugs, network partition

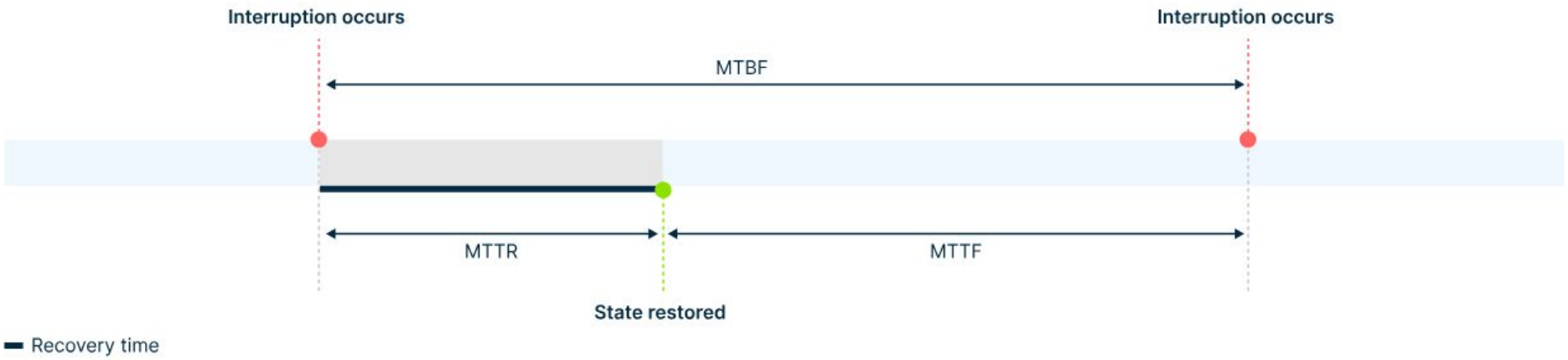
Source: https://nebius.com/blog/posts/how-we-build-reliable-clusters

Как измерить стабильность?



Source: https://nebius.com/blog/posts/how-we-build-reliable-clusters

Как измерить стабильность?



Source: https://nebius.com/blog/posts/how-we-build-reliable-clusters

Здоровье кластера

До обучения:

- On-site factory tests, Node deployment tests,
- NVLink, IB tests
- MLPerf, LINPACK, NCCL tests

Во время обучения:

- GPU: температура, энергопотребление
- Утилизация
- Логи

Q&A