分布式训练系列

集群通信软硬件

ZOMI

关于本内容

1. 内容背景

- AI集群+大模型+分布式训练系统

2. 具体内容

- **AI集群服务器架构**：参数服务器模式 – 同步与异步并行 - 环同步算法
- **AI集群软硬件通信**：通信软硬件实现 - 通信实现方式
- **分布式通信原语**：通信原语
- **框架分布式功能**：并行处理硬件架构 – AI框架中的分布式训练

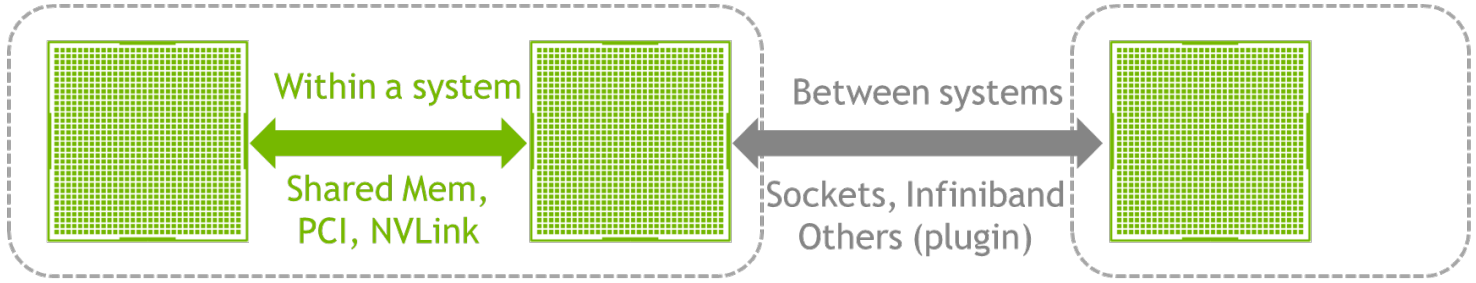
- **大模型算法**：挑战 – 算法结构 – SOTA大模型
- **分布式并行**：数据并行 – 张量并行 – 自动并行 – 多维混合并行

通信系统

- 计算机网络通信中最重要两个衡量指标主要是 **带宽** 和 **延迟。**
- 分布式训练中需要传输大量的网络模型参数。

通信实现方式

- 机器内通信
  - 共享内存
  - PCIe
  - NVLink（直连模式）
- 机器间通信
  - TCP/IP网络
  - RDMA网络（直连模式）

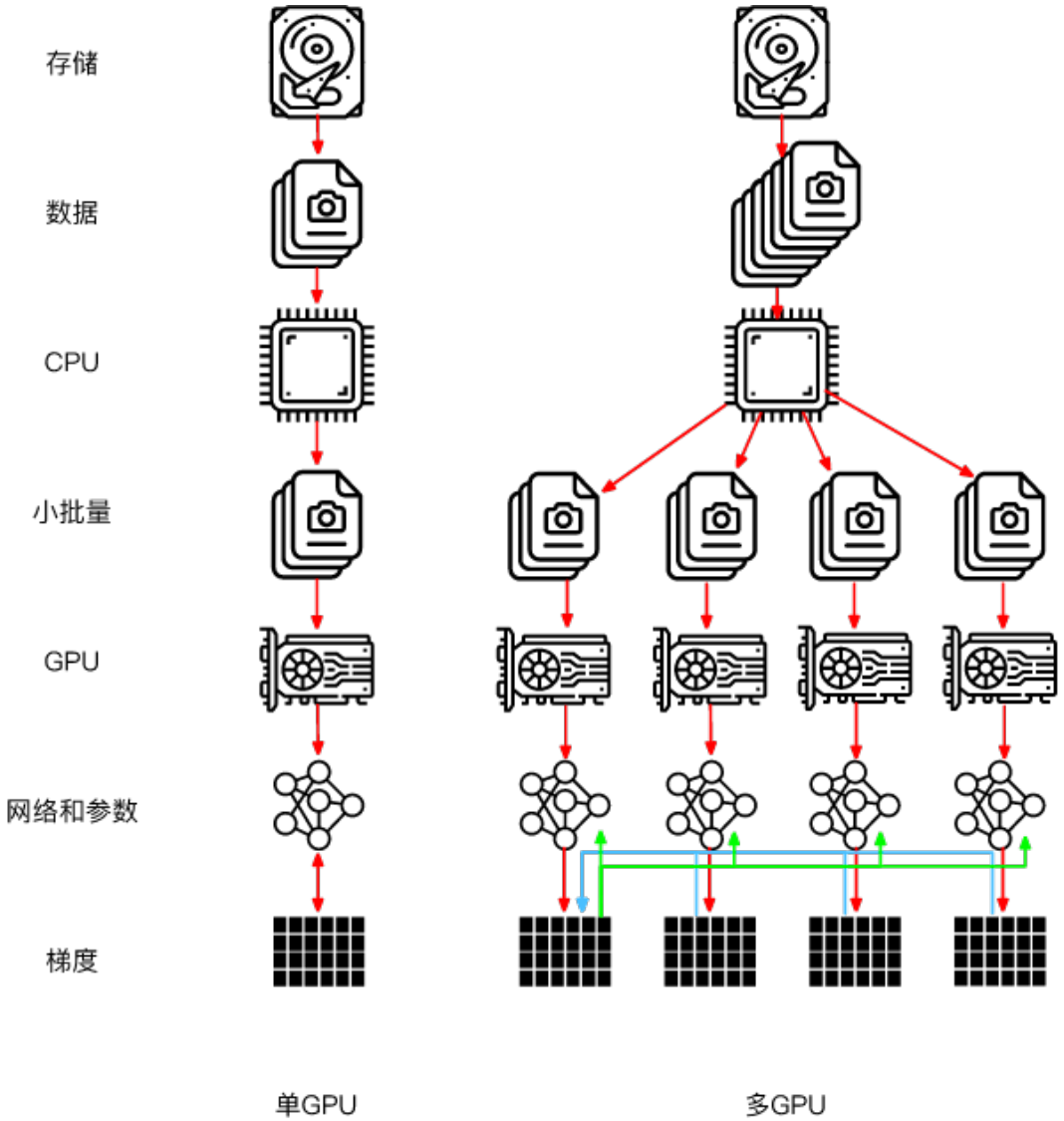


6-9 QPI (shared memory)	1.2 10GbE, TCP/IP Sockets
9-12 PCI Express Gen3 x16	12 100Gb IB or RoCE, RDMA (IB verbs)
42 NVLink/P9, V100 (P2P)	24 2x 100Gb (Summit)
62 NVLink,P100 (P2P)	47 4x 100Gb (DGX1)
132 NVLink,V100 (P2P)	82 8x 100Gb (DGX2)
247~262 NVLink, A100 (P2P)	~185* 8x 200Gb (DGX A100)

Effective bandwidth in GB/s

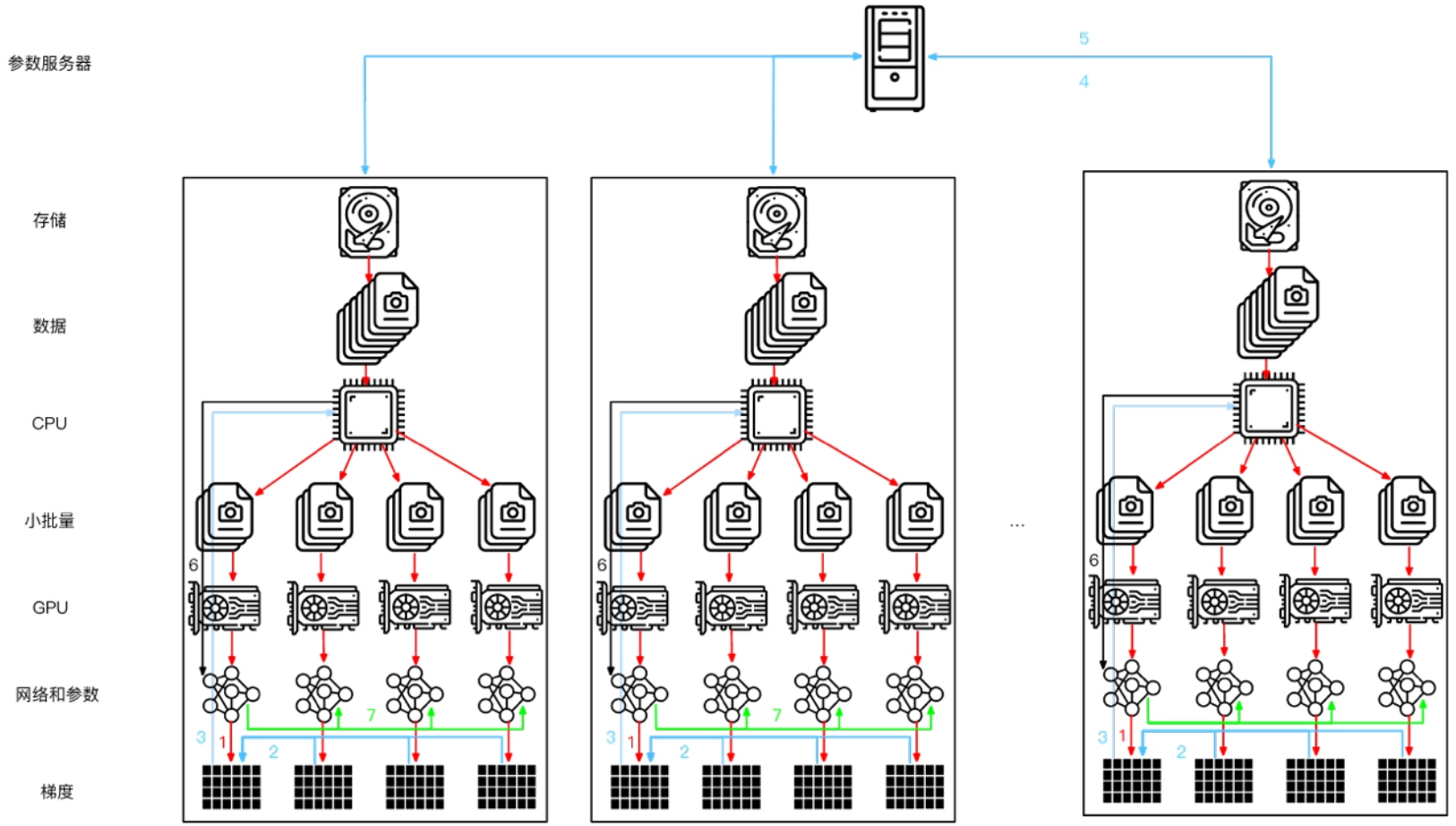
通信实现方式（I）

- 机器内通信
  - 共享内存
  - PCIe
  - NVLink（直连模式）
- 机器间通信
  - TCP/IP网络
  - RDMA网络（直连模式）

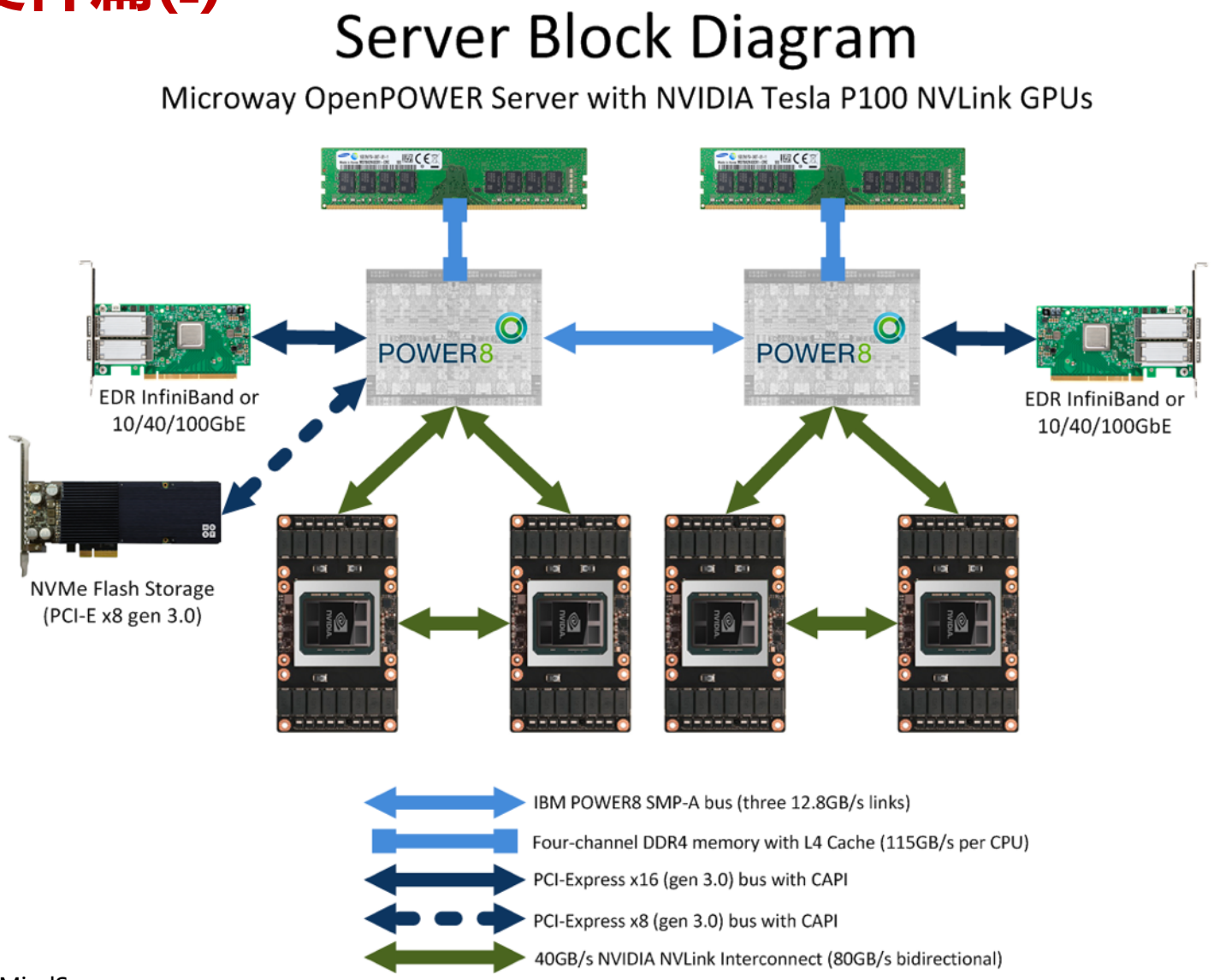


通信实现方式（II）

- 机器内通信
  - 共享内存
  - PCIe
  - NVLink（直连模式）
- 机器间通信
  - TCP/IP网络
  - RDMA网络（直连模式）

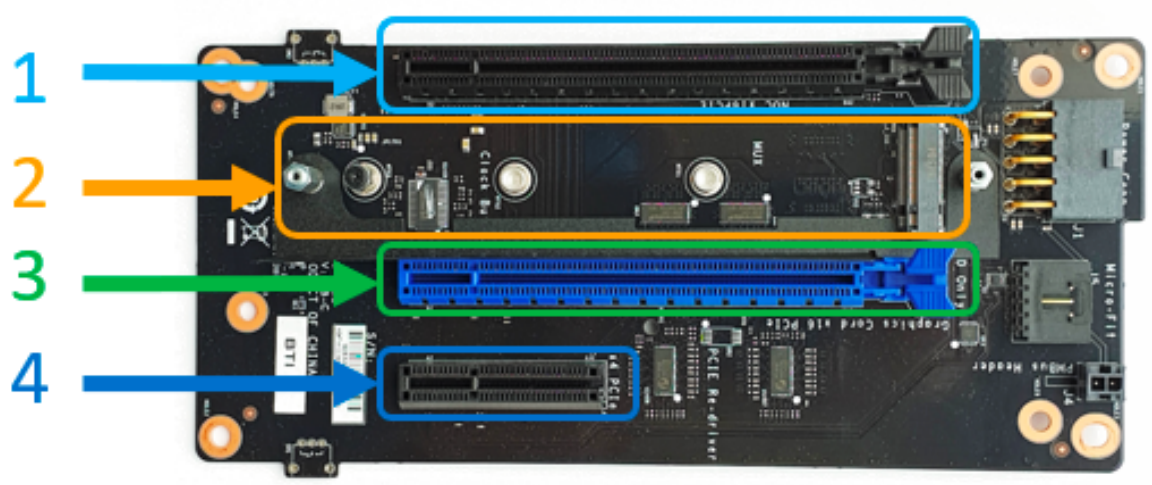


通信协调：硬件篇(I)



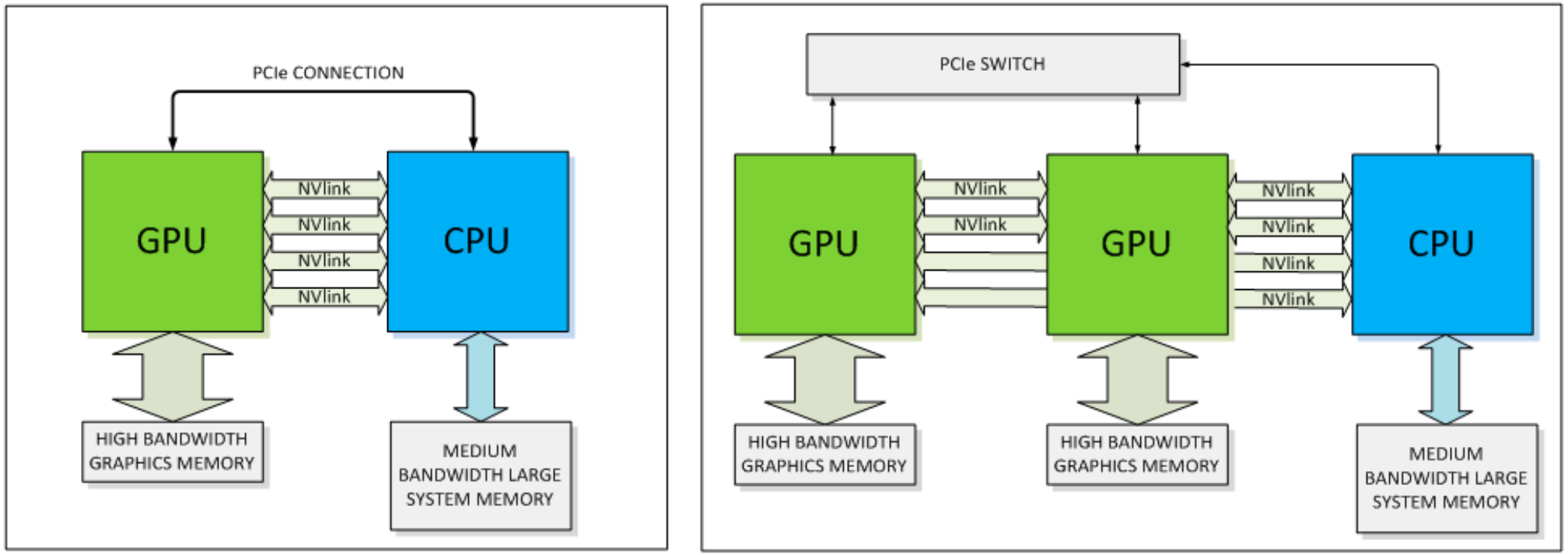
通信协调：硬件篇(I) PCIe

PCIe Architecture	Raw Bit Rate	Interconnect Bandwidth	Bandwidth Lane Direction	Total Bandwidth for x16 link
PCIe 1.1	2.5 GT/s	2 Gb/s	~250 MB/s	~8 GB/s
PCIe 2.0	5.0 GT/s	4 Gb/s	~500 MB/s	~16 GB/s
PCIe 3.0	8.0 GT/s	8 Gb/s	~1 GB/s	~32 GB/s
PCIe 4.0	16.0 GT/s	16 Gb/s	~2 GB/s	~64 GB/s
PCIe 5.0	32.0 GT/s	32 Gb/s	~4 GB/s	~128 GB/s



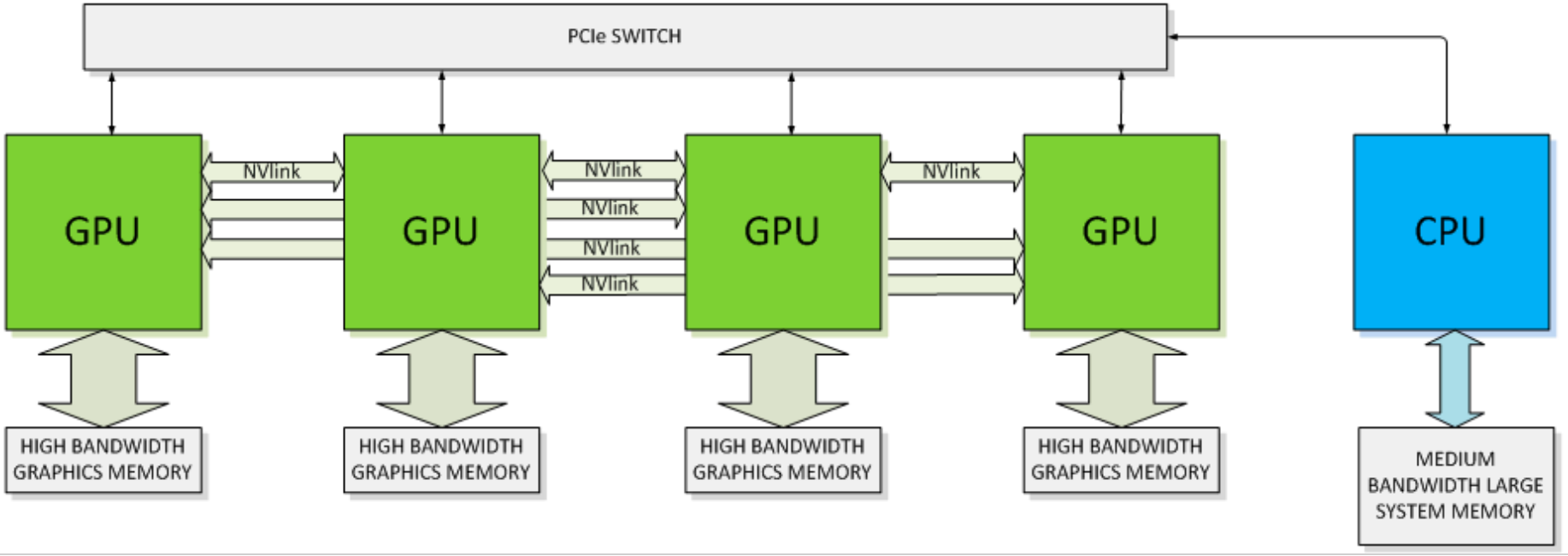
通信协调：硬件篇(II) NVLink

NVLink 的基本构建块是高速、8 通道、差分、双单工双向链路



通信协调：硬件篇(II) NVLink

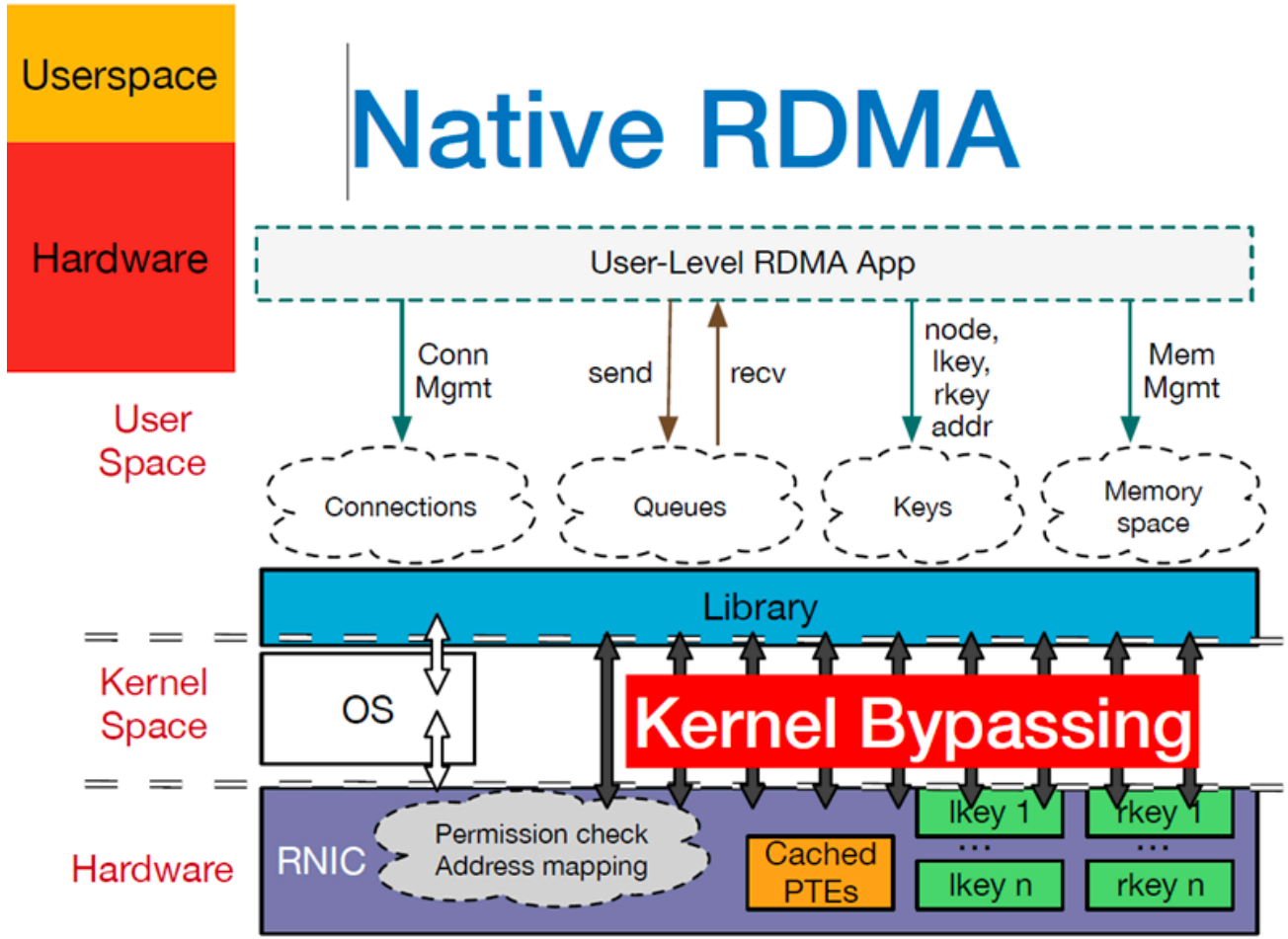
启用 NVLink 的系统中，CPU 发起的事务（如控制和配置）仍然通过 PCIe 连接。保留 PCIe 编程模型，同时在连接带宽方面提供巨大的优势。



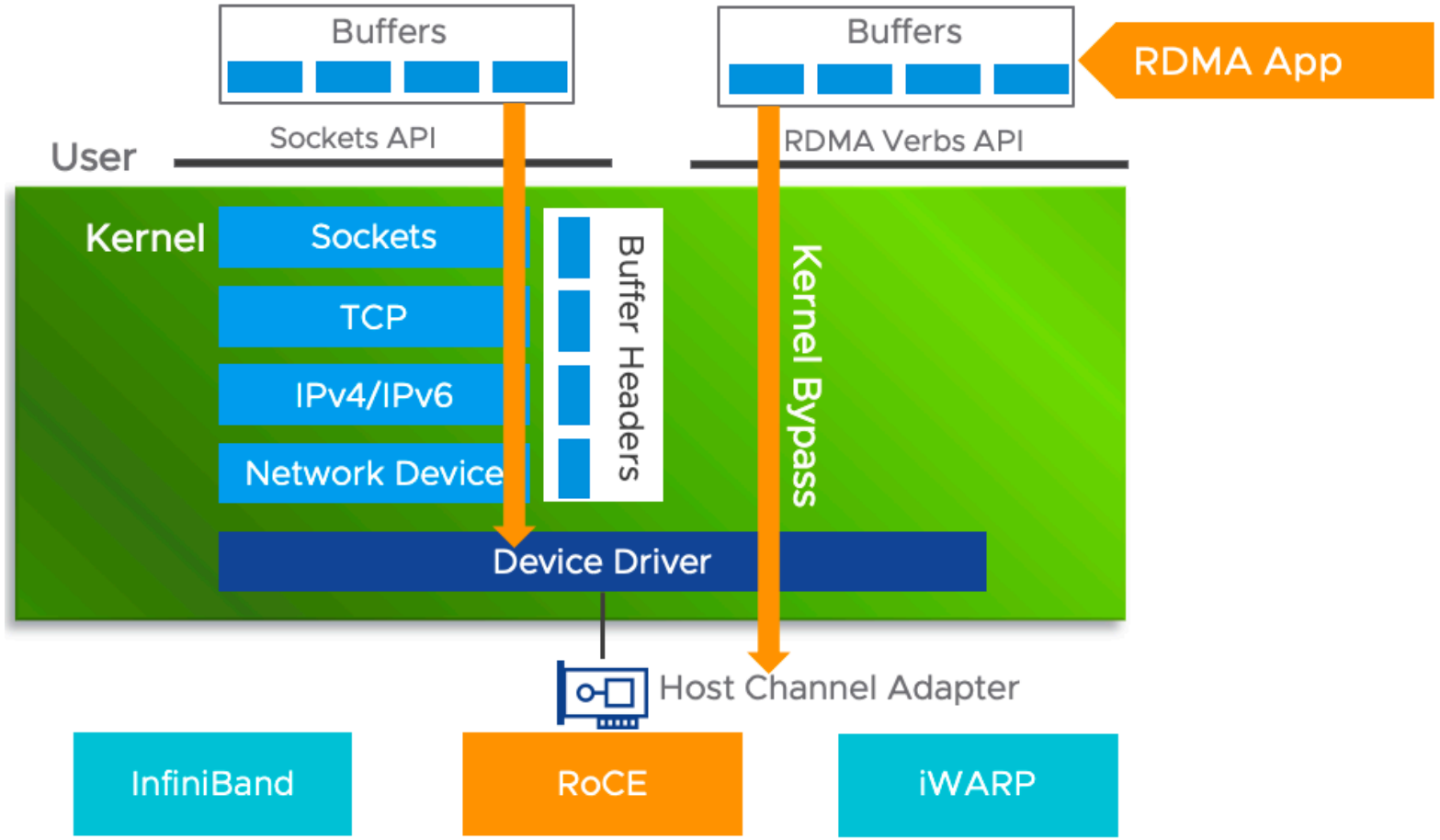
通信协调：硬件篇(III) RDMA (Remote Direct Memory Access)

- **CPU Offload**：无需CPU干预，远程主机CPU缓存(cache)不会被访问的内存内容所填充
- **Kernel Bypass**：专有 Verbs interface ,应用程序可以直接在用户态执行数据传输
- **Zero Copy**：每个应用程序都能直接访问集群中的设备的虚拟内存

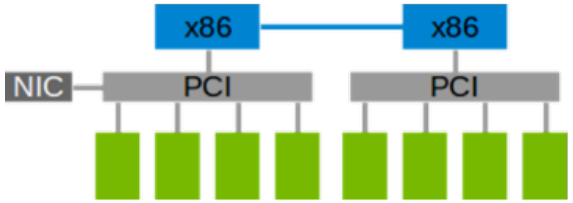
通信协调：硬件篇(III) RDMA (Remote Direct Memory Access)



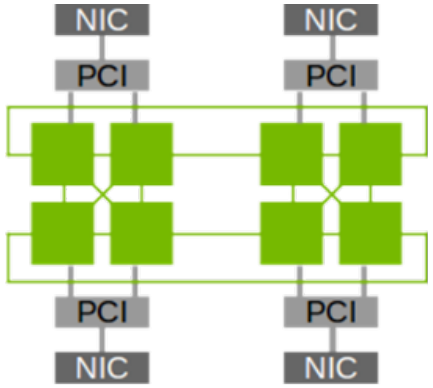
通信协调：硬件篇(III) RDMA (Remote Direct Memory Access)



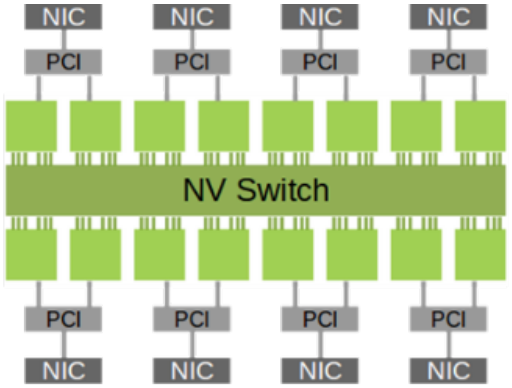
互联拓扑



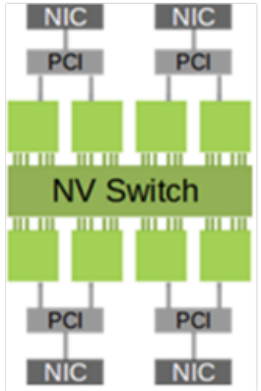
PCI only



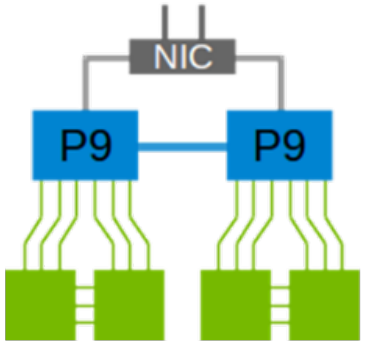
DGX-1



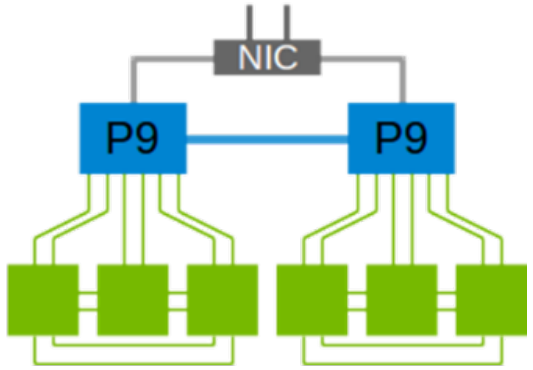
DGX-2



DGX-3



P9/4 V100



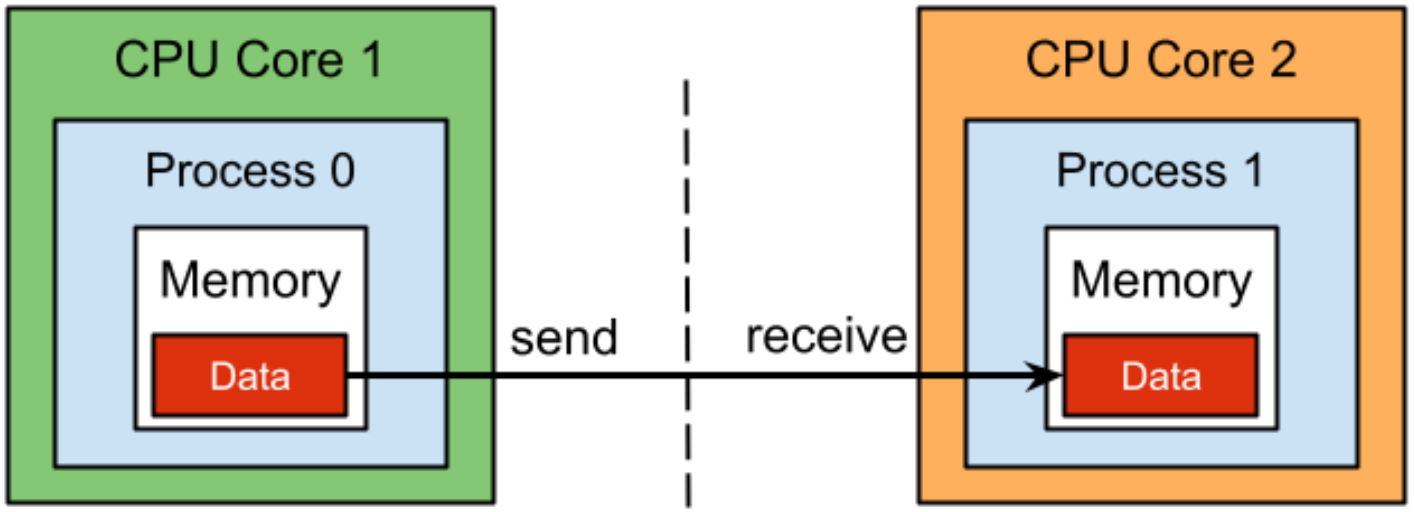
P9/6 V100

通信协调：软件篇

- **MPI**
  - 通用接口，可调用 Open-MPI, MVAPICH2, Intel MPI, etc.
- **NCCL / HCCL**
  - GPU通信优化，仅支持集中式通信
- **Gloo**
  - Facebook 集体通信库,提供集合通信算法如：barrier, broadcast, allreduce

通信协调：软件篇(I) MPI

MPI（Message Passing Interface）：定义了多个原语的消息传递接口，这一接口主要被用于多进程间的通信。MPI 系统通信方式是建立在点对点通信之上。而集合通讯是建立在端到端通信的基础上，在一组进程内的通讯原语。



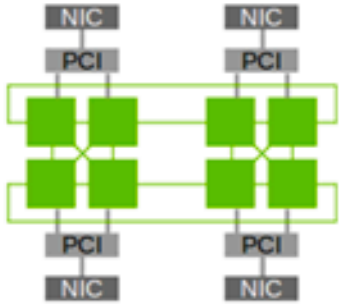
通信协调：软件篇(II) NCCL

NCCL 架构和工作流程： NVIDIA AI 库依赖 NCCL 提供编程抽象，通过高级拓扑检测、通用路径搜索和针对 NVIDIA 架构优化的算法，针对每个平台和拓扑进行高度调整。 NCCL API 从 CPU 启动，GPU 执行，在 GPU 内存之间移动或交换数据。最后利用 NVLink 聚合多个高速 NIC 的带宽。

Topology detection

Build graph with all GPUs, NICs, CPUs, PCI switches, NVLink, NVSwitch.

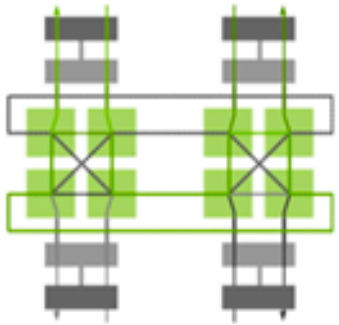
Topology injection for VMs.



Graph search

Extensive search to find optimal set of rings or trees.

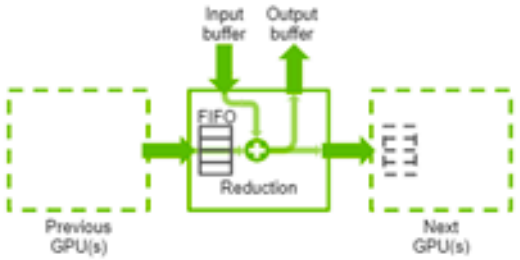
Performance prediction of each algorithm and auto-tuning.



CUDA kernels

Optimized reductions and copies for a minimal SM usage.

CPU threads for network communication.



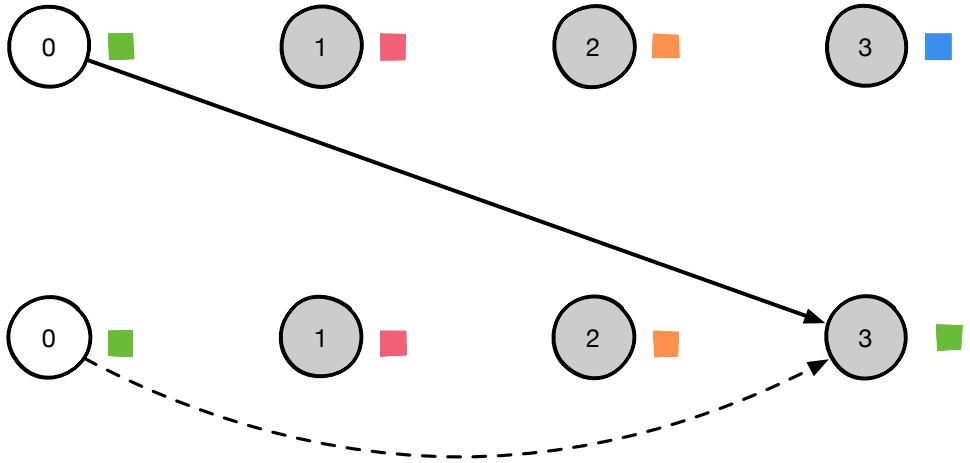
通信协调：软件篇

Backend	gloo		mpi		nccl	
Device	CPU	GPU	CPU	GPU	CPU	GPU
send	✓	✘	✓	?	✘	✓
recv	✓	✘	✓	?	✘	✓
broadcast	✓	✓	✓	?	✘	✓
all_reduce	✓	✓	✓	?	✘	✓
reduce	✓	✘	✓	?	✘	✓
all_gather	✓	✘	✓	?	✘	✓
gather	✓	✘	✓	?	✘	✓
scatter	✓	✘	✓	?	✘	✘
reduce_scatter	✘	✘	✘	✘	✘	✓
all_to_all	✘	✘	✓	?	✘	✓
barrier	✓	✘	✓	?	✘	✓

通信实现方式

- **点对点通信 Send/Recv**
  - TCP/IP
  - RDMA
- **集合式通信 All-Reduce**
  - TCP/IP
  - NCCL

通信实现方式（I）：点对点通信



通信实现方式（I）：点对点通信

PyTorch 点对点通信（同步）

可以实现用户指定的同步send/recv

例如：rank 0 send -> rank 1 recv

```
"""Blocking point-to-point communication."""

def run(rank, size):
    tensor = torch.zeros(1)
    if rank == 0:
        tensor += 1
        # Send the tensor to process 1
        dist.send(tensor=tensor, dst=1)
    else:
        # Receive tensor from process 0
        dist.recv(tensor=tensor, src=0)
    print('Rank ', rank, ' has data ', tensor[0])
```

通信实现方式（I）：点对点通信

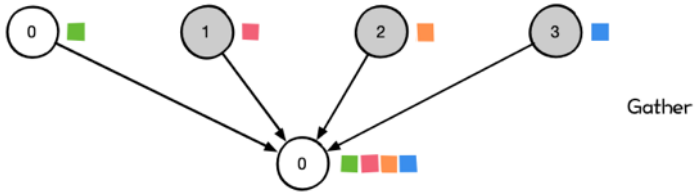
PyTorch 点对点通信（异步）

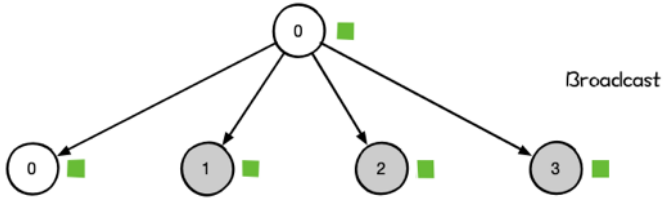
可以实现用户指定的异步send/recv

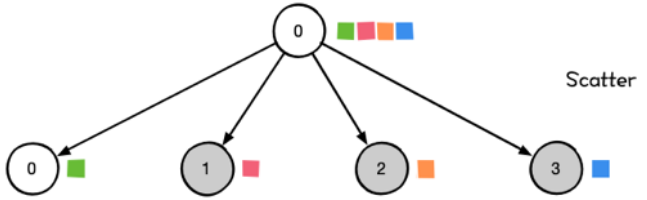
例如：rank 0 send -> rank 1 recv

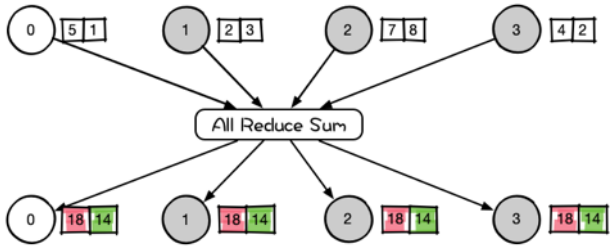
```
"""Non-blocking point-to-point communication."""

def run(rank, size):
    tensor = torch.zeros(1)
    req = None
    if rank == 0:
        tensor += 1
        # Send the tensor to process 1
        req = dist.isend(tensor=tensor, dst=1)
        print('Rank 0 started sending')
    else:
        # Receive tensor from process 0
        req = dist.irecv(tensor=tensor, src=0)
        print('Rank 1 started receiving')
    req.wait()
    print('Rank ', rank, ' has data ', tensor[0])
```

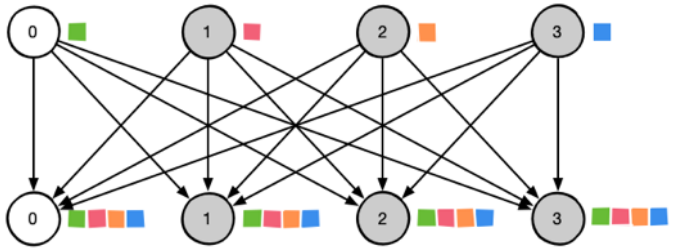
通信实现方式（II）：集合通信

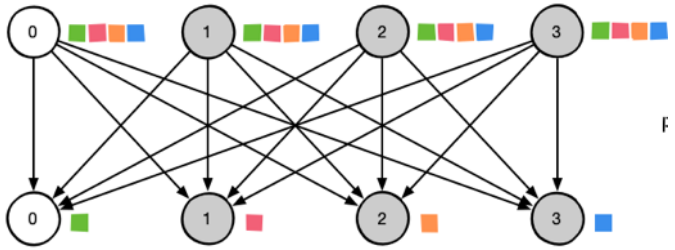












通信实现方式（II）：集合通信

PyTorch 集合通信

```
# Code runs on each rank.
dist.init_process_group("nccl", rank=rank, world_size=2)
output = torch.tensor([rank]).cuda(rank)
s = torch.cuda.Stream()
handle = dist.all_reduce(output, async_op=True)
# Wait ensures the operation is enqueued, but not necessarily complete.
handle.wait()
# Using result on non-default stream.
with torch.cuda.stream(s):
    s.wait_stream(torch.cuda.default_stream())
    output.add_(100)
if rank == 0:
    # if the explicit call to wait_stream was omitted, the output below will be
    # non-deterministically 1 or 101, depending on whether the allreduce overwrote
    # the value after the add completed.
    print(output)
```

Summary

1. 讨论机器内通信、机器间通信的实现方式；
2. 讨论通信协调中硬件通过PCIe、NVLink、RDMA来针对不同场景实现硬件通信能力；
3. 讨论通信协调中软件通过MPI、GLOO、XCCL实现具体通信方式；
4. 讨论点对点通信和集合通信的差异；