



deepseek

Day2: DeepEP

深度解读




ZOMI



 Day 2 of #OpenSourceWeek: DeepEP

Excited to introduce DeepEP - the first open-source EP communication library for MoE model training and inference.

- ✓ Efficient and optimized all-to-all communication
- ✓ Both intranode and internode support with NVLink and RDMA
- ✓ High-throughput kernels for training and inference prefilling
- ✓ Low-latency kernels for inference decoding
- ✓ Native FP8 dispatch support
- ✓ Flexible GPU resource control for computation-communication overlapping

 GitHub: [github.com/deepseek-ai/De...](https://github.com/deepseek-ai/DeepEP)



DeepSeek 开源 DeepEP

- DeepEP 提供高吞吐量和低延迟的 all-to-all Hopper 架构 GPU Kernel, 包括 MoE dispatch and combine。支持 FP8 低精度运算, 特别适用于 DeepSeek 系列模型。
- Github 开源地址: <https://github.com/deepseek-ai/DeepEP>
- 本 PPT 开源: <https://github.com/chenzomi12/AllInfra/>
- 夸克链接: <https://pan.quark.cn/s/374bc7960241>



DeepSeek 开源 DeepEP

1. 高效优化的 All-to-All 通信
2. 支持 NVLink 和 RDMA 的节点内 / 跨节点通信
3. 训练 Training 及推理预填充 Prefill 阶段的高吞吐量计算核心
4. 推理解码 Decoder 阶段的低延迟计算核心
5. 原生支持 FP8 数据分发
6. 灵活控制 GPU 资源，实现计算与通信的高效重叠



视频目录大纲（上）

1. DeepSeek MoE: MoE 架构通信
2. MoE Demo: 原理与实现
3. DeepEP 使用核心工具 (Hopper & NVSCHMEM)
4. DeepEP 之前是怎么用的?



视频目录大纲（下）

1. DeepEP 之前是怎么用的？
2. DeepEP：项目基本介绍
3. DeepEP：代码注释与解读
4. 思考与小结



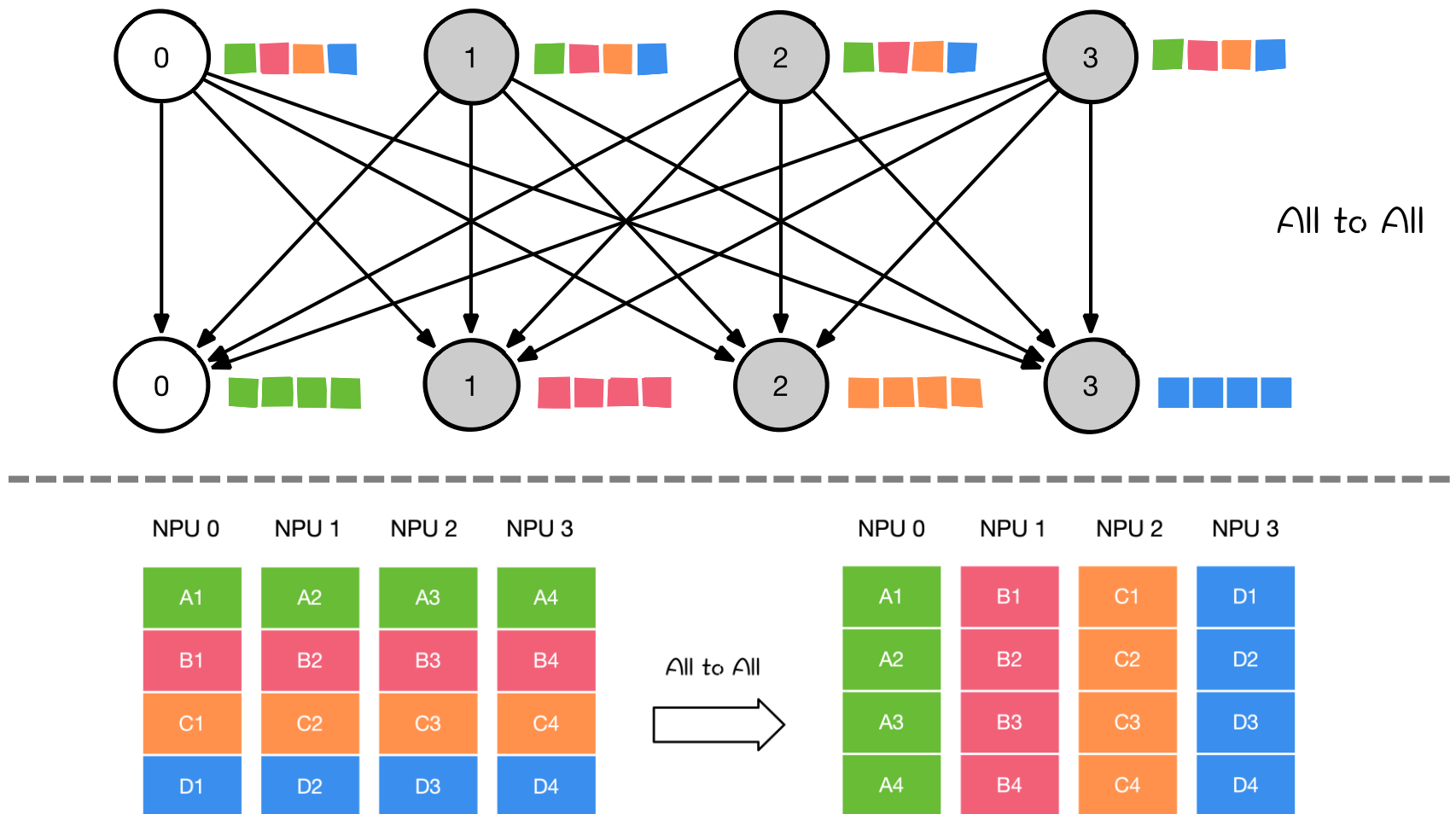
01

DeepSeek MoE 通信原理



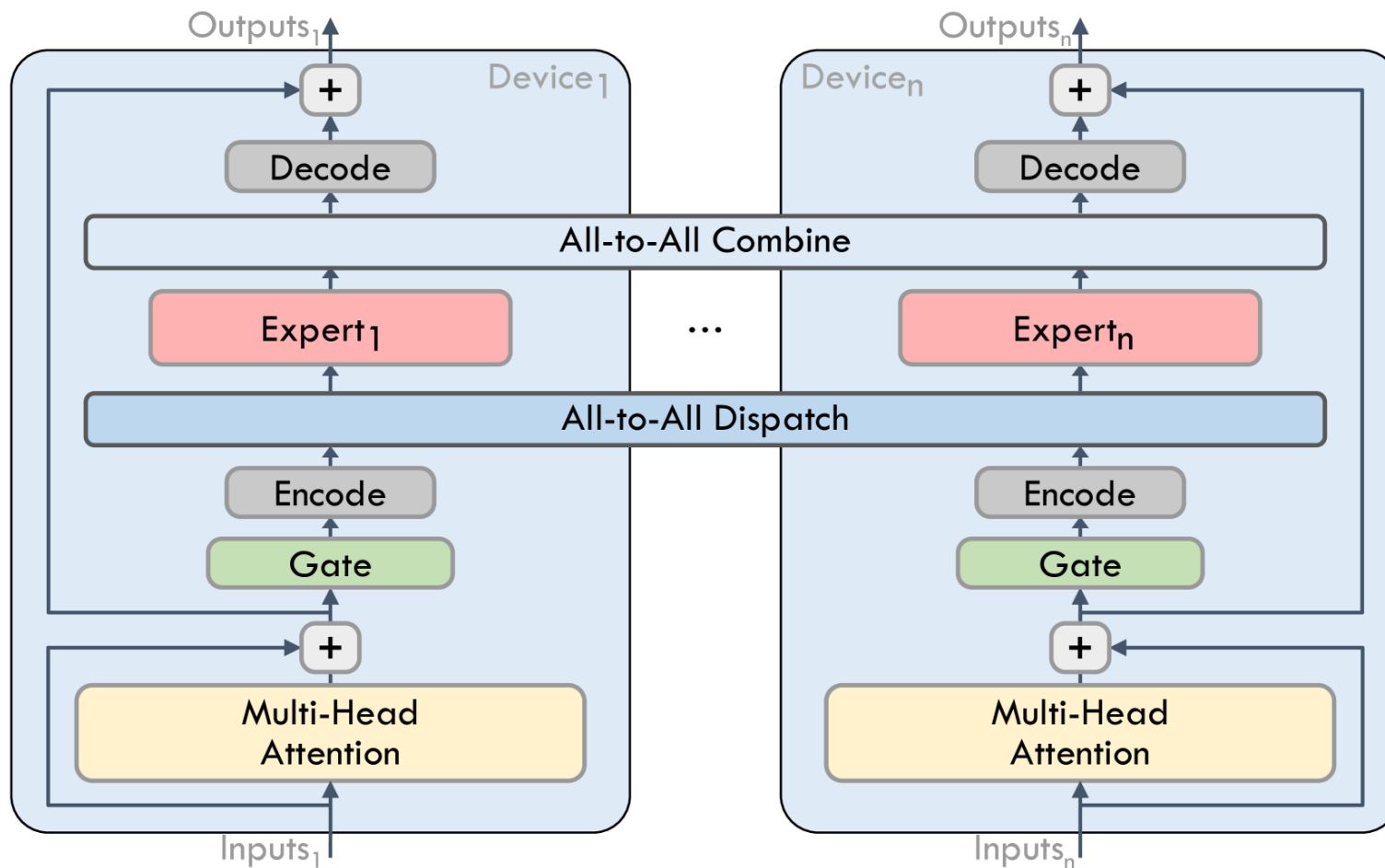
All-to-All 通信

- All-to-All 通信：确保输入数据可以正确分发到各个专家，并将专家的输出结果正确聚合。



MoE (Mixture of Experts)

- MoE (Mixture of Experts) 过程中, All-to-All 通信两个关键步骤是 dispatch 和 combine:



Dispatch 分发

- 在 MoE 模型中，dispatch 目的是将输入数据分发到不同的专家 experts 进行处理。由于每个输入 token 只需要激活 Top-K 个专家，dispatch 需要根据路由结果将数据分发到对应的专家。具体过程：

1. 路由计算：

- 使用一个路由网络（FFN）计算每个 token 对所有专家的分值。
- 根据分值选择 Top-K 个专家，并生成对应的索引和权重。

2. 数据分发：

- 根据 Top-K 索引，将输入数据分发到对应的专家。每个专家会接收到属于自己的输入数据。

3. 通信机制：

- 使用 All-to-All 通信模式，确保每个节点上的数据可以被正确分发到所有其他节点上的专家。



Combine 合并

- Combine 目的是将各个专家的输出结果合并回一个完整的输出张量。由于每个 token 的输出是由 Top-K 个专家的输出加权求和得到的，因此需要将这些部分结果重新组合。具体过程：

1. 结果聚合：

- 根据 Top-K 索引和权重，将每个专家的输出结果聚合到对应的 token 上。
- 通常使用加权求和的方式进行聚合。

2. 通信机制：

- 使用 All-to-All 通信模式，将各个节点上的专家输出数据发送到所有其他节点，以便进行聚合。

02

MoE Demo

实践





03

核心介绍

Hopper & NVSCHMEM



NVIDIA GPU 架构发展

架构名称	Fermi	Kepler	Maxwell	Pascal	Volta	Turing	Ampere	Hopper
中文名字	费米	开普勒	麦克斯韦	帕斯卡	伏特	图灵	安培	赫柏
发布时间	2010	2012	2014	2016	2017	2018	2020	2022
核心参数	16个SM, 每个SM包含32个CUDA Cores, 一共512 CUDA Cores	15个SMX, 每个SMX包括192个FP32+64个FP64 CUDA Cores	16个SM, 每个SM包括4个处理块, 每个处理块包括32个CUDA Cores +8个LD/ST Unit + 8 SFU	GP100有60个SM, 每个SM包括64个CUDA Cores, 32个DP Cores	80个SM, 每个SM包括32个FP64 +64 Int32+64 FP32+8个Tensor Cores	102核心92个SM, SM重新设计, 每个SM包含64个Int32+64个FP32+8个Tensor Cores	108个SM, 每个SM包含64个FP32+64个INT32+32个FP64+4个Tensor Cores	132个SM, 每个SM包含128个FP32+64个INT32+64个FP64+4个Tensor Cores
特点&优势	首个完整GPU计算架构, 支持与共享存储结合的Cache层次GPU架构, 支持ECC GPU架构	游戏性能大幅提升, 首次支持GPU Direct技术	每组SM单元从192个减少到每组128个, 每个SMM单元拥有更多逻辑控制电路	NVLink第一代, 双向互联带宽160 GB/s, P100拥有56个SM HBM	NVLink2.0, Tensor Cores第一代, 支持AI运算	Tensor Core2.0, RT Core第一代	Tensor Core3.0, RT Core2.0, NVLink3.0, 结构稀疏性矩阵MIG1.0	Tensor Core4.0, NVLink4.0, 结构稀疏性矩阵MIG2.0
纳米制程	40/28nm 30亿晶体管	28nm 71亿晶体管	28nm 80亿晶体管	16nm 153亿晶体管	12nm 211亿晶体管	12nm 186亿晶体管	7nm 283亿晶体管	4nm 800亿晶体管
代表型号	Quadro 7000	K80 K40M	M5000 M4000 GTX 9XX系列	P100 P6000 TTX1080	V100 TiTan V	T4, 2080TI RTX 5000	A100 A30系列	H100



Hopper 赫柏架构

- GPC 8组，66组TPC、132组SM，总计有16896个CUDA核心、528个Tensor核心、50MB二级缓存。
显存为新一代HBM3，容量80GB，位宽5120-bit，带宽高达3TB/s



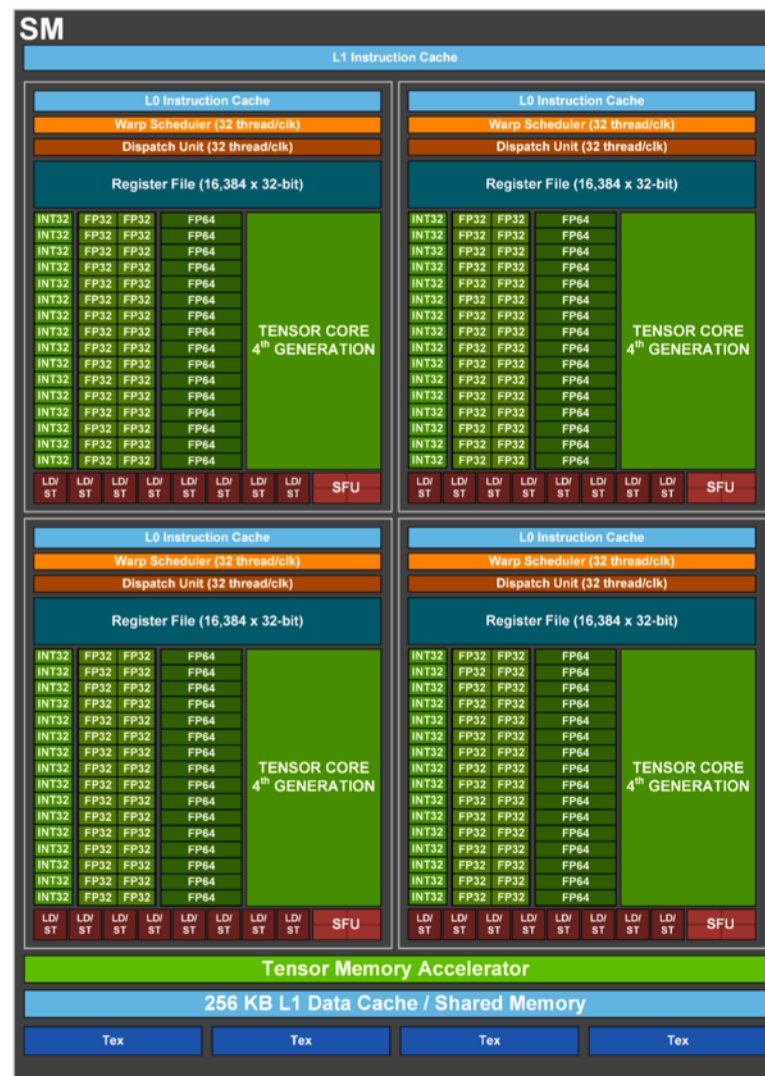
Hopper 赫柏架构

- SM 结构:

1. 4 个 Warp Scheduler, 4 个 Dispatch Unit (与 A100 一致)
2. 128 个 FP32 Core (4 * 32) (相比 A100 翻倍)
3. 64 个 INT32 Core (4 * 16) (与 A100 一致)
4. 64 个 FP64 Core (4 * 16) (相比 A100 翻倍)
5. 4 个 TensorCore (4 * 1)
6. 32 个 LD/ST Unit (4 * 8) (与 A100 一致)
7. 16 个 SFU (4 * 4) (与 A100 一致)
8. 相比 A100 增加了一个 Tensor Memory Accelerator

- 每个 Process Block:

1. 1 个 Warp Scheduler, 1 个 Dispatch Unit (与 A100 一致)
2. 32 个 FP32 Core (相比 A100 翻倍)
3. 16 个 INT32 Core (与 A100 一致)
4. 16 个 FP64 Core (相比 A100 翻倍)
5. 1 个 TensorCore
6. 8 个 LD/ST Unit (与 A100 一致)
7. 4 个 SFU (与 A100 一致)



NVSHMEM 解释

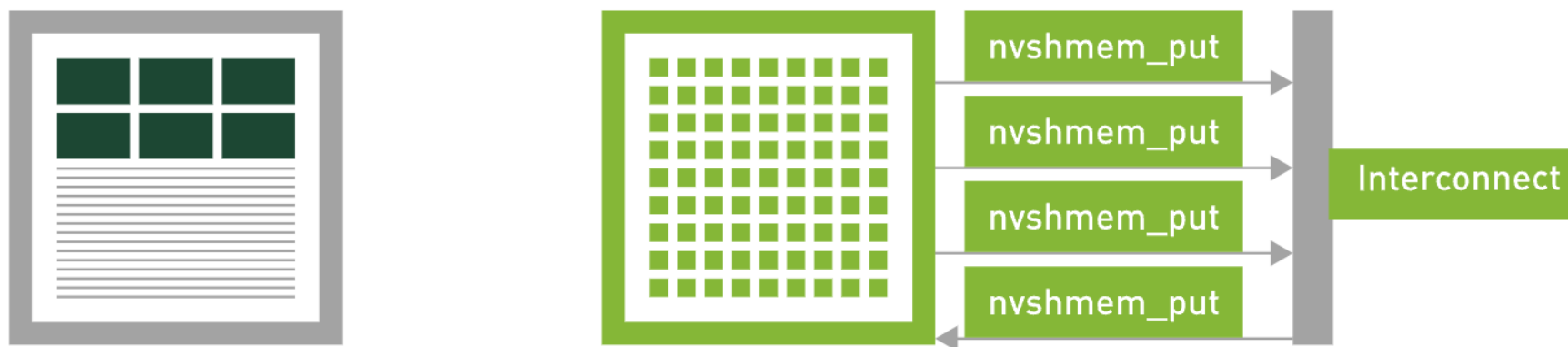
- NVSHMEM 是基于 OpenSHMEM 的并行编程接口，为 NVIDIA GPU 集群提供高效且可扩展的通信。
 - NVSHMEM 为数据创建了一个全局地址空间（a.k.a., PGAS 模型 Partitioned Global Address Space, 分区全局地址空间），该空间跨越多个 GPU 内存，可以通过细粒度的 GPU 发起的操作、CPU 发起的操作以及 CUDA 流上的操作进行访问。
-
- <https://docs.nvidia.com/cuda/gpudirect-rdma/>
 - <https://docs.nvidia.com/nvshmem/api/sla.html>

NVSCHEM介绍

MPI



NVSCHEM



2、NVSHMEM 特性

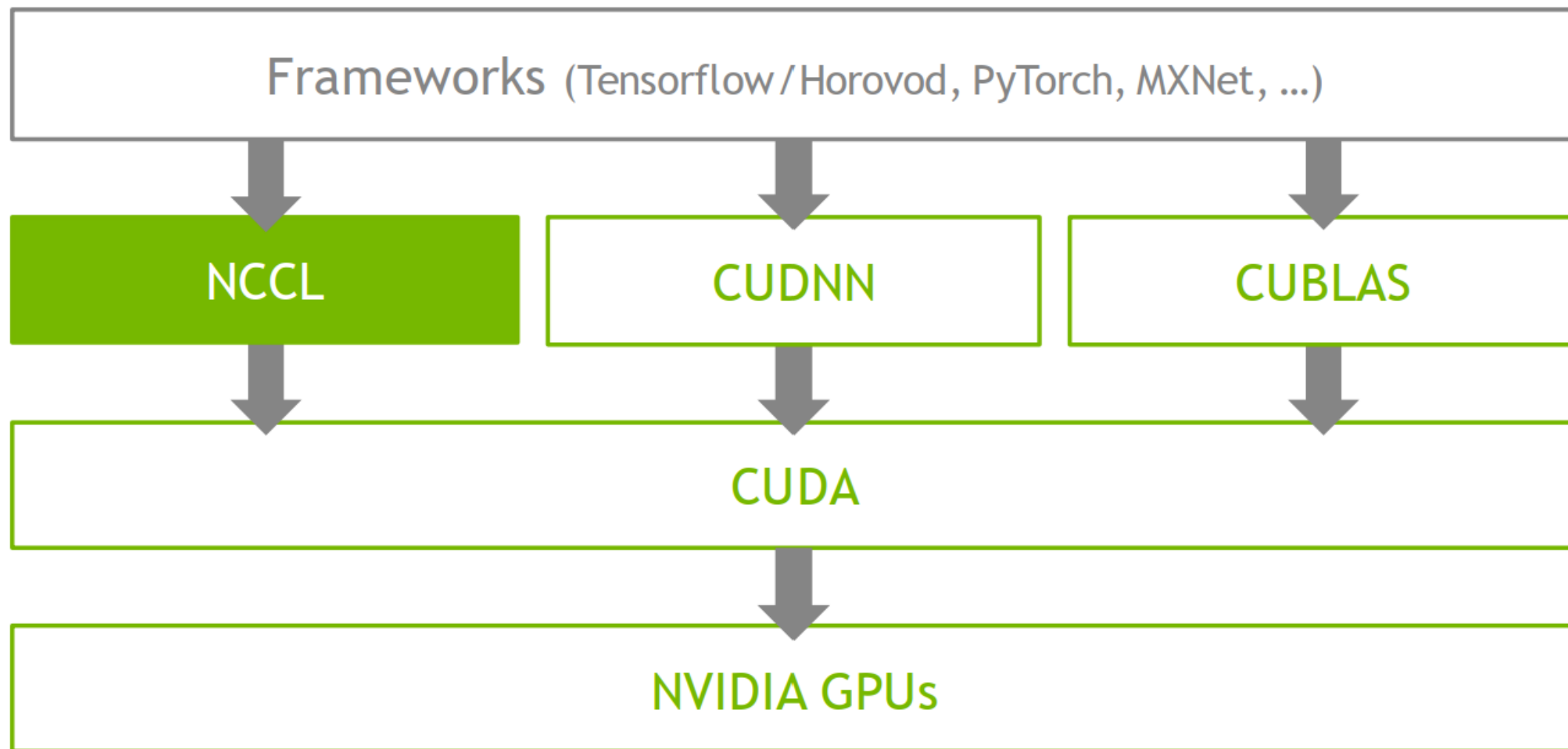
1. 将多个 GPU 的内存组合成一个分区的全局地址空间，可通过 NVSHMEM API 访问
 2. 包含一个低开销的内核通信 API，供 GPU 线程使用
 3. 包括基于流和 CPU 启动的通信 API
 4. 支持 x86 和 Arm 处理器
 5. 可与 MPI 和其他 OpenSHMEM 实现互操作
- NVSHMEM 通过 PGAS 模型和 GPU 直接通信机制，为多 GPU 多节点应用提供了高效的编程接口，尤其适合需要频繁跨设备数据交换的高性能计算场景。尽管需开发者管理内存一致性，但其性能优势和编程便捷性使其成为替代传统 MPI 的重要选择。

04

DeepEP Before & After



NVSCHMEM



为 AI 而生的通信库



NVIDIA/**nccl**



facebookincubator/ **gloo**



microsoft/**msccl**



Alibaba

腾讯

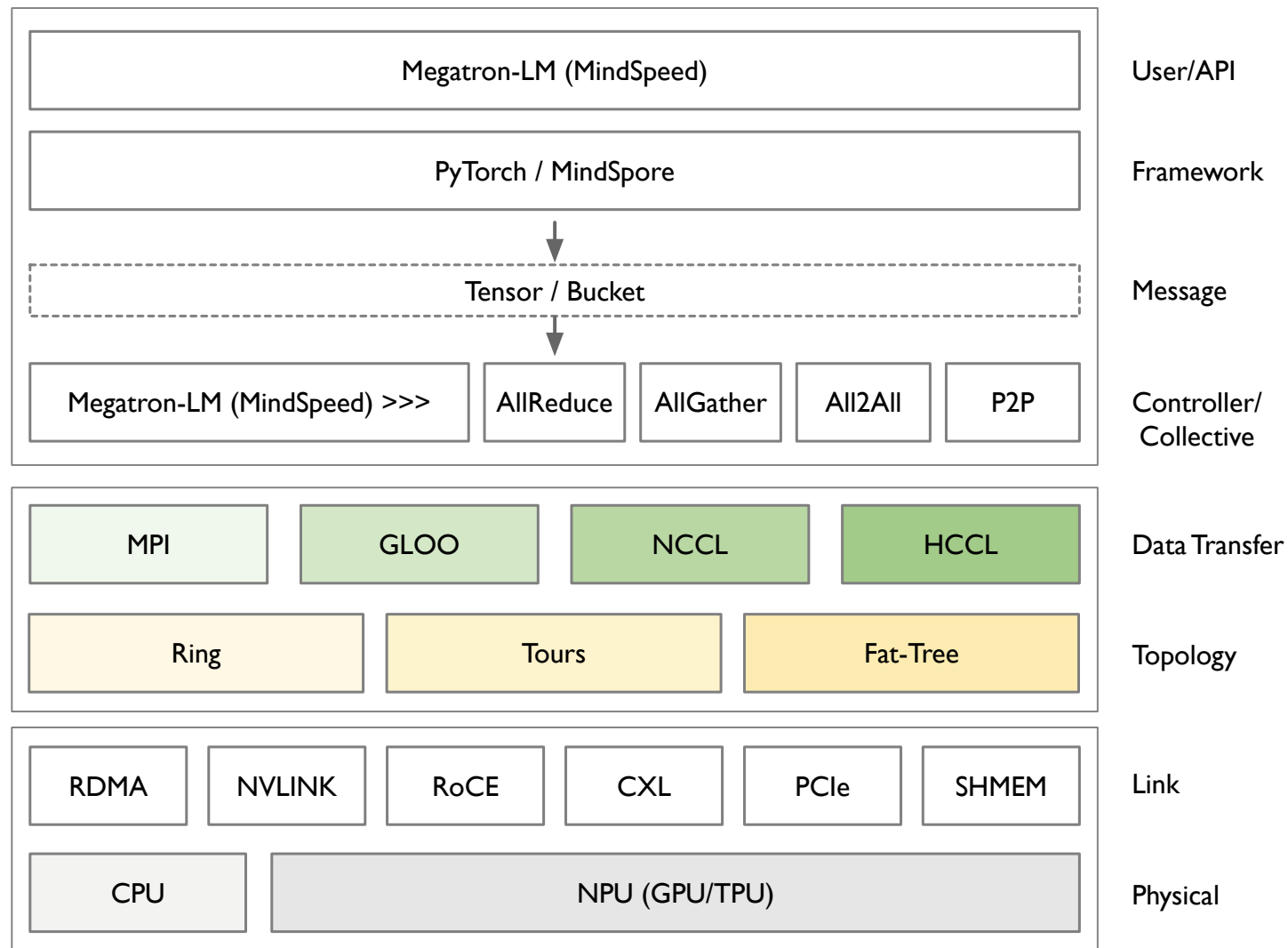
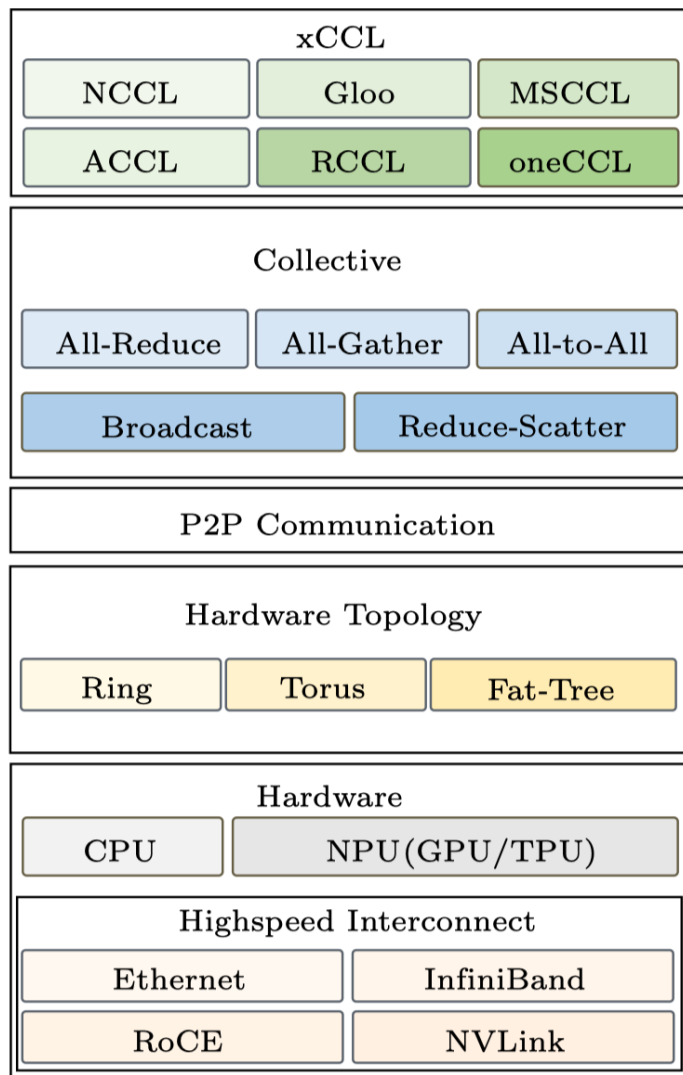
Tencent



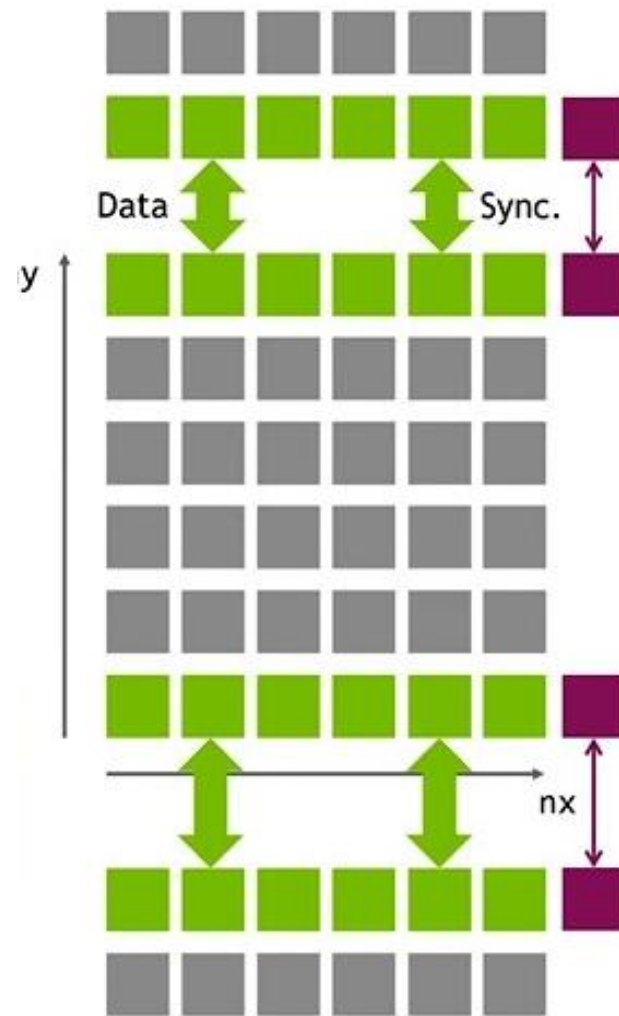
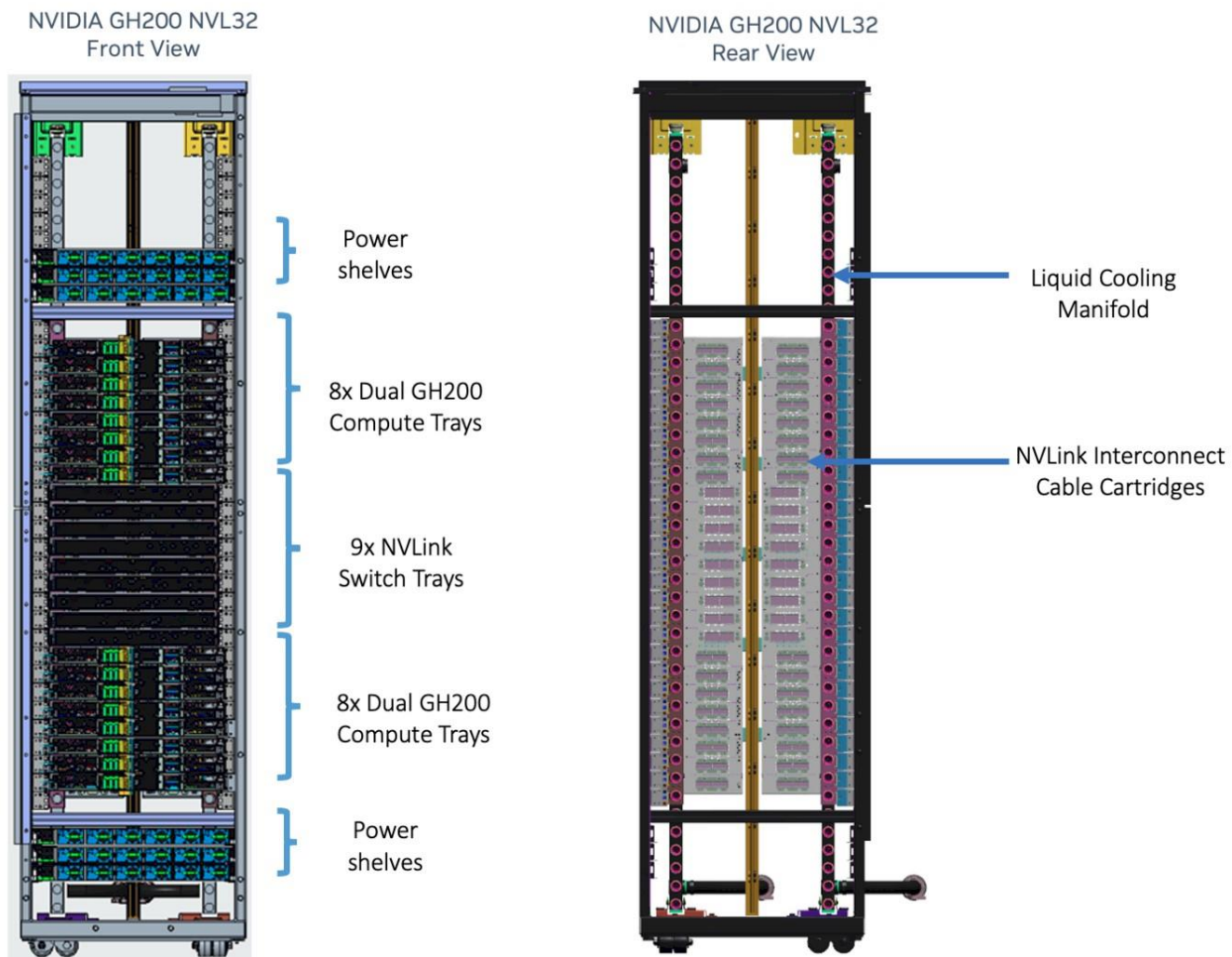
Ascend



XCCL 在 AI 系统中的位置



XCCL 使用 MPI 对等的 API 进行集群管理



NCCL vs NVSHMEM

特性	NCCL	NVSHMEM
主要用途	针对集合通信（如AllReduce、Broadcast）优化，专为深度学习分布式训练设计。	基于PGAS模型的细粒度内存访问，支持任意GPU/节点间的直接内存读写。
设计哲学	强调高吞吐、低延迟的集合操作，适合紧密同步的并行任务。	提供全局地址空间抽象，支持灵活的异步通信，适合非规则或动态通信模式。
信协议	基于NVIDIA NVLink/InfiniBand优化集合通信算法（如Ring AllReduce）。	利用PGAS模型和CUDA-aware技术，直接操作远程内存地址。

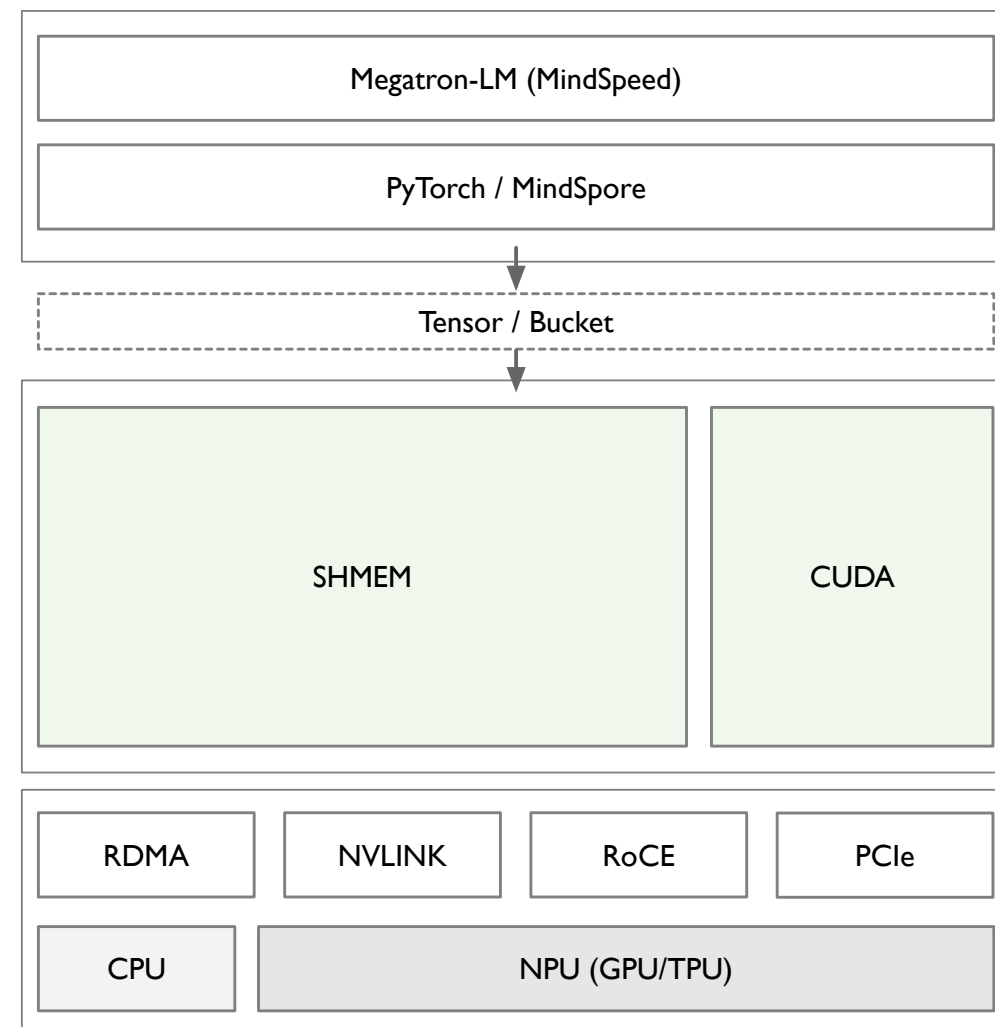
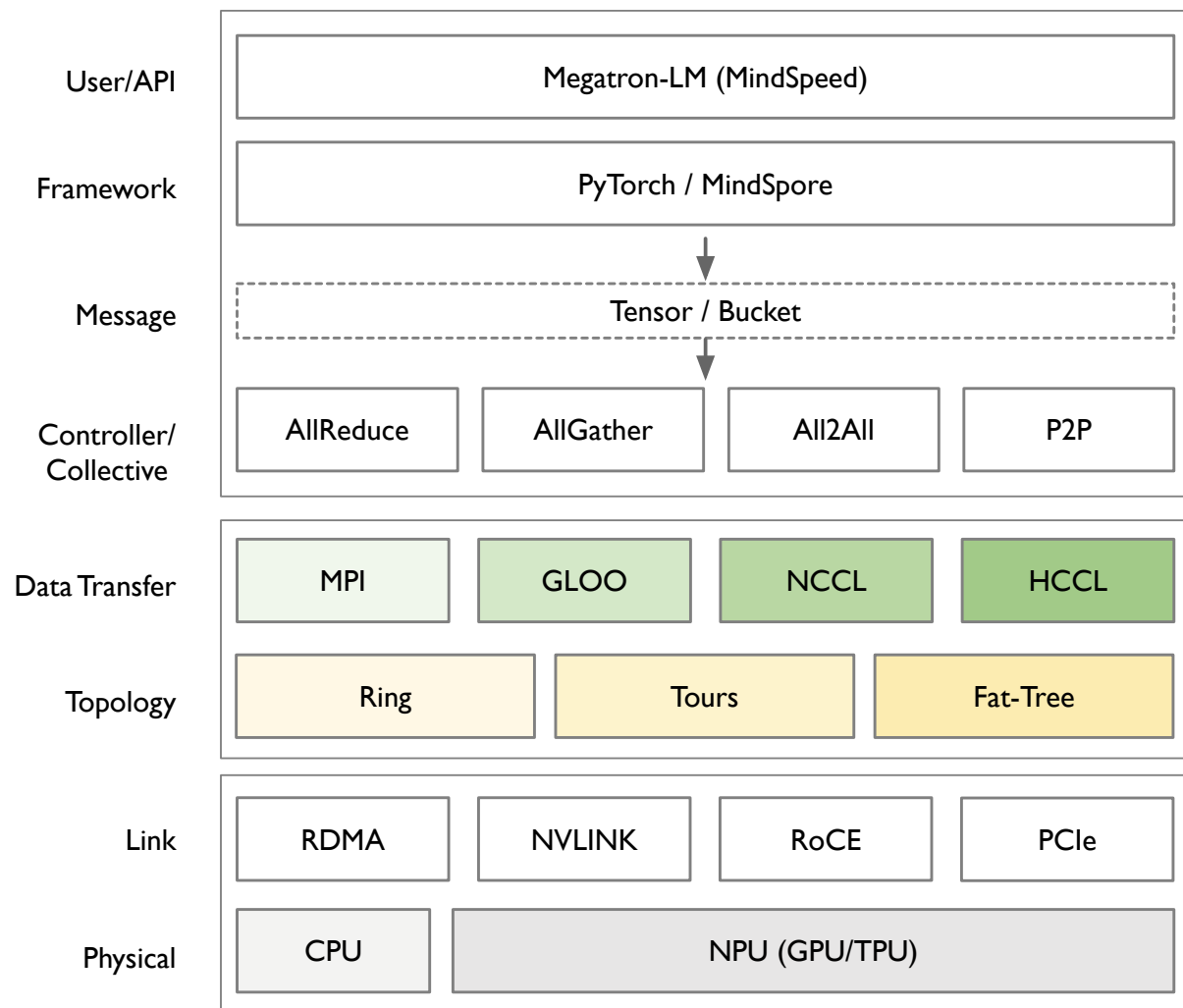


NCCL vs NVSHMEM

特性	NCCL	NVSHMEM
通信粒度	基于集体操作（如AllReduce、AllGather），需要所有进程参与同一操作。	支持单边通信（Put/Get/Atomics），允许单个GPU直接读写远程内存。
同步机制	隐式同步，操作完成后自动保证数据一致性。	需显式同步（如nvshmem_fence或nvshmem_quiet）确保内存可见性。
编程范式	通过显式调用通信函数（如ncclAllReduce）。	类似共享内存的地址直接访问（如nvshmem_put）。



XCCL 在 AI 系统中的位置



End

总结与思考



优化特性



引用与参考

- Github 开源地址: <https://github.com/deepseek-ai/FlashMLA>
- 本 PPT 开源: <https://github.com/chenzomi12/AllInfra/>
- 夸克链接: <https://pan.quark.cn/s/374bc7960241> (代码、论文、注释)





Thank you

把AI系统带入每个开发者、每个家庭、
每个组织，构建万物互联的智能世界

Bring AI System to every person, home and
organization for a fully connected,
intelligent world.

Copyright © 2024 XXX Technologies Co., Ltd.
All Rights Reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. XXX may change the information at any time without notice.



ZOMI

GitHub github.com/chenzomi12/AllInfra



ZOMI