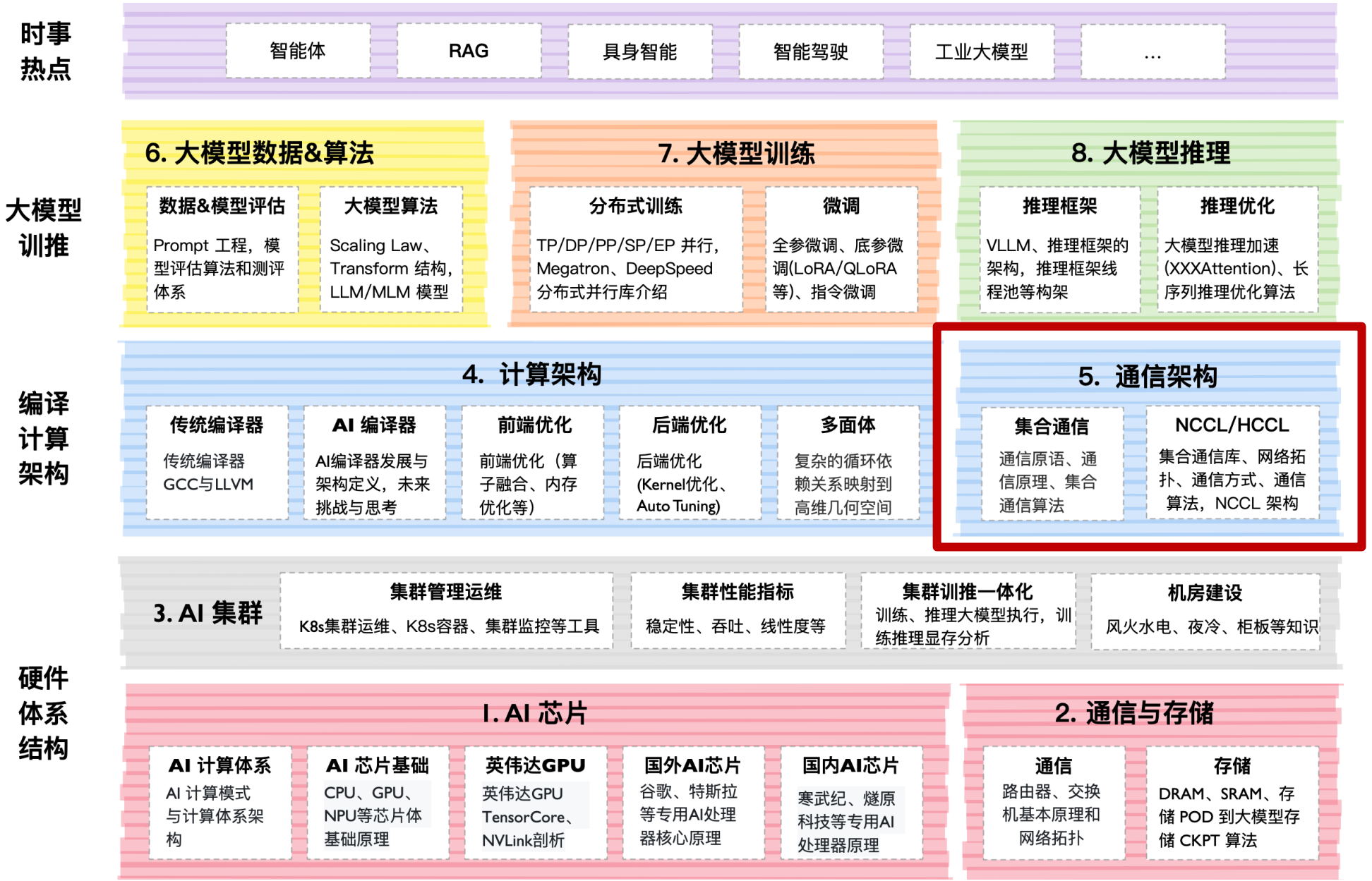
大模型系列 – 集合通信库

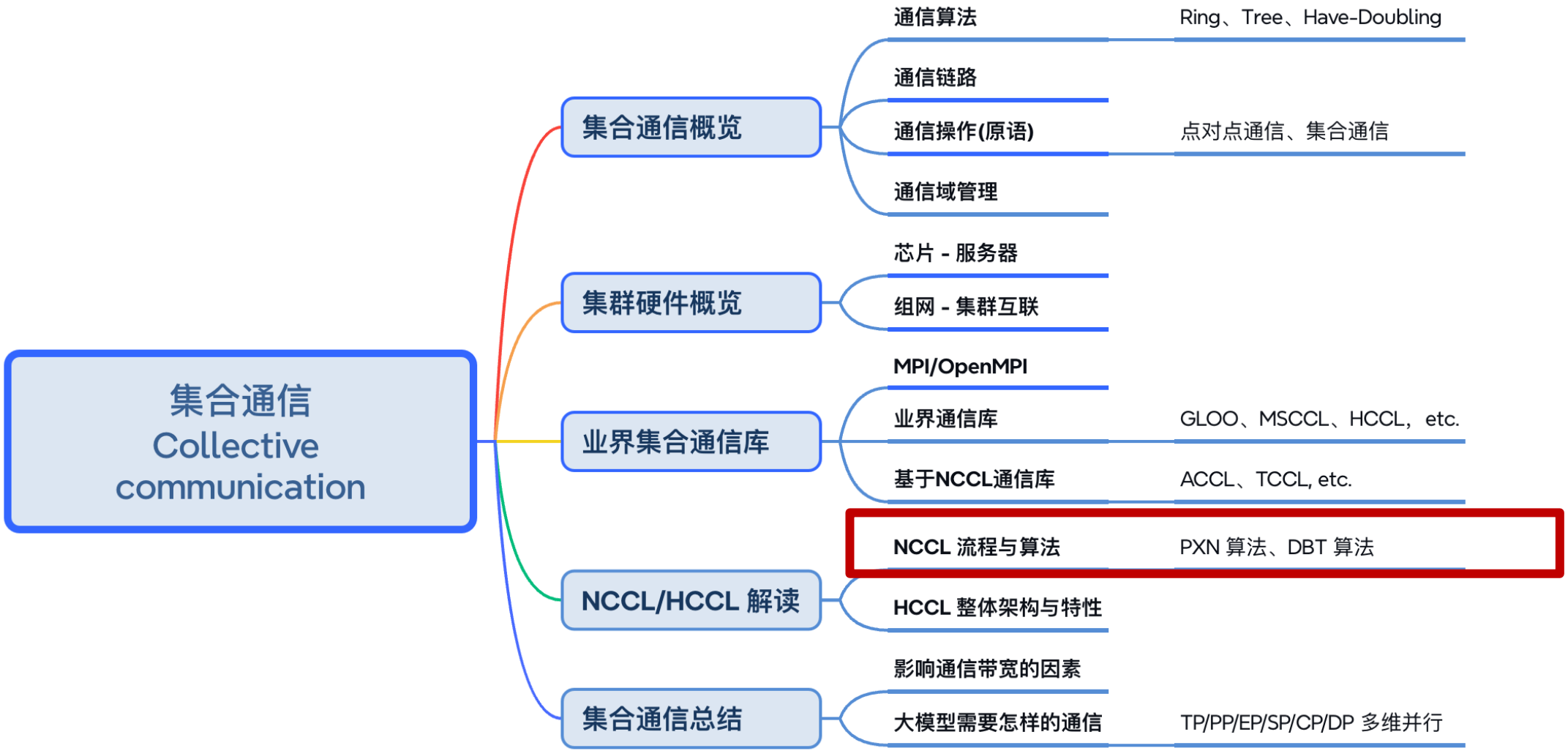
Double Binary Tree

双二叉树

ZOMI


时事热点
智能体
RAG
具身智能
智能驾驶
工业大模型
...
大模型训推
6. 大模型数据&算法
数据&模型评估
Prompt 工程，模型评估算法和测评体系
大模型算法
Scaling Law、Transform 结构，LLM/MLM 模型
7. 大模型训练
分布式训练
TP/DP/PP/SP/EP 并行，Megatron、DeepSpeed 分布式并行库介绍
微调
全参微调、底参微调(LoRA/QLoRA 等)、指令微调
8. 大模型推理
推理框架
VLLM、推理框架的架构，推理框架线程池等构架
推理优化
大模型推理加速(XXXAttention)、长序列推理优化算法
编译计算架构
4. 计算架构
传统编译器
传统编译器 GCC与LLVM
AI 编译器
AI编译器发展与架构定义，未来挑战与思考
前端优化
前端优化（算子融合、内存优化等）
后端优化
后端优化(Kernel优化、Auto Tuning)
多面体
复杂的循环依赖关系映射到高维几何空间
5. 通信架构
集合通信
通信原语、通信原理、集合通信算法
NCCL/HCCL
集合通信库、网络拓扑、通信方式、通信算法，NCCL 架构
硬件体系结构
3. AI 集群
集群管理运维
K8s集群运维、K8s容器、集群监控等工具
集群性能指标
稳定性、吞吐、线性度等
集群训推一体化
训练、推理大模型执行，训练推理显存分析
机房建设
风火水电、夜冷、柜板等知识
1. AI 芯片
AI 计算体系
AI 计算模式与计算体系架构
AI 芯片基础
CPU、GPU、NPU等芯片体基础原理
英伟达GPU
英伟达GPU TensorCore、NVLink剖析
国外AI芯片
谷歌、特斯拉等专用AI处理器核心原理
国内AI芯片
寒武纪、燧原科技等专用AI处理器原理
2. 通信与存储
通信
路由器、交换机基本原理和网络拓扑
存储
DRAM、SRAM、存储 POD 到大模型存储 CKPT 算法

思维导图 XMind


集合通信
Collective
communication
集合通信概览
通信算法
Ring、Tree、Have-Doubling
通信链路
通信操作(原语)
点对点通信、集合通信
通信域管理
集群硬件概览
芯片 - 服务器
组网 - 集群互联
业界集合通信库
MPI/OpenMPI
业界通信库
GLOO、MSCCL、HCCL，etc.
基于NCCL通信库
ACCL、TCCL, etc.
NCCL/HCCL 解读
NCCL 流程与算法
PXN 算法、DBT 算法
HCCL 整体架构与特性
集合通信总结
影响通信带宽的因素
大模型需要怎样的通信
TP/PP/EP/SP/CP/DP 多维并行

Question

1. 英伟达 NCCL 说自己在大规模GPU 集群训练大模型，通信上使用了双二叉树（Double Binary Tree），那么，DBT 是什么？跟我们了解到的 Ring-All Reduce 有什么区别?

本节内容

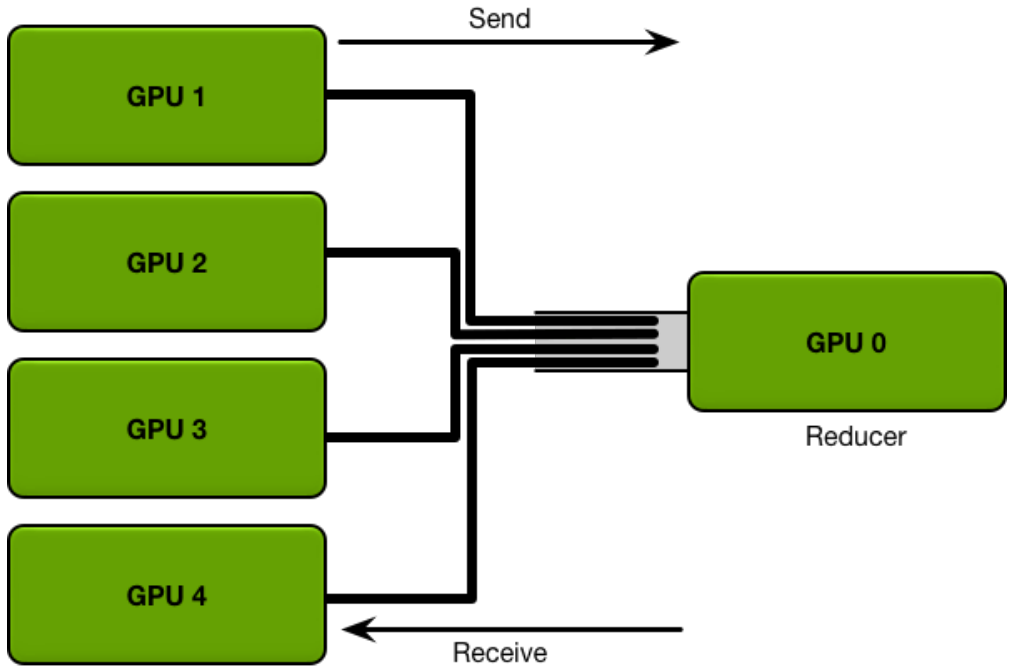
1. Ring 算法 -- Fat ring
2. 分层 Ring 算法 -- Hierarchical rings
3. 朴素二叉树 -- Native binary Tree
4. 双二叉树 -- double binary Tree
5. 具体效果与选择 -- when Tree and Ring

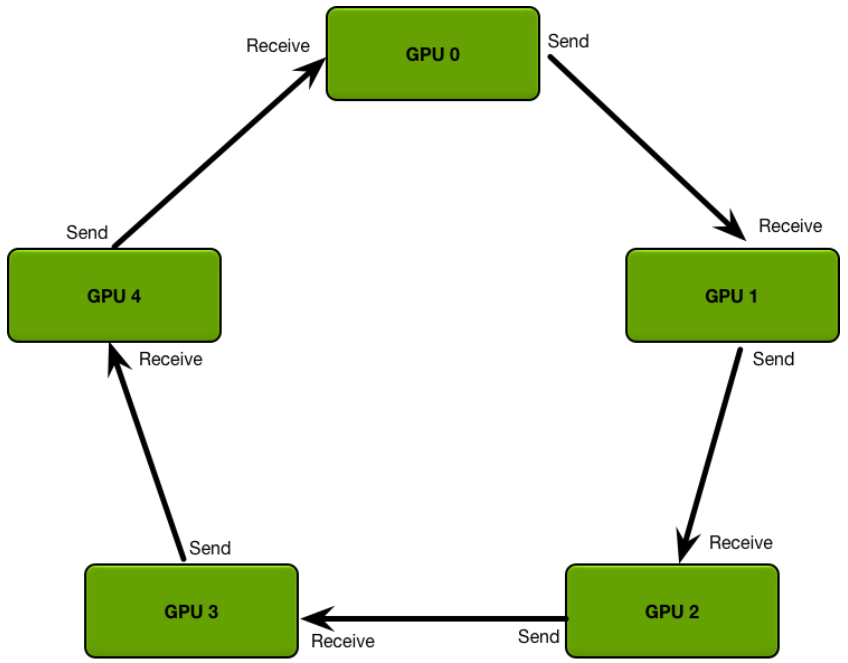
01. Ring 算法

Flat ring

Baidu All-Reduce：节点间分布式训练场景

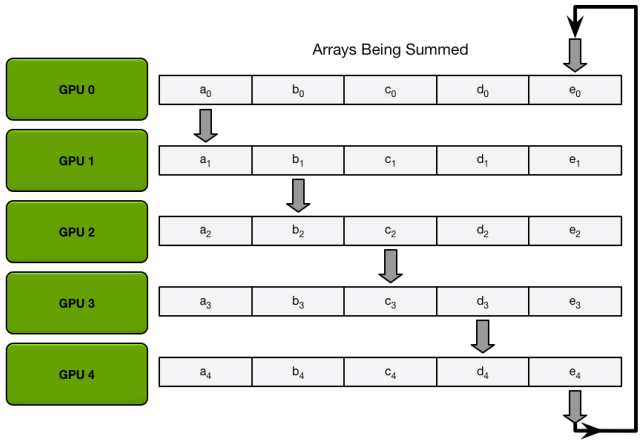
- 使用 Ring 方式 all reduce 算法提高大 Message 传输效率 （2017）
- 使用 MPI-Send 和 MPI receive 点到点通信接口实现 all reduce 操作；
- **效果**：减少GPU间传输时间， OpenMPI 在多 GPU 下效率提升。



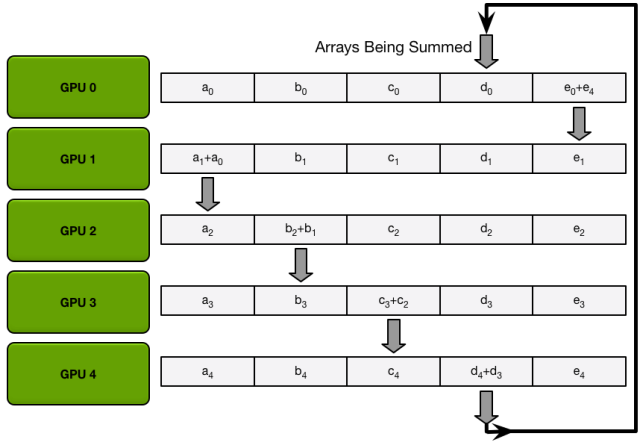


Baidu All-Reduce：节点间分布式训练场景

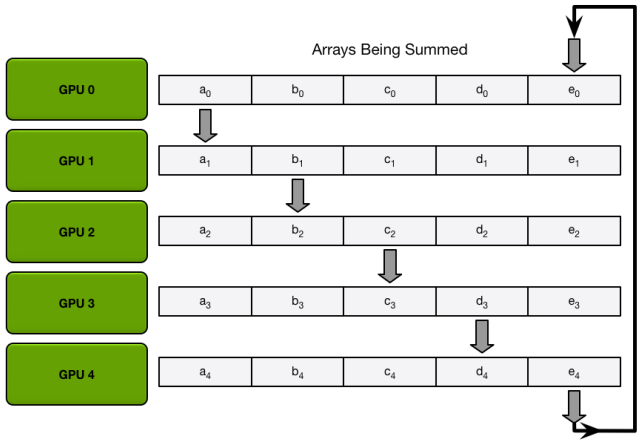
Data transfers in the first iteration of scatter-reduce



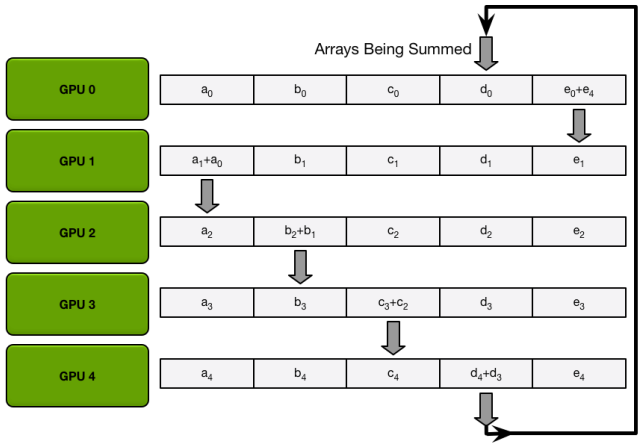
Scatter-reduce data transfers (iteration 1)



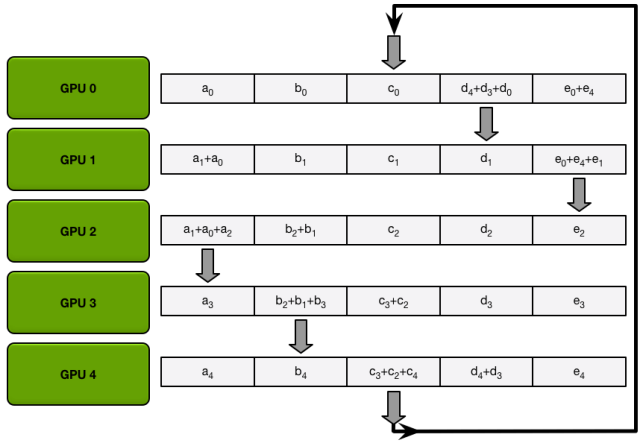
sums after the first iteration of scatter-reduce is complete



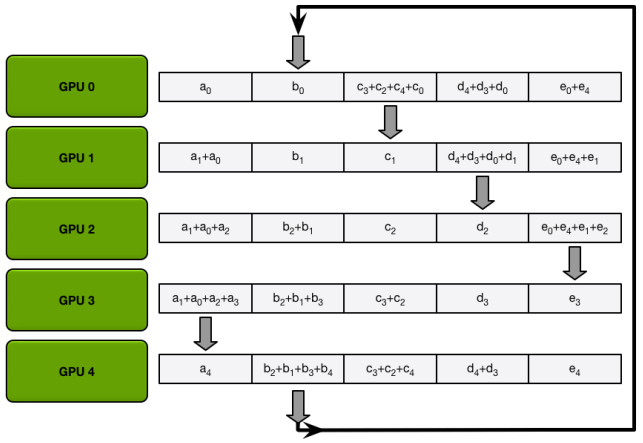
Scatter-reduce data transfers (iteration 2)



Scatter-reduce data transfers (iteration 3)

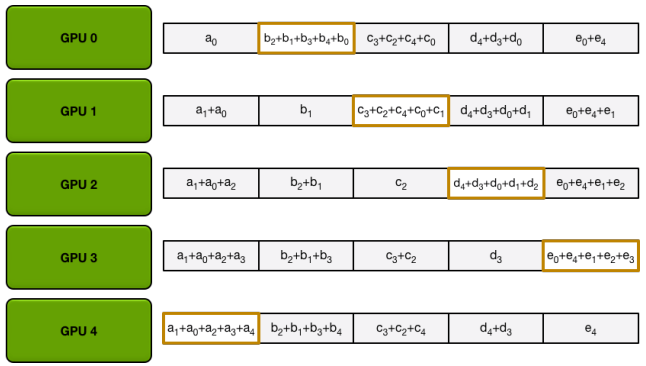


Scatter-reduce data transfers (iteration 4)

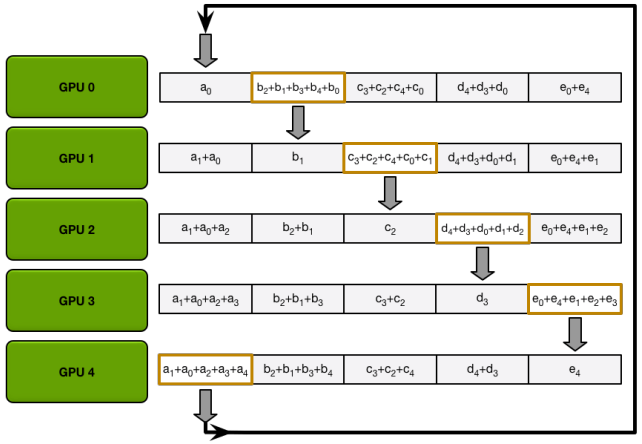


Baidu All-Reduce：节点间分布式训练场景

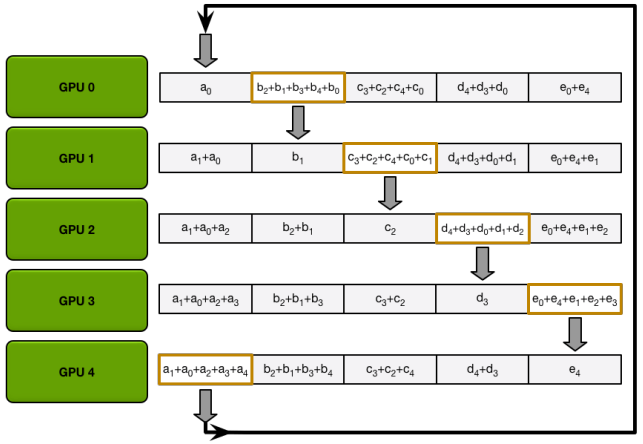
Final state after all scatter-reduce transfers



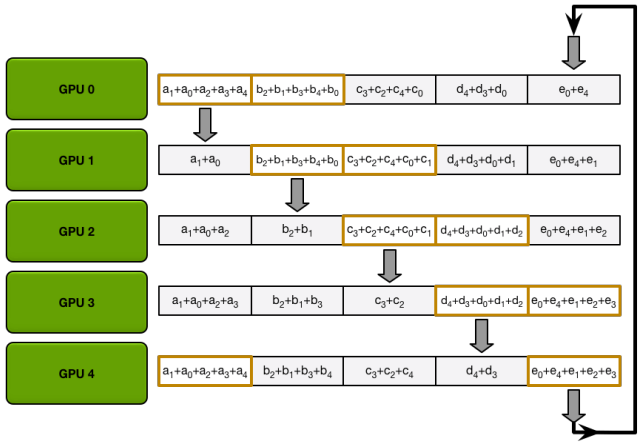
Data transfers in the first iteration of the AllGather



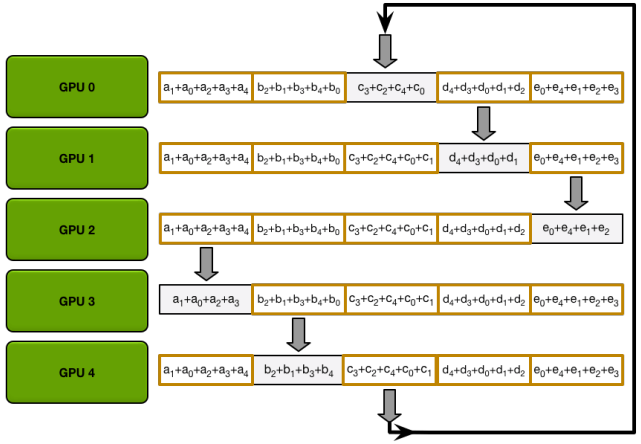
AllGather data transfers (iteration 1)



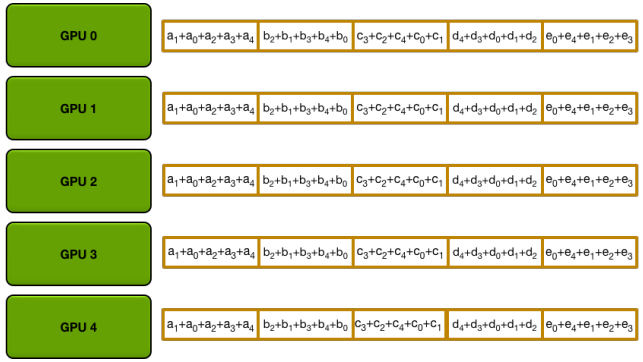
AllGather data transfers (iteration 2)



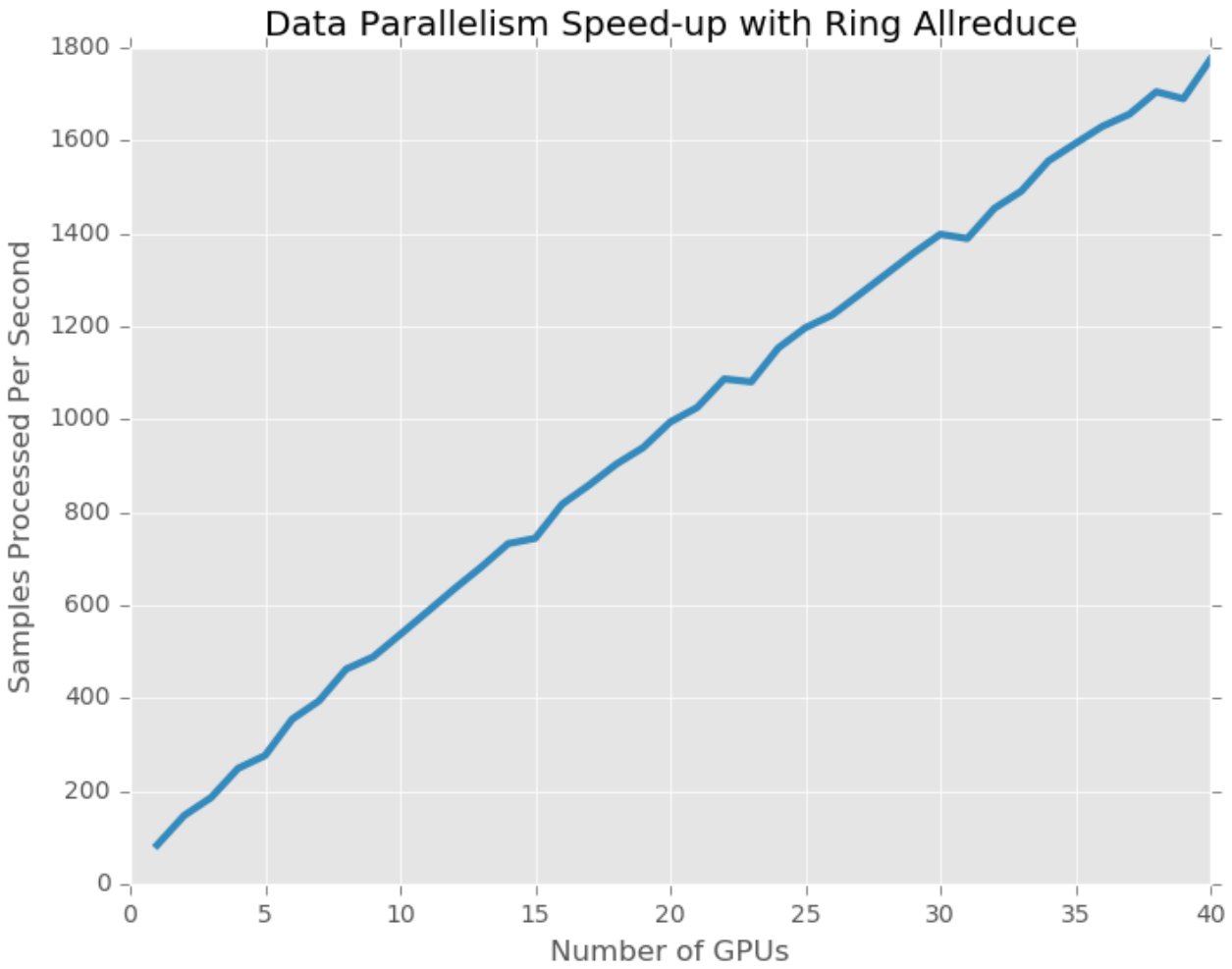
AllGather data transfers (iteration 4)



Final state after all AllGather transfers



Baidu All-Reduce：节点间分布式训练场景



$$Data\ Transferred = 2(N-1)K/N$$

3 亿参数 LLM 训练每秒处理的样本数量与 GPU 数量成线性比例

集合通信原语的含义

https://space.bilibili.com/517221395/channel/collectiondetail?sid=3130927

大模型的集合通信内容介绍 #大模型 #通信 #集合通信

3654 6-2

为什么需要集合通信？NCCL的架构是什么样？ #大模型 #通信 #集合通

3439 6-3

集合通信的操作/原语/算子是什么？ #大模型 #通信 #集合通信

2830 6-5

AI 对集合通信算法的诉求有什么？集合通信算法是啥？ #大模型 #通

2342 6-12

大模型并行的集合通信算法具体实现细节纰漏！#大模型 #集合通信

2156 6-14

通信域是什么概念？PyTorch 如何实现集合通信？#大模型 #集合通信

2383 6-15

研究大模型在 AI 集群的通信，还要了解芯片内互联技术？Yes！#大模

3008 6-30

终于到了大模型集群互联，看昇腾 Atlas 900集群细节！#大模型 #集

3882 7-2

02. 分层环算法

Hierarchical Rings

Question

1. Ring 算法这么香？NV 都引入 NCCL2.X 版本了，为什么还需要介绍 DBT 算法呢？

Ring 算法不足

- **基本介绍：**最常见实现算法基于 Ring All Reduce，NVIDIA NCCLv1.X通信库采用该算法，每次跟相邻的两个节点进行通信，每次通信数据总量的 1/N。

- **适用拓扑：**Star、Tree 等小规模集群；**通信步骤：**$2\times(N-1)$ Step；

- **优点：**实现简单，能充分利用每个节点的上行和下行带宽；

- **缺点：**通信延迟随着节点数线性增加，特别是对于小包延迟增加比较明显；Ring太大，Ring All Reduce 效率也会变得很低。

引入 Tree 算法

- All reduce 操作用于对多个 GPU 上的梯度求和，通常使用环 Ring 来实现全带宽。环的随着训练任务中使用的 GPU 个数增加，ring all-reduce 延迟会线性增长；

- 开始出现大规模分布式并行实验，采用分层的二维环算法（Hierarchical Rings）取代了扁平环（Flat Ring），以获得更好的带宽，同时降低延迟。

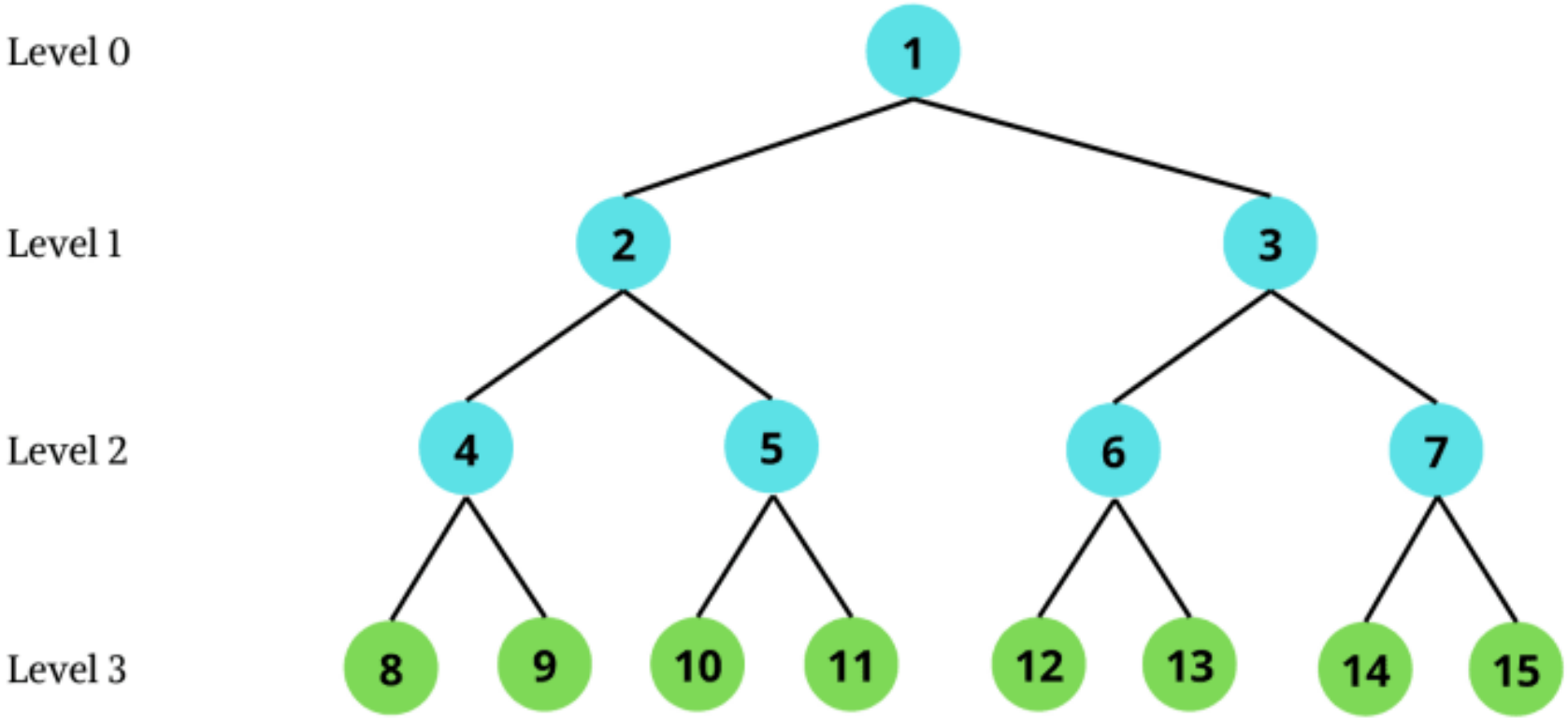
- 分层环仍然为了解决这个问题，NCCL 2.4 版本引入了 tree 算法，即 double binary tree；2.4 版本后，NCCL 同时具备 Tree 和 Ring 算法；

03. 朴素二叉树

Native binary Tree

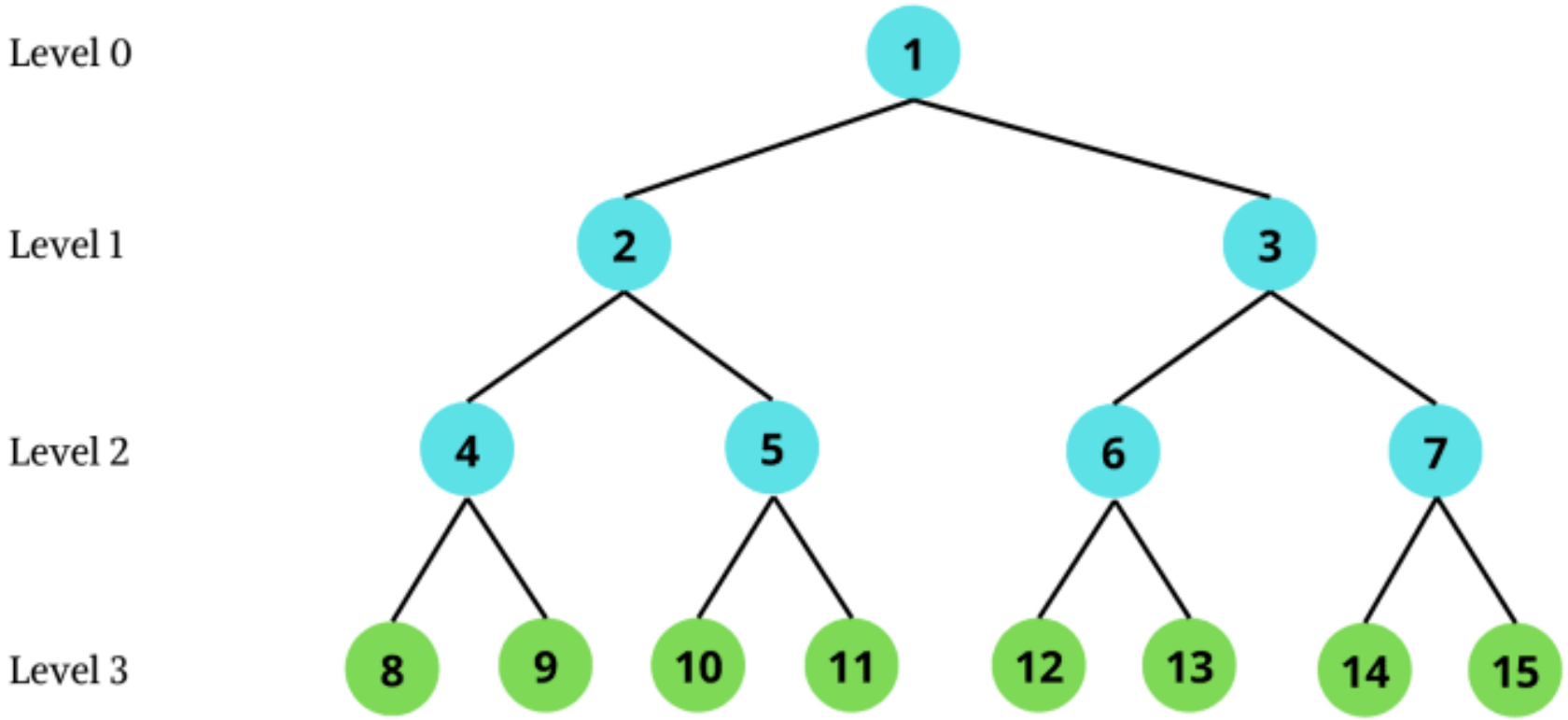
朴素二叉树

- 朴素二叉树（Native binary Tree）算法将所有 NPU 节点构造成一棵二叉树，支持 broadcast，reduce，前缀和。



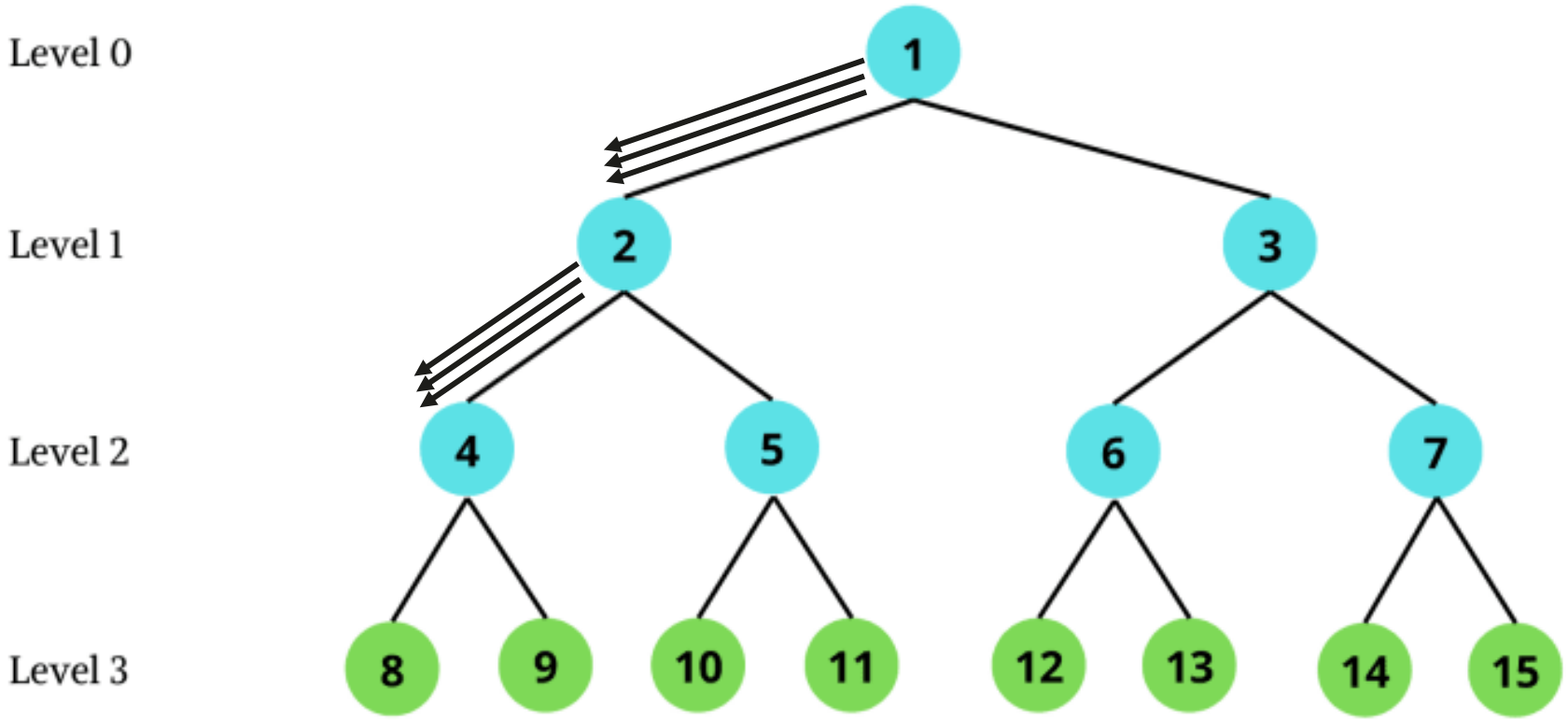
朴素二叉树

- 假设 root 节点要 broadcast 一个消息 M 给所有 NPU 节点，root 会将 M 发送给他的子节点，其他所有节点收到消息 M 后再发送给子节点，叶节点因为没有子节点，所以叶结点只会接收 M。



朴素二叉树怎么做并行?

- 将 M 切分为 k 个 block，从而充分利用网络带宽（Stream 数），进行可以流水线并行起来。



Question

1. 朴素二叉树的问题在哪里呢？叶节点只接收 & 不发送数据，因此只利用了带宽的一半。

04. 双二叉树

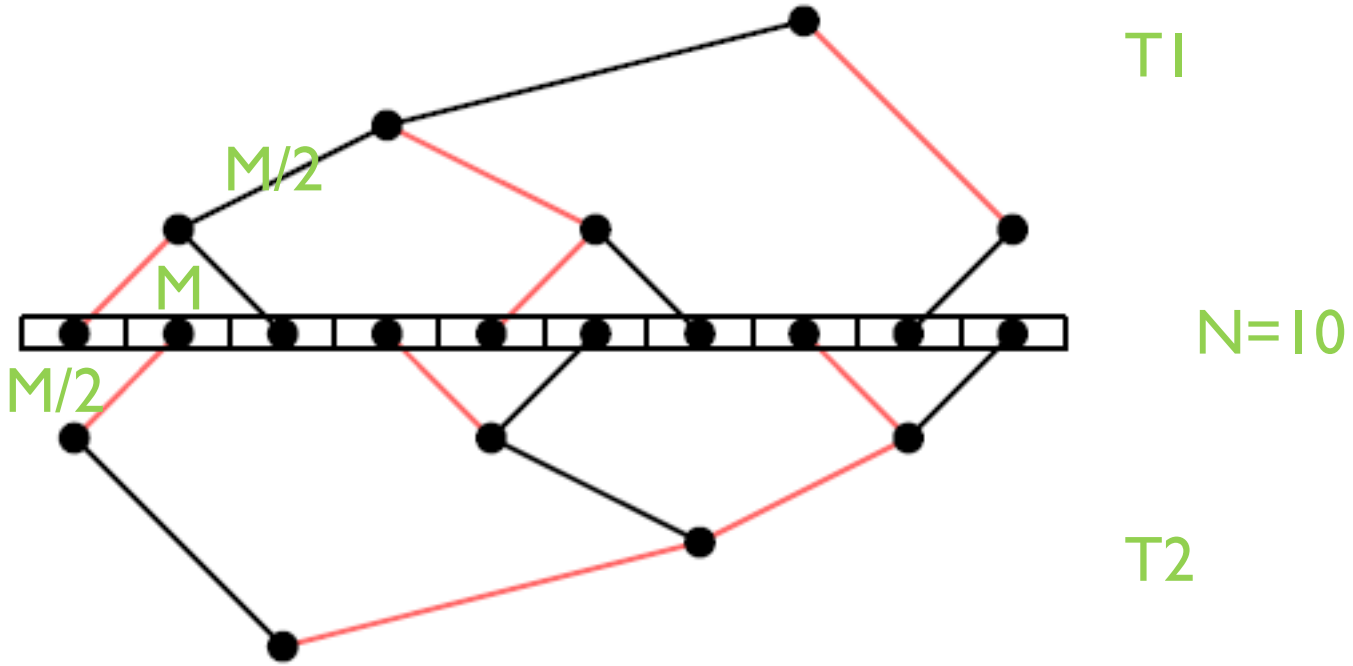
double Binary Tree

MPI 引入 Double Binary Tree 算法

- 2009 年 MPI 引入 double binary tree 算法，结合算法中广播和归约操作的全带宽（可以组合成一个 all reduce，先执行归约，然后执行广播）和对数延迟的优势，使得在小型 or 中型 CPU 集群的集合通信操作上性能更好。

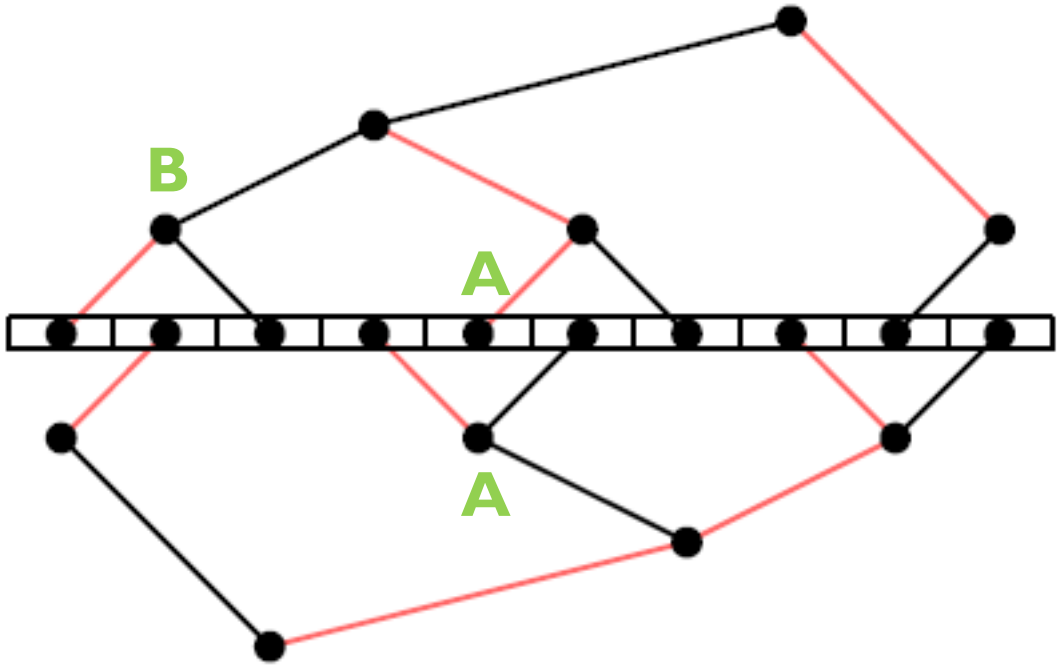
Double Binary Tree 算法介绍

- 假设一共有 N 个设备，MPI 构建两颗大小为 N 树 T1 和 T2，T1 中间节点在 T2 为叶节点，T1 和 T2 同时通信，各自负责消息 M 的一半，这样每个节点的双向带宽可以都被利用到。
- 以 N = 10 为例，构建出的结构如下：



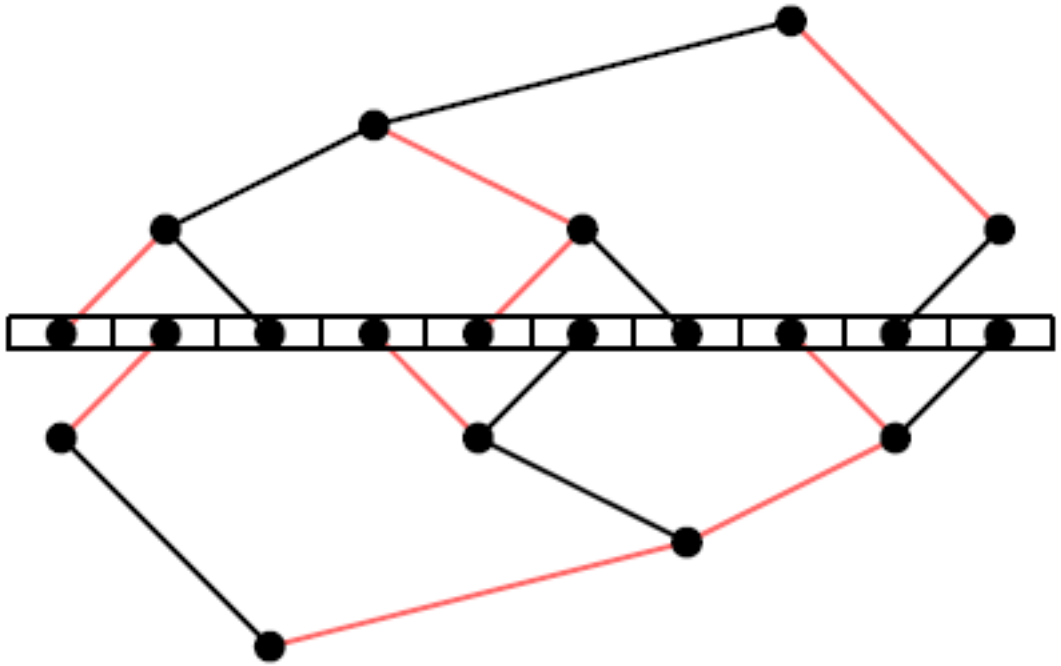
Double Binary Tree 算法特点

- 不会有节点在 T1 和 T2 中连到父节点边的颜色相同，如 node A 在 T1 中通过红色的边连到父节点，那 T2中 A 一定通过黑色的边连到父节点。
- 不会有节点连到子节点的边颜色相同，如 node B 在 T1 中是中间节点，如果 B 通过红色的边连到左子节点，那么 B 一定通过黑色的边连到右子节点。



Double Binary Tree 算法收益

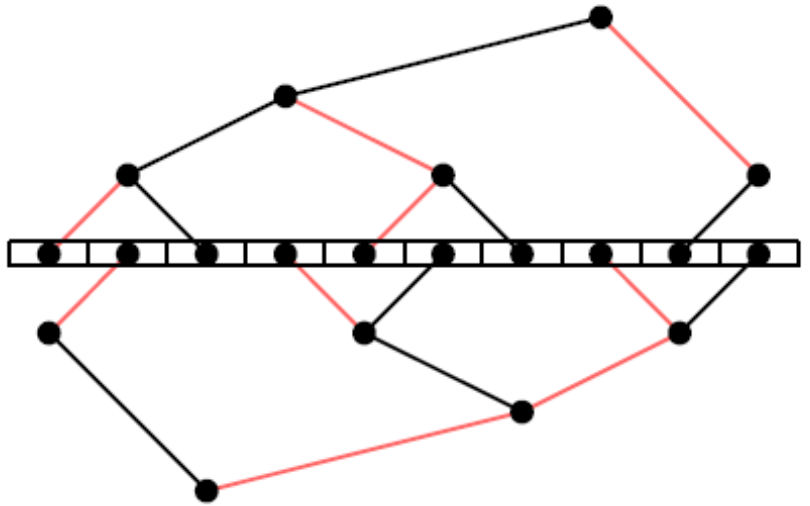
- 两棵树同时工作，在每一步中从父节点中收数据，并将上一步中收到的数据发送给子节点，如在偶数步骤中使用红色边，奇数步骤中使用黑色边，一个步骤中可以同时收发，从而利用了双向带宽。



Double Binary Tree 算法介绍

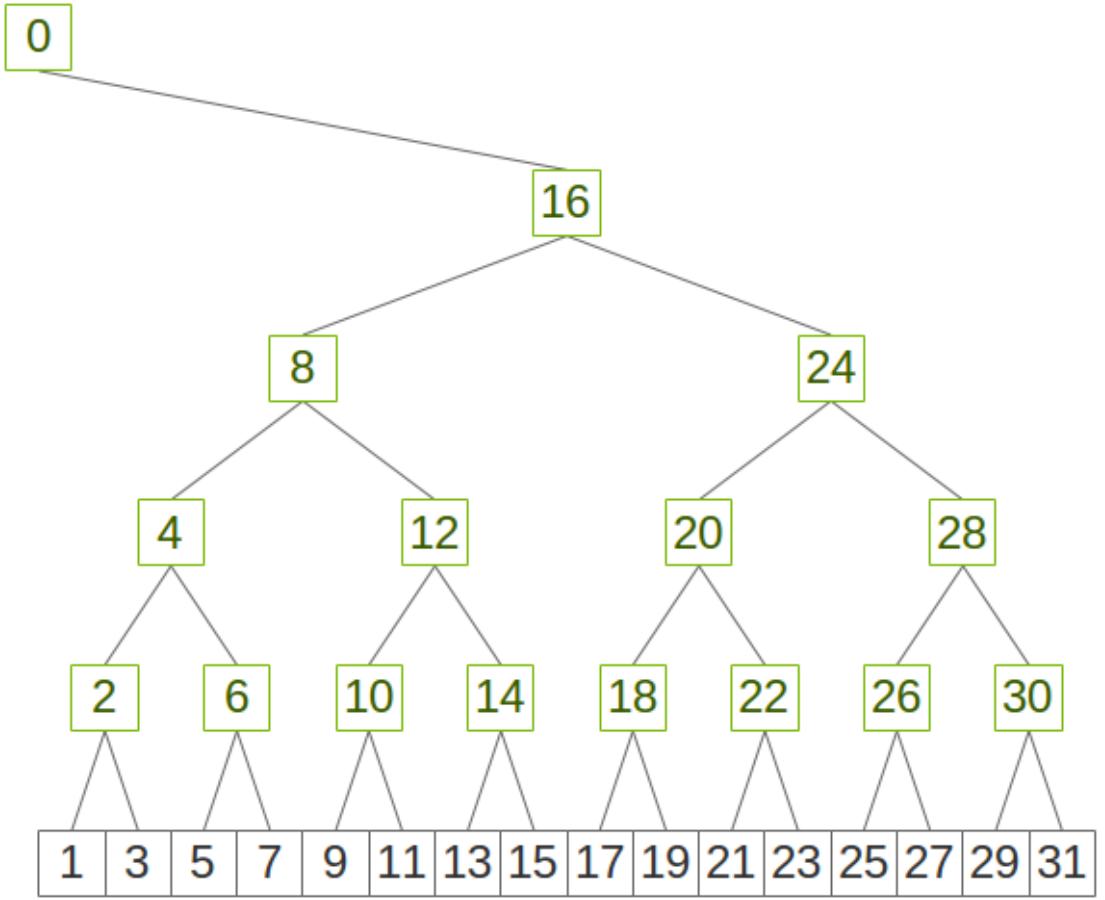
- T2 构树两种方式：

  1. shift 移位，将 rank 向左 shift 一位，如 rank10 变成 rank9，构树与 T1 保持一致，使得 T1 和 T2 树结构完全一致；
  2. mirror，将 rank 镜像，如 rank0 镜像为 rank9，使得 T1 和 T2 树结构是镜像对称，不过 mirror 方式只能用于设备（NPU/GPU）数为偶数的场景，否则会存在节点在两棵树中都是叶节点。



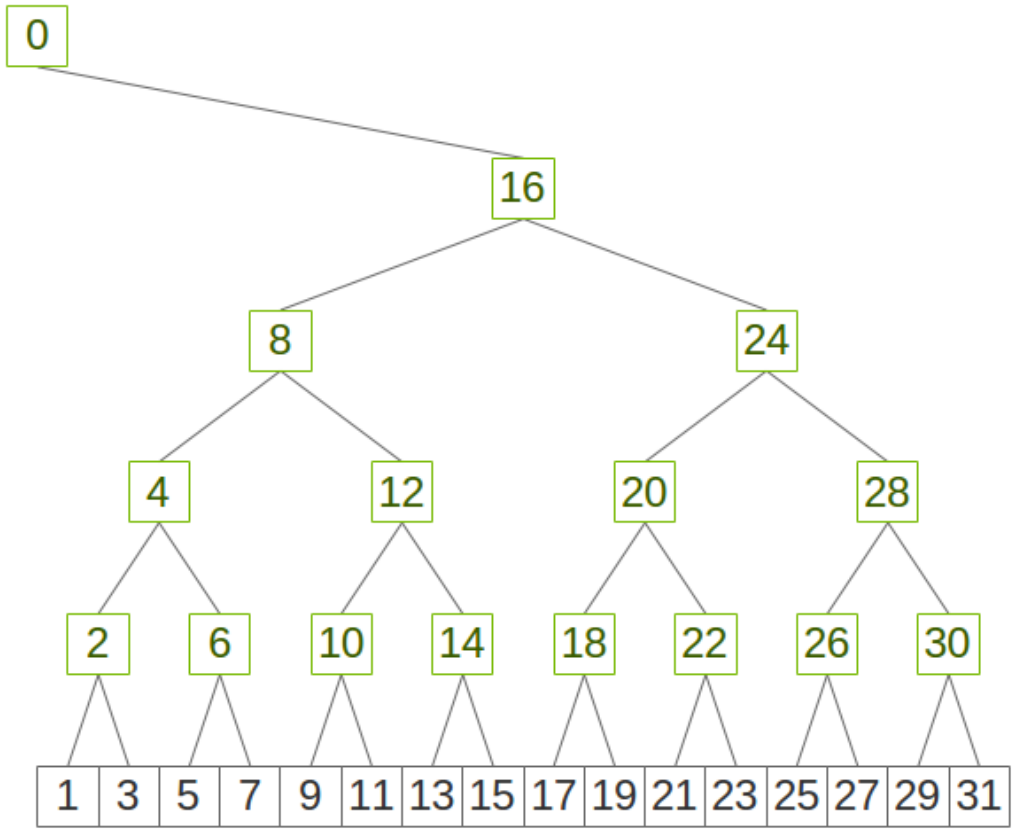
英伟达 Double Binary Tree 算法

- NCCL 中，使用最大化局部性的模式来构建二叉树：



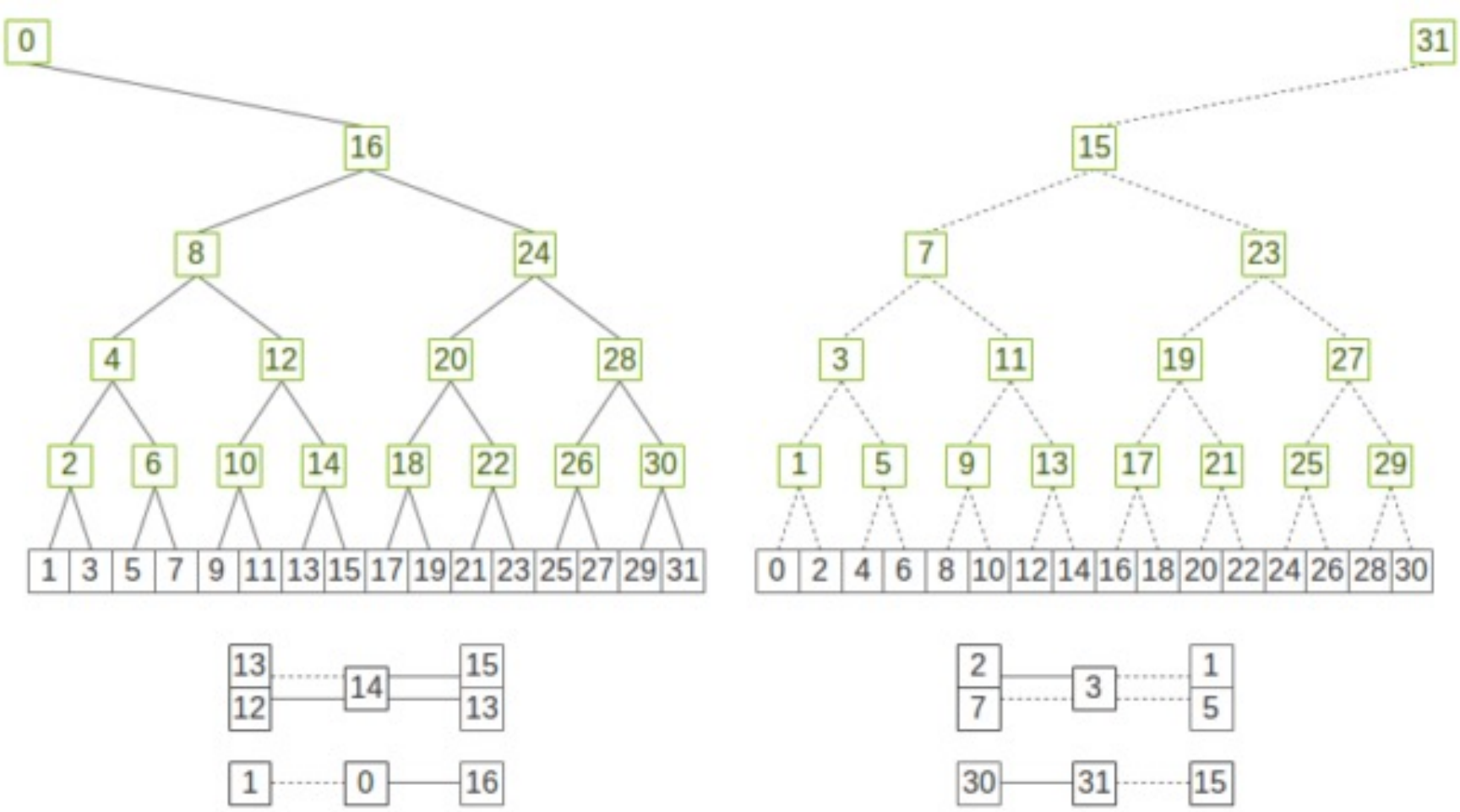
英伟达 Double Binary Tree 算法

- 二叉树中一半或更少的等级是节点，一半（或更多）的等级是叶子。为每个二叉树构建第二棵树，使用叶子作为节点，反之亦然。可能有一个等级在两棵树上都是叶子，但没有等级在两棵树上都是节点。



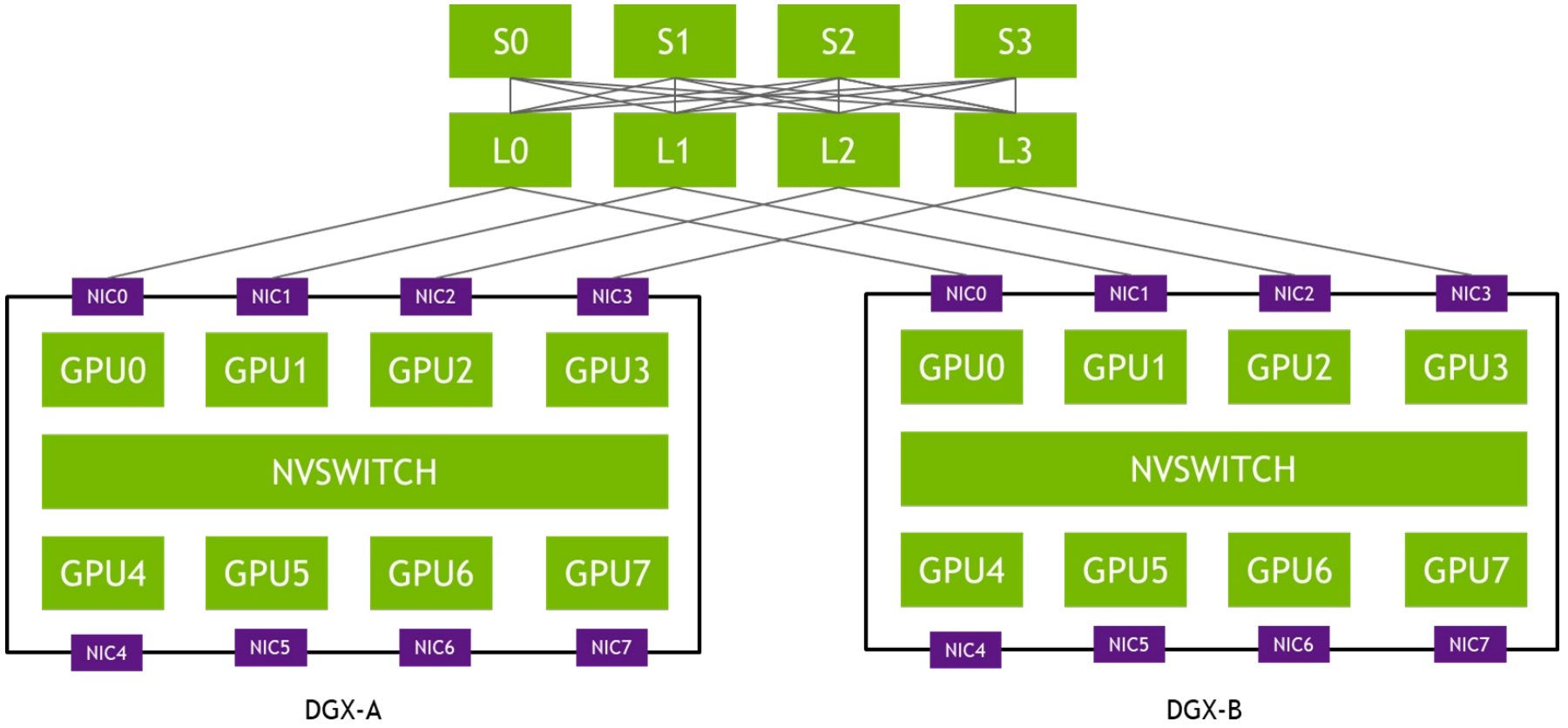
英伟达 Double Binary Tree 算法

- 通过翻转树来反转节点和叶子，从而构建双二叉树。NCCL 中 tree 只用于节点之间，节点内是一条链。



英伟达 Double Binary Tree 算法

- 通过翻转树来反转节点和叶子，从而构建双二叉树。NCCL 中 tree 只用于节点之间，节点内是一条链。



英伟达 Double Binary Tree 算法效果

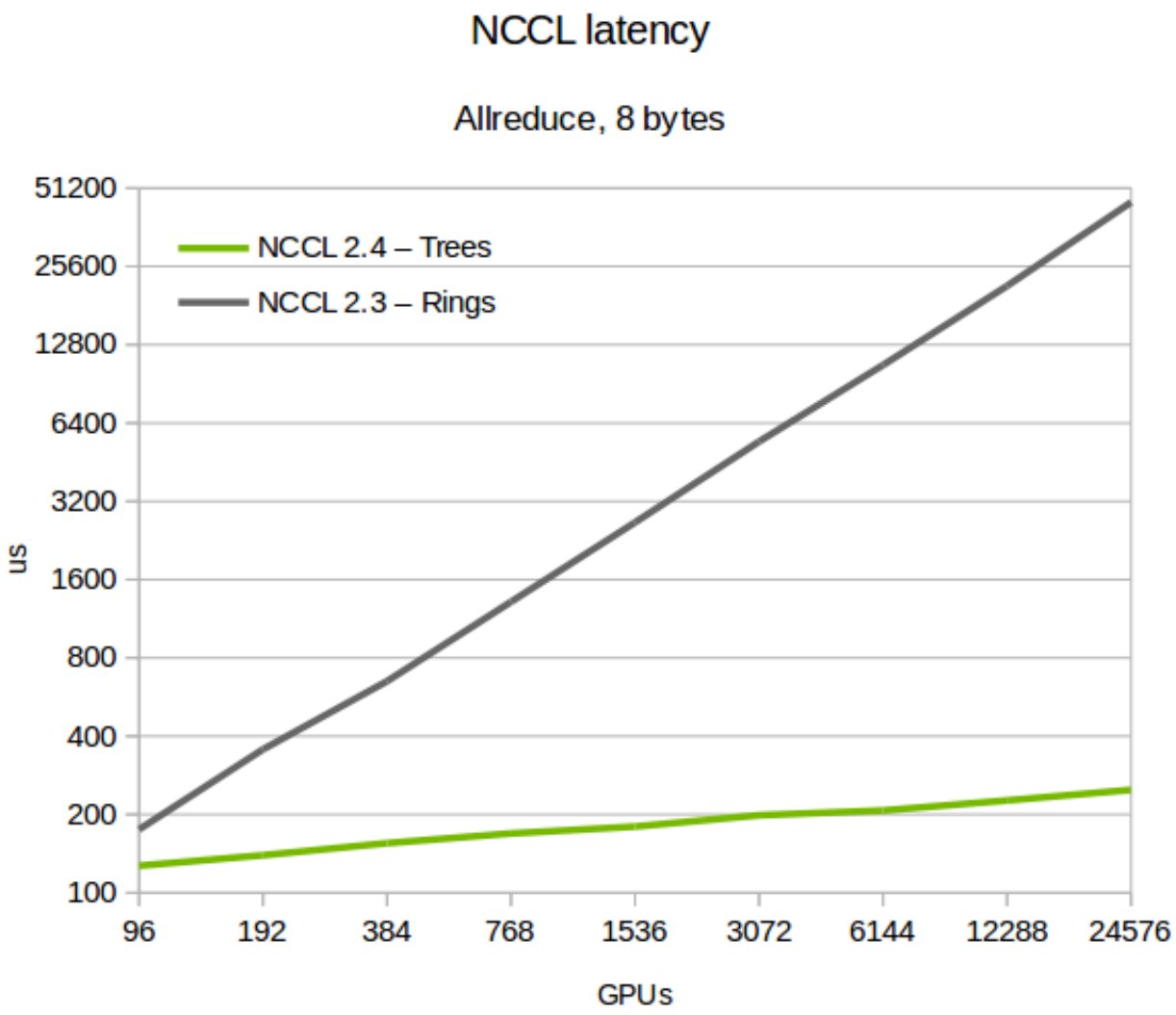
- 如果将这两棵树叠加起来，除了根等级只有一个父级和一个子级外，所有等级都有两个父级和两个子级。如果使用这两棵树中的每一棵树来处理一半的数据，则每个等级最多会接收一半的数据两次，发送一半的数据两次，在发送/接收数据方面与环一样优化。

05. 效果 &

环和树的选择

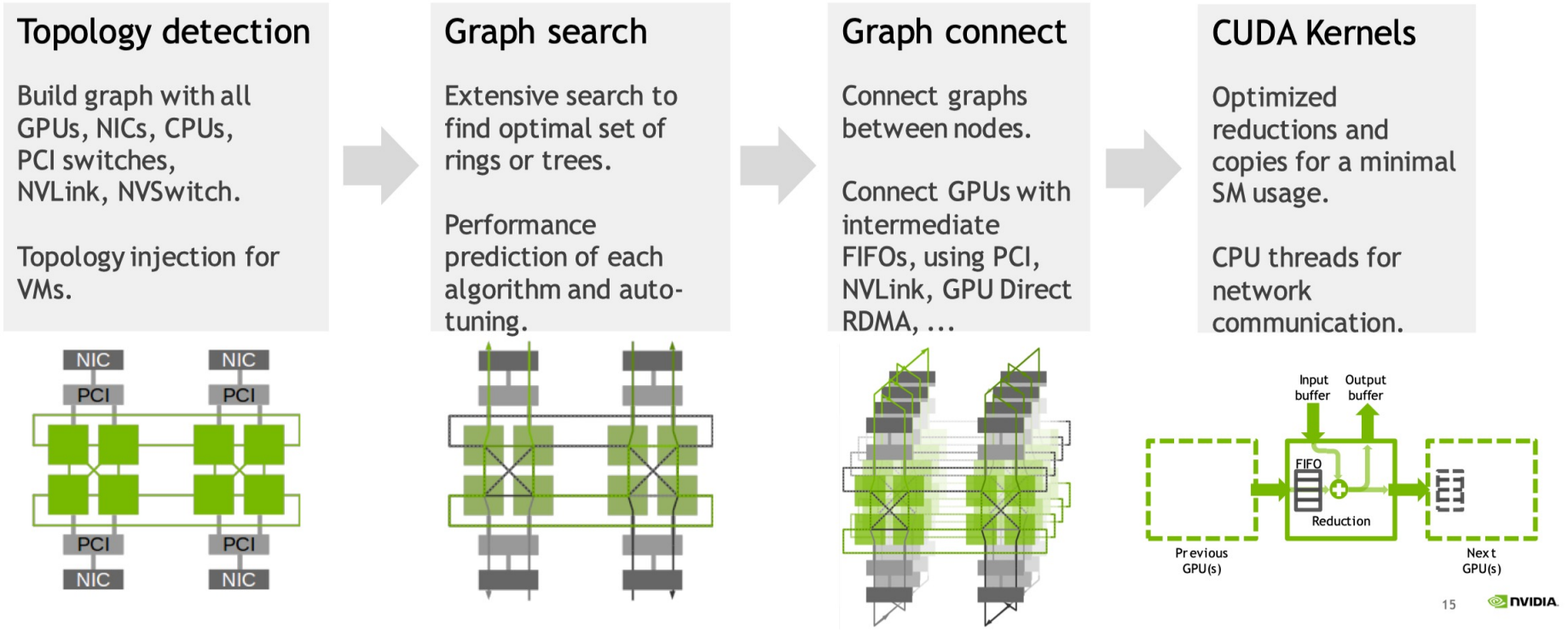
节点规模越大，时延衰减不明显

- 24,576 个 GPU。Trees 显著改善延迟。与 Ring 差异随着规模的扩大而增大，在 24k GPU 上最高可提高 180 倍。
- 使用 DBT 可保持全带宽。在规模上，当我们在 InfiniBand 结构中跨越 L3 交换机时，带宽会略有下降，可能是由于 NCCL 通信模式和 InfiniBand 路由算法之间的效率低下造成。



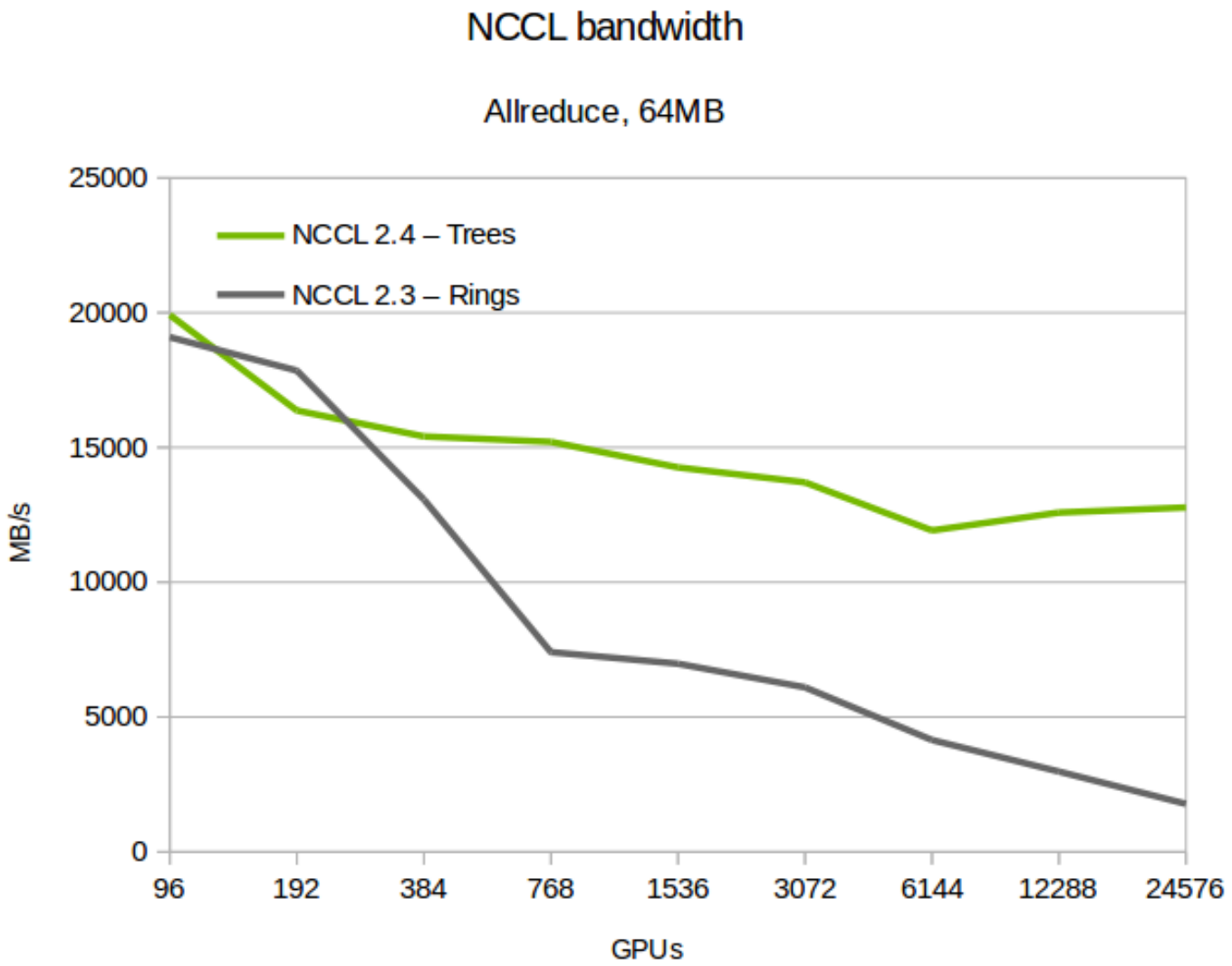
NCCL 基本介绍

- NVIDIA 集合通信库 (NCCL) 是一个类 MPI 通信库，可实现 GPU 的集合通信算法相关操作：
- all-gather / all-reduce / broadcast / reduce / reduce-scatter / point-to-point send and receive



节点规模越大，时延衰减不明显

- 由于树的初始延迟较小，即使在带宽有限的情况下，树仍然显示出明显的优势。但是，当该模式导致带宽更大时，NCCL 会自动切换回环。



ring和tree的选择

- 用户传入 $n\ Bytes$ 长度数据，总耗时：

$$time = latency + n\ Bytes\ /\ algo\_bw$$

- $algo\_bw$ 为算法带宽，基础总线带宽为bus_bw（每个 channel 带宽乘channel数），然后根据实测的数据对带宽进行一些修正，如 Tree 乘 0.9。
- 计算算法带宽，Tree 时除以 2，上行一次，下行一次相当于发送两倍数据量，Ring 除以 $2x(nranks\ -\ 1)\ /\ nranks$。
- NCCL 最后将计算出的每种协议和算法的带宽延迟保存到bandwidths和latencies。当用户执行allreduce api的时候会通过getAlgoInfo计算出每种算法和协议组合的执行时间，选出最优的。

Ring 和 Tree 的选择

- 用户传入 $n\ Bytes$ 长度数据，总耗时：

$$time = latency + n\ Bytes\ /\ algo\_bw$$

- NCCL 将计算出的每种协议和算法，在集群规模下的带宽延迟保存到 bandwidths 和 latencies。当开发者执行 allreduce API 时通过 getAlgoInfo 计算出每种算法和协议组合的执行时间，选出最优方案。

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.

ZOMI

Course chenzomi12.github.io

GitHub github.com/chenzomi12/AIFoundation

参考资料

1. https://blog.csdn.net/KIDGIN7439/article/details/135102930
2. https://developer.nvidia.com/blog/massively-scale-deep-learning-training-nccl-2-4/
3. Baidu Allreduce https://github.com/baidu-research/baidu-allreduce
4. Horovod https://github.com/uber/horovod
5. Xianyan Jia, Shutao Song, Wei He, Yangzihao Wang, Haidong Rong, Feihu Zhou, Liqiang Xie, Zhenyu Guo, Yuanzhou Yang, L iwei Yu, Tiegang Chen, Guangxiao Hu, Shaohuai Shi, Xiaowen Chu; Highly Scalable Deep Learning Training System with Mixe d-Precision: Training ImageNet in Four Minutes
6. Hiroaki Mikami, Hisahiro Suganuma, Pongsakorn U-chupala, Yoshiki Tanaka, Yuichi Kageyama; ImageNet/ResNet-50 Training in 224 Seconds
7. Chris Ying, Sameer Kumar, Dehao Chen, Tao Wang, Youlong Cheng; Image Classification at Supercomputer Scale
8. Peter Sanders; Jochen Speck, Jesper Larsson Träff (2009); Two-tree algorithms for full bandwidth broadcast, reduction and s can
9. Summit Supercomputer https://www.olcf.ornl.gov/summit/
10. Two-Tree Algorithms for Full Bandwidth Broadcast, Reduction and Scan