Session6: AI Infrastructure

- AutoCCL: Automated Collective Communication Tuning for Accelerating Distributed and Parallel DNN Training.(NSDI25)
  通过在线调优方式，对NCCL通信库参数选择进行优化提升训练速度
- White-Boxing RDMA with Packet-Granular Software Control.（NSDI25）
  通过仿照SDN设计了软件控制的RNIC，实现了数据包粒度的软件控制
- Unlocking ECMP Programmability for Precise Traffic Control.(NSDI25)
  通过ECMP组解锁ECMP可编程性，实现ECMP与精确流量控制并存。
- WLB-LLM: Workload-Balanced 4D Parallelism for Large Language Model Training.(OSDI25)
  通过平衡不同GPU的工作负载提升LLM的训练速度

AutoCCL: Automated Collective Communication Tuning for Accelerating Distributed and Parallel DNN Training

NSDI'25

Guanbin Xu[1], Zhihao Le[1], Yinhe Chen[1], Zhiqi Lin[1], Zewen Jin[1]

Youshan Miao[2], Cheng Li[1 3]

University of Science and Technology of China[1], Microsoft Research[2]

Hefei Comprehensive National Science Center[3]

1

2

3

Presenter: Yujie Chen

2025.9.20

Author

研究方向

- 并行与分布式智能计算系统
- 以数据为中心的系统架构研究
- 面向大模型的数据全周期管理与优化

近期发表论文：

- BigMac: A Light All-to-All Mixture-of-Experts Model Structure for Fast Training and Inference. (AAAI 2025)
- Eliminating Data Processing Bottlenecks in GNN Training over Large Graphs via Two-level Feature Compression. (VLDB 2024)
- Noctua: Towards Automated and Practical Fine-grained Consistency Analysis.(EuroSys 2024)
- nnScaler: Constraint-Guided Parallelization Plan Generation for Deep Learning Training. (OSDI 2024)

Cheng Li (李诚)

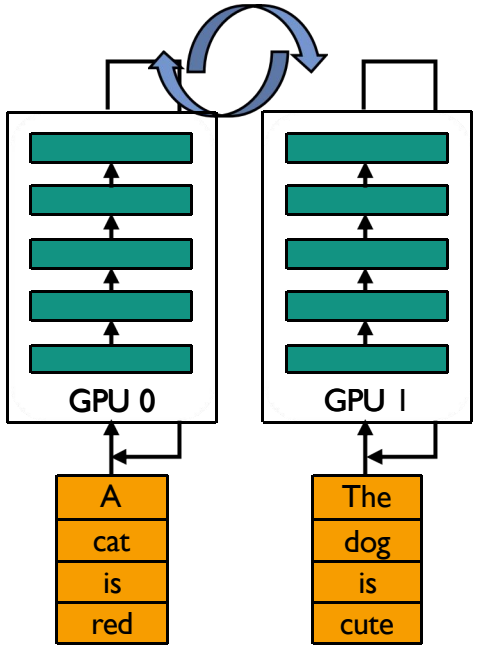
https://mr-cheng-li.github.io/

Contents

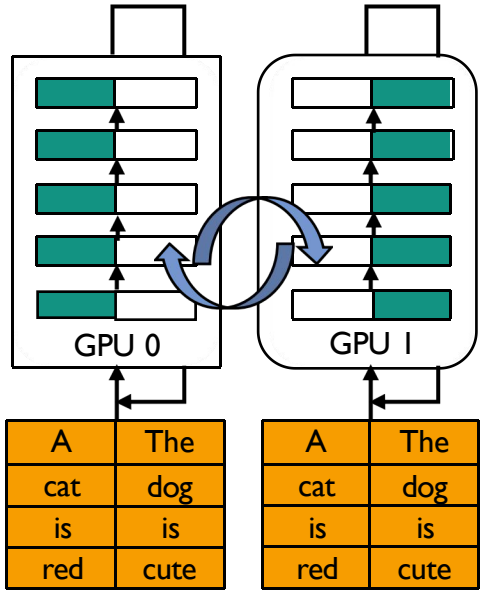
- Background
- Design
- Evaluation
- Thinking

Background

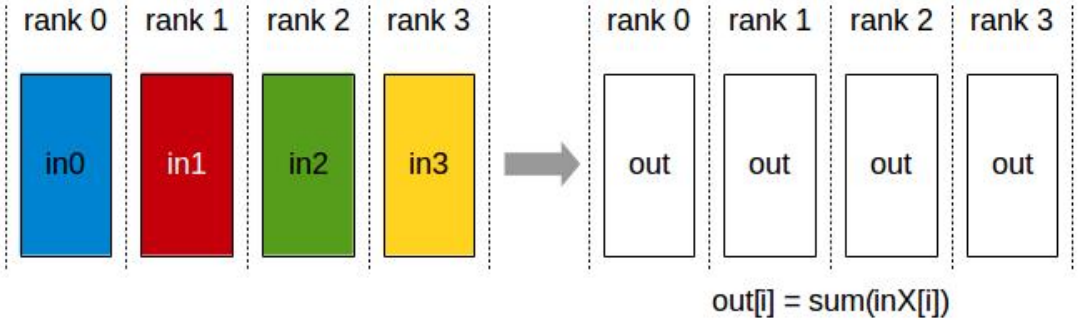
Distributed DNN Training



Data Parallelism (DP)



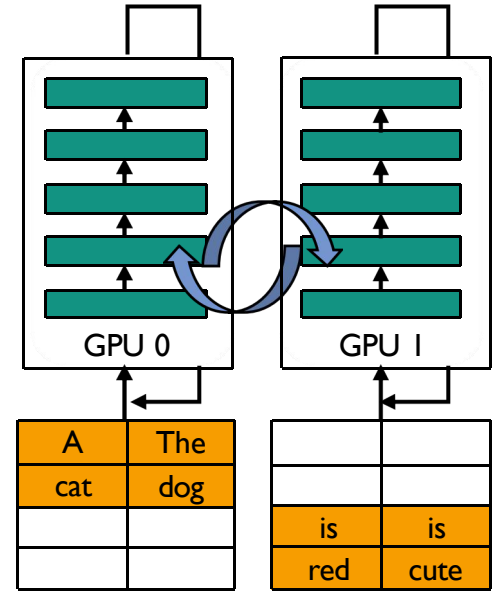
Tensor Parallelism (TP)



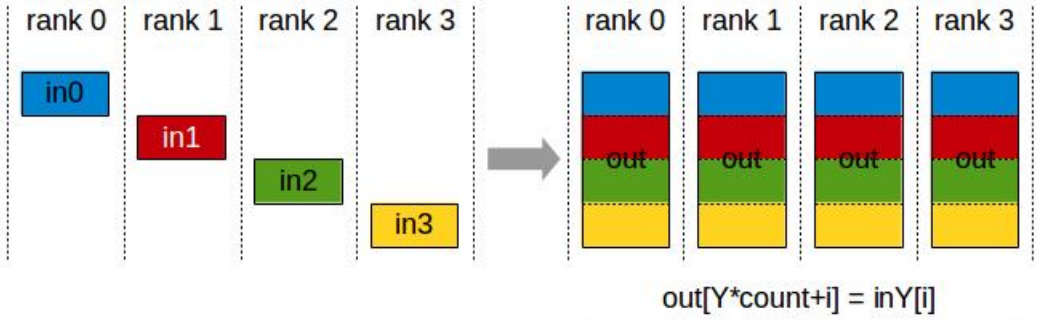
AllReduce: Tree-, Ring-, …

Background

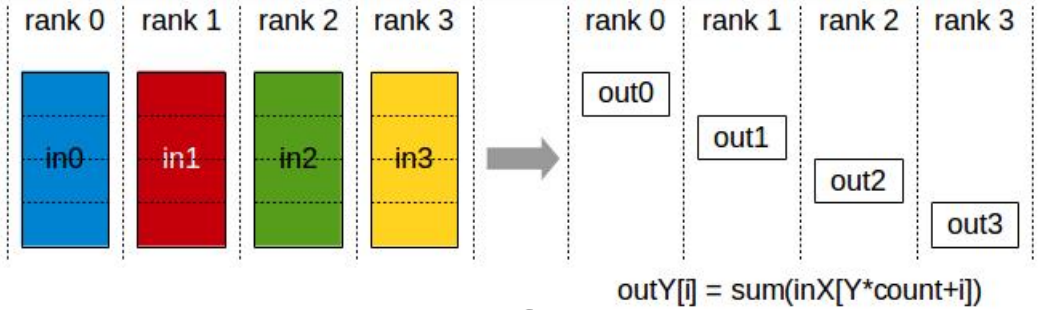
Distributed DNN Training



Sequence Parallelism (SP)



AllGather



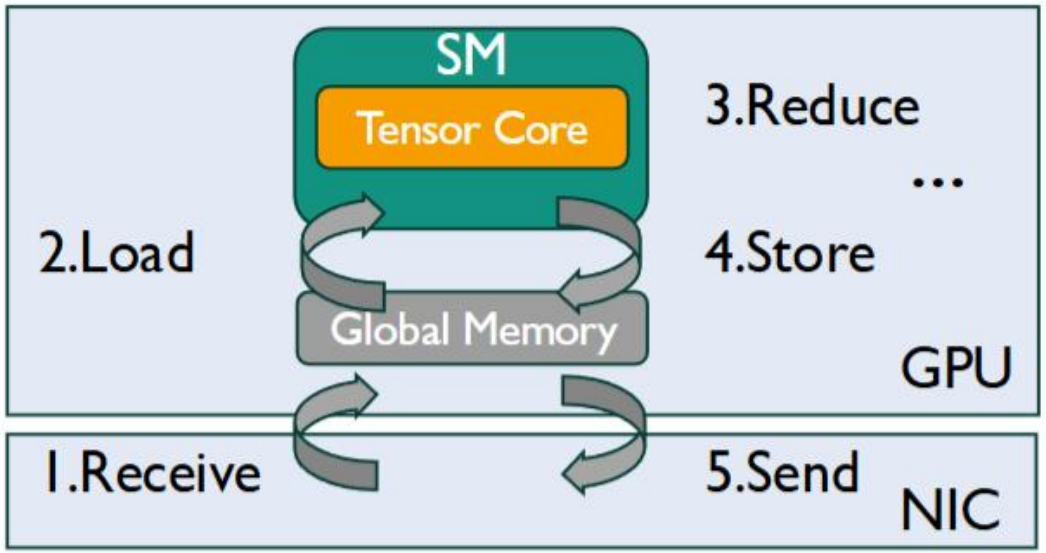
ReduceScatter

多个GPU之间的通信是分布式和并行DNN训练的关键

Background

集合通信

- 在多个GPU之间高效地同步模型数据（如梯度或参数），以确保所有并行计算单元的模型状态保持一致，从而共同完成训练任务。

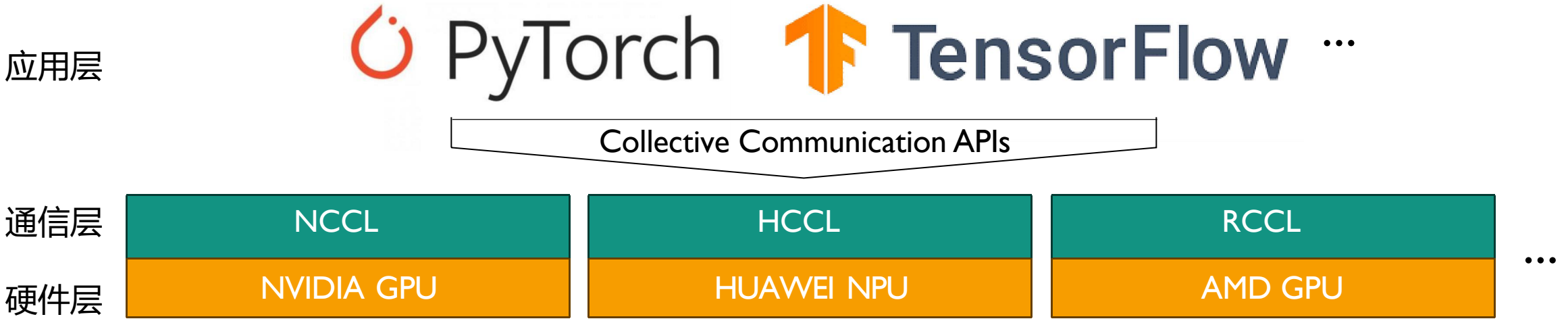


单次集合通信的数据流：

从网络接口卡（NIC）接收->加载到GPU内存->计算核心（SM）处理->存储结果->同步结果发送回网络

Background

集合通信



SOTA Models	Training Cost (USD)
GPT3	1.13 million
OPT-175B	1.65 million
Megatron-Turing NLG 530B	3.04 million

尽管NCCL高度优化，但通信问题依然是分布式训练中的性能和成本瓶颈（占据42%训练时长[1]）

[1] Wang S, et al. Overlap communication with dependent computation via decomposition in large deep learning models ASPLOS22

Background

通信优化方法

- 计算与通信重叠：通过让计算和通信并行执行，从而隐藏通信延迟 。

Horovod 将 All-Reduce 通信与反向计算重叠以提升性能。

Horovod: fast and easy distributed deep learning in tensorflow. (arXiv'18)

- 通信数据压缩：通过量化或稀疏化技术在不严重影响模型精度的情况下，减少网络传输数据。

HiPress 引入了一个框架来高效压缩 All-Reduce 中的梯度数据。

Gradient compression supercharged high-performance data parallel dnn training. (SOSP'21)

不足：上述方法通过引入新的算法实现，缺乏对主流的集体通信库的全面分析。

- 通信配置调优：使用性能模型或机器学习方法来预测通信效率并选择配置 。

AFNFA 使用机器学习和离线分析来改进对通信性能的预测。

Afnfa: An approach to automate nccl configuration exploration. (APNet'23)

不足：基于离线预测分析无法适应**动态变化**的复杂通信场景

Background-Motivation

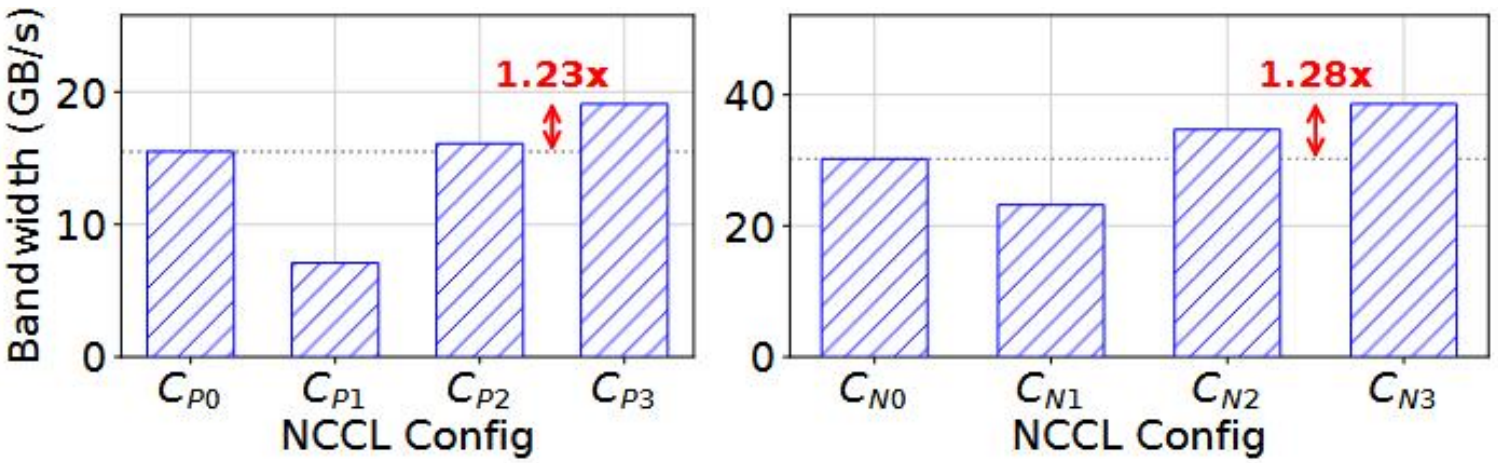
底层的通信库（NCCL）本身是否已经被充分调优?

通信任务 AllGather：一台机器中的8张A40 GPU之间聚合80MB的数据。

性能差距显著:

- 在 PCIe 环境下，最优配置(C_P3)的带宽是NCCL默认配置(C_P0)的 1.23倍 。
- 在高度优化的 NVLink 环境中，最优配置(C_N3)的带宽是默认配置(C_N0)的1.28倍.

尽管NCCL被认为是高度优化的库，但其基于内置成本模型选择的默认配置往往是次优的，其底层参数存在巨大的、被忽视的调优潜力。



(a) PCIe (b) NVLink

Background-Challenge

巨大的参数搜索空间：

- NCCL的配置参数组合可达百万级别，遍历这样的空间并比较不同配置的延迟和带宽非常耗时。

最优配置不具备可移植性：

- 针对特定任务（如64MB AllReduce）的最优配置，在应用于另一相似任务（15MB AllReduce）时性能会显著下降，且往往不是最优配置项 。

并行计算的干扰：

- 在真实的DNN训练中，通信任务和计算任务并发执行，争抢GPU资源，将导致各自性能下降，且计算对通信的干扰难以提前预测和建模。

Task	Config	A	NC	NT	C	Bwd (GB/s
64 MB, 16	C_0^*	Ring	2	256	512KB	4.0
64 MB, 16	C_1	Tree	2	256	512KB	5.4
64 MB, 16	C_2	Tree	8	160	59 KB	8.9
15 MB, 8	C_3	Tree	8	160	59 KB	8.1
15 MB, 8	C_4	Ring	10	128	27 KB	8.8

(a) 8-GPU 节点上执行 AllReduce

Interference level	NCCL Bwd	Tuned Bwd
Communication-only	30.08 GB/s	38.62 GB/s
Light computation	26.21 GB/s	35.14 GB/s
Heavy computation	18.26 GB/s	32.44 GB/s

(b) 8-GPU节点上执行AllGather的带宽

为了充分释放通信性能，必须采用一种能够自动、动态地适应不同运行环境的调优方法。

Design-Build Tuning Space

Original Parameters		Abstracted Parameters	
# of parameters	# of key parameters	Categories	# of Choices
158	28	1 for Algorithm (A)	2
		3 for Protocol (P)	3
		3 for Transport (T)	2
		11 for Nchannel (NC)	128
		3 for Nthread (NT)	20
		7 for Chunk size (C)	8192

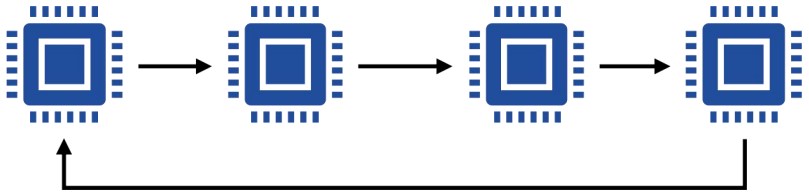
2×3×2×128×20×8192 > 1 millions

如何寻找最优解?

Design-Build Tuning Space

划分为可建模的子空间

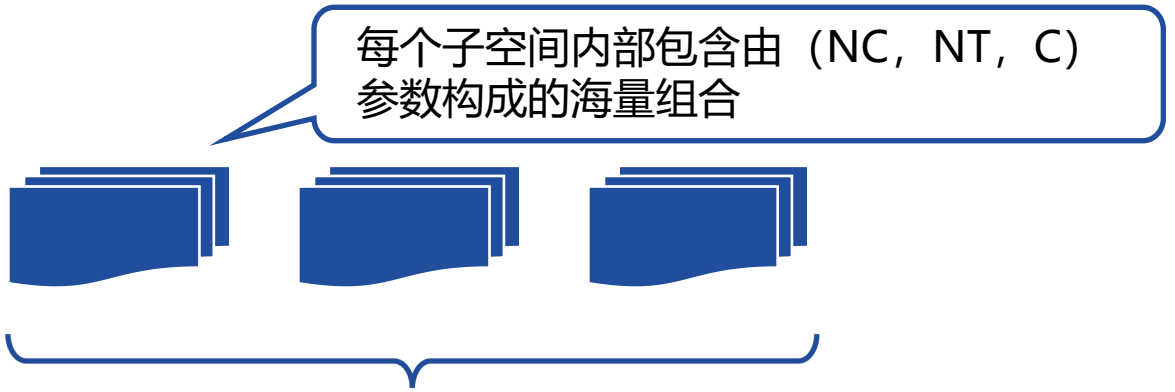
Parameter	Choices
Algorithm (A)	Tree, Ring
Protocol (P)	LL, LL128, Simple
Transport (T)	P2P, SHM
Nchannel (NC)	
Nthread (NT)	
Chunk size (C)	



Algorithm = Ring
Protocal = LL
Transport = P2P

通信策略相关参数

特点：共同决定**通信的宏观模式**，难以建立精确的性能模型，但其组合数量（2x3x2=12种）有限，作为独立子空间被逐一遍历和评估。



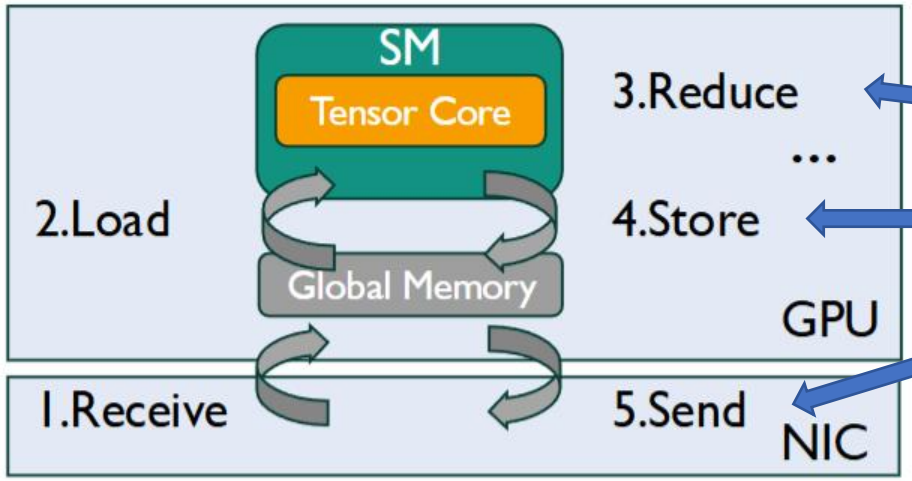
Design-Build Tuning Space

划分为可建模的子空间

Parameter	Choices
Algorithm (A)	Tree, Ring
Protocol (P)	LL, LL128, Simple
Transport (T)	P2P, SHM
Nchannel (NC)	$1 \leq n \leq 128, n \in \mathbb{N}$
Nthread (NT)	$n = 32 \times i, i \in \{1,2, \ldots, 20\}$
Chunk size (C)	$n = 256 \times i, i \in \{1,2,3 \ldots, 8k\}$

资源分配参数

特点： 决定特定策略下通信的**微观并行粒度**，参数组合空间巨大，但性能影响呈现规律性，可被**建模**并进行高效搜索。



NC决定了有多少个数据分区可以并行处理。

NT决定了处理时的线程数。

C定义了数据传输的最小块的大小。

Design-Build Tuning Space

性能建模：两阶段瓶颈模型

阶段一：数据传输 (Transport Phase))

- 从其他GPU读取数据至本地缓冲区
- 性能主要依赖 NC (通道数) 和 C (块大小)

阶段二：协议处理 (Protocol Phase)

- 从缓冲区加载数据至SM核心进行计算
- 性能主要依赖 NC (通道数) 和 NT (线程数)

$$\beta(NC,NT,C)=\min(\beta_0(NC,C),\beta_1(NC,NT))$$

以阶段一为例：

t_0 ： 传输阶段的总耗时 = 传输串行步数 × （一步）传输所花费的时间

$$t_0(NC,C)=\frac{M}{NC\times C}\times(\alpha_0+\frac{NC\times C}{\overline{\beta_0}\times\gamma})$$

同理可得阶段二：

$$\beta_1(NC,NT)=\frac{NC\times NT}{\alpha_1+\frac{NC\times NT}{\overline{\beta_1}\times\gamma}}$$

β_0 ： 传输阶段分析性能的带宽模型

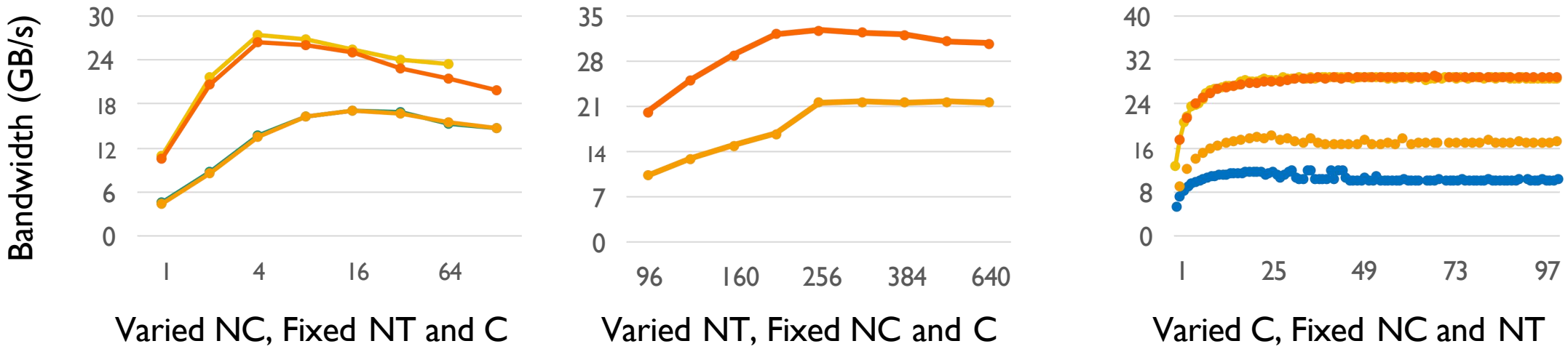
$$\beta_0(NC,C)=\frac{M}{t_0(NC,C)}=\frac{NC\times C}{\alpha_0+\frac{NC\times C}{\overline{\beta_0}\times\gamma}}$$

- 当 NC 和 C 很小时，分母中的固定启动延迟 α_0 占主导地位，整体带宽 β_0 很低。
- 随着 NC 和 C 的增加，α_0 的影响被分摊，带宽 β_0 开始上升。
- 但当 NC 和 C 过大时，拥塞系数 γ 会显著增加，限制了带宽的进一步增长，甚至导致带宽下降。

Design-Build Tuning Space

性能建模

The trends of AllGather(80MB) with config <A, P,T, *, *, *> on 8×A40-NVLink



Varied NC, Fixed NT and C　　Varied NT, Fixed NC and C　　Varied C, Fixed NC and NT

核心发现：单峰函数特性

- 固定任意两个参数并改变第三个时，带宽均呈现出先上升，在达到峰值后趋于平稳或下降的形态。
- 基于这一关键的单峰函数特性，采用高效的坐标下降法 (coordinate descent method)。
- 核心思想：不必在庞大且复杂的多维参数空间里进行盲目搜索，而是可以沿着每一个参数的坐标轴，持续向着性能提升的方向移动。通过对各个参数维度进行交替迭代优化，就能快速地收敛，定位到全局最优的参数配置。

Design-Build Tuning Space

坐标下降法 (coordinate descent method)

1.初始化

- 在当前子空间，随机生成一个初始配置 p 作为当前的最优解，从任一维度出发（NC，NT，C）。

2.在线评测

- 执行配置 p，并**在线测量**其在真实负载下的通信带宽。

3.决策与更新

- 若性能提升:
  - 将该配置 p 更新为新的最优解。
  - 计算一个**自适应学习率**，其大小正比于性能提升的百分比，从而加速收敛。
  - 重置搜索状态，从新的最优解出发，重新审视所有维度。
- 若性能未提升:
  - 切换到下一个参数维度进行探索 (例如，从 NC 切换到 NT)。

4.迭代与收敛

- 重复上述过程，直到连续遍历所有维度都无法找到任何性能提升时，算法收敛并返回该子空间内的最终最优解 。

Algorithm 2: Coordinate Descent Search

Input: Subspace (s).

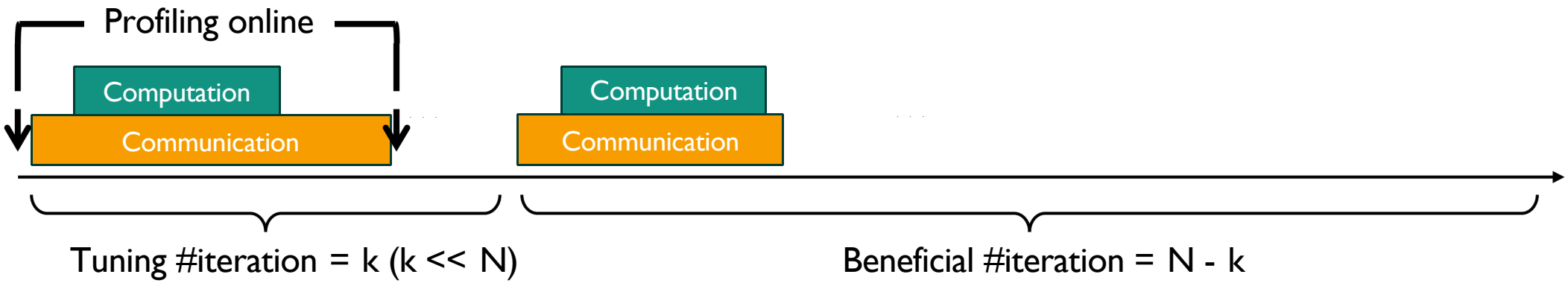
M ← Number of resource parameters.
Randomly generate a config p from subspace s.
optimum ← p
dim, tuned_dim, lr ← 0, 0, 0.01
while tuned_dim ≤ M **do**
    p.ProfileBw()
    if p.BwDelta(optimum) > 0 **then**
        lr ← $\frac{p.BwDelta(optimum)}{p.Bw()}$
        optimum, tuned_dim ← p, 0
    else
        tuned_dim ← tuned_dim + 1
        dim, lr ← dim + 1, 0.01
    p ← optimum
    p[dim] ← p[dim] + lr
return optimum

Design-Implementation

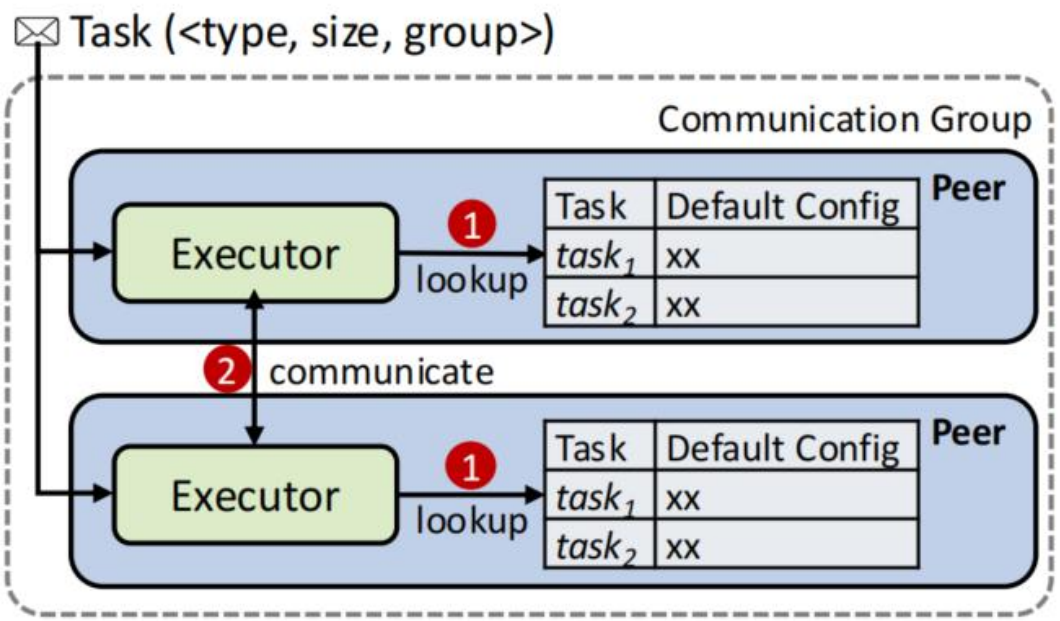
在线调优

- DNN训练是一个**漫长且高度重复**的过程。相同的计算和通信模式会在数万甚至数十万次的迭代中反复出现。
- 不再进行离线测试，将调优过程嵌入到训练任务的初始迭代中，在真实负载下寻找最优配置。

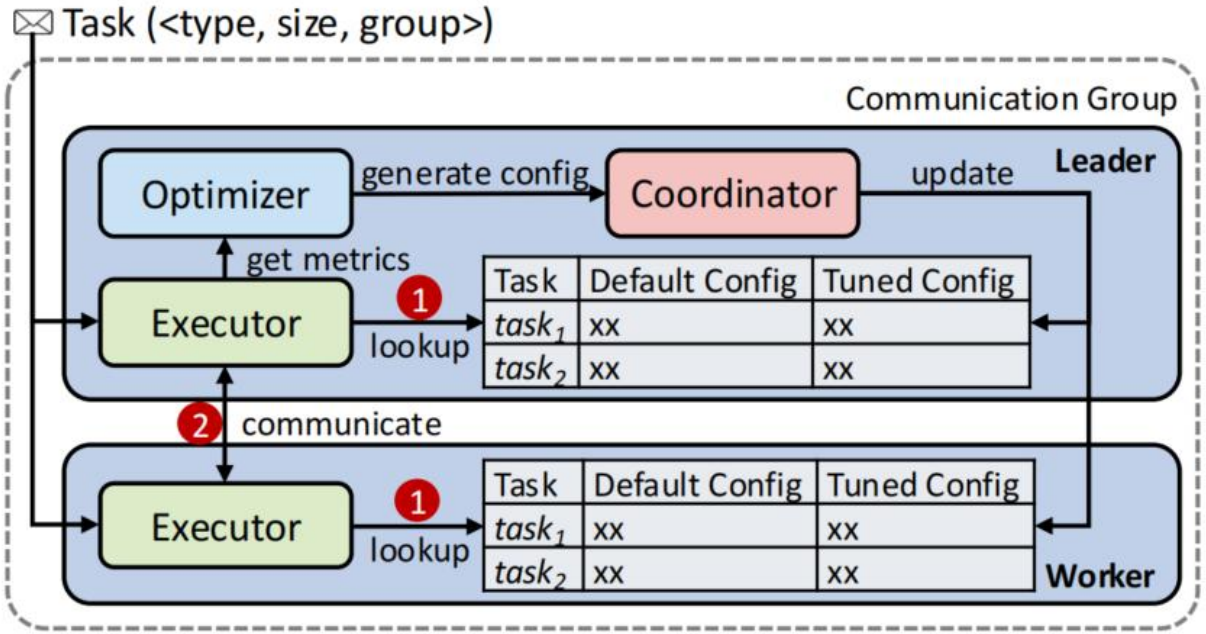
Model	Training Time
Megatron-355M	300 K iteraions
DeepSeek-V3[2]	2 months
MegaScale[3]	70 days



Design-Implementation



The architecture of NCCL



The architecture of AutoCCL

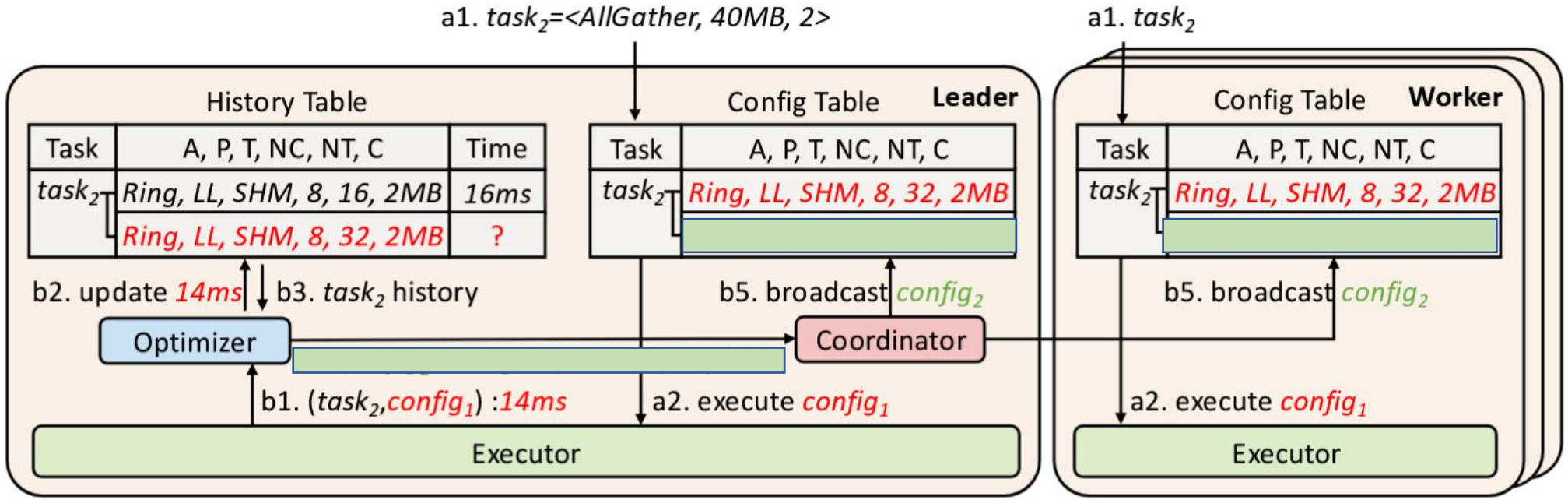
Peer-to-Peer架构

- 每一个GPU独立地根据一个内置的成本模型来为通信任务生成一个默认配置。
- 配置是静态的，无法根据真实的、动态变化的运行环境进行调整和学习。

Leader-Worker架构

- 优化器 (Optimizer)。从执行器接收真实的性能数据，并根据我们之前设计的算法来智能地生成新的、更优的候选配置。
- 协调器 (Coordinator)。负责将优化器找到的最优配置，广播给集群中的所有其他节点。

Design-Implementation



1.执行当前配置 (a1 - a2):

- Leader和所有Worker节点接收到任务后，查询本地Config Table，获得最优配置，使用该配置执行通信任务。

2.性能反馈与分析 (b1 - b3):

- 任务完成后，仅Leader节点的Executor将本次执行的真实耗时（14ms）反馈给Optimizer，更新History Table，
- 分析历史数据发现：将NT从16增加到32，耗时从16ms降至14ms，证明增加NT是有效的优化方向。

3.生成新配置 (b4):

- 根据坐标下降法的原则，Optimizer决定继续沿此方向探索，生成了一个NT值更大的新配置config2（NT=64）。

4.广播与同步 (b5):

- Coordinator通过广播操作，将新生成的config2同步给包括Leader在内的所有节点。

Evaluation

硬件平台：

- 集群A (NVLink): 2节点，16x A40 GPUs。节点内采用高速NVLink互联，节点间为400Gbps InfiniBand。
- 集群B (PCIe): 4节点，32x A40 GPUs。节点内采用PCIe 4.0互联，节点间为100Gbps InfiniBand。

评测模型

涵盖了三种不同规模的大语言模型 (LLM) 和一种计算机视觉 (CV) 模型，采用复杂的混合并行策略。

Baseline：

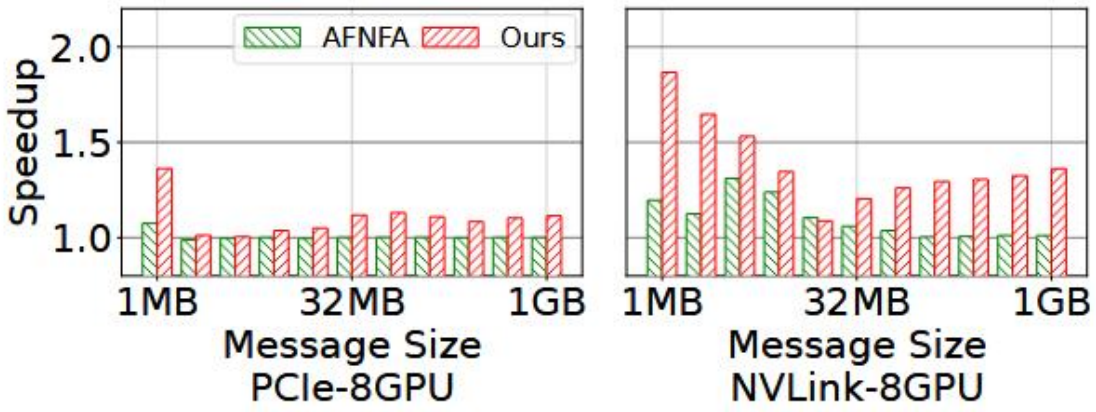
- NCCL v2.18.3-1: 业界领先的通信库，并启用其内置的调优功能。
- AFNFA: 学术界最新的、基于离线分析的自动化NCCL调优工具。

Table 8: DNN model statistics

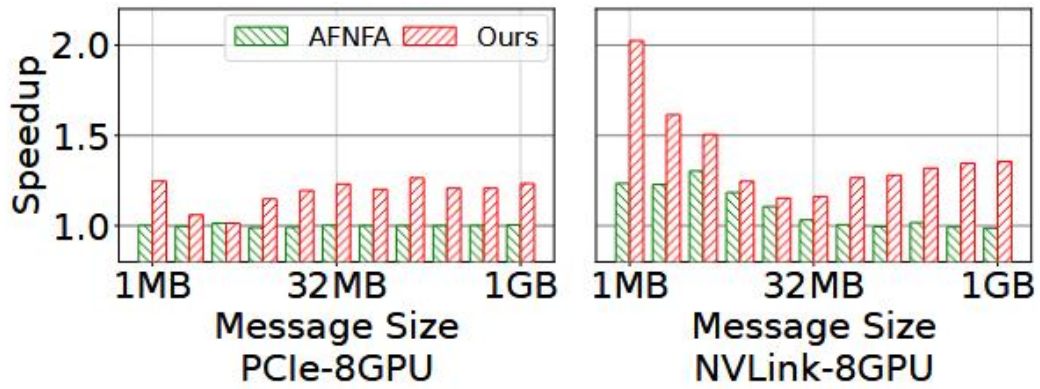
Model	MBS	GBS	TP	PP	DP
Phi-2-2B	8	512	8	1	1-4
Llama-3.1-8B	2	256	8	1	1-4
Yi-1.5-34B	1	1,024	8	4	1
VGG-19-0.14B	32	32×[8,16,32]	1	1	8-32

- 在 GPU 内存限制下将Micro-Batch设置为尽可能大，以提高硬件利用率。
- Global Batch大小遵循具有相似模型规模的GPT系列的设置

Evaluation



(a) *AllGather*

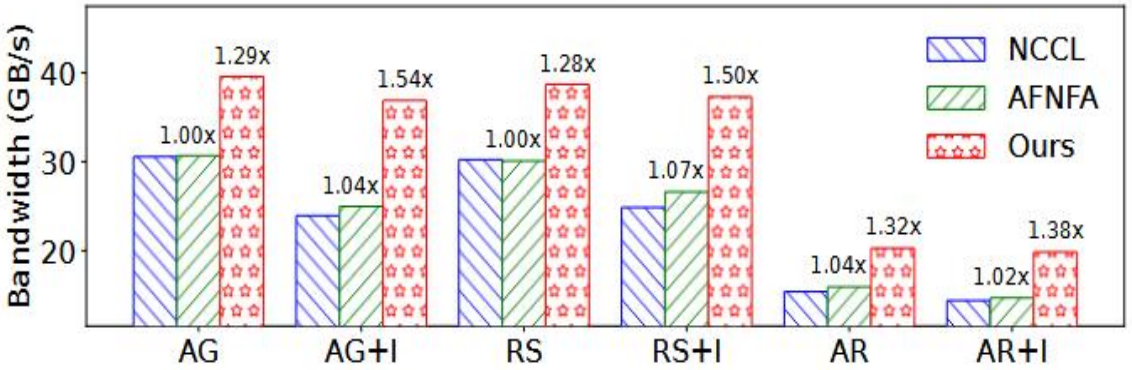


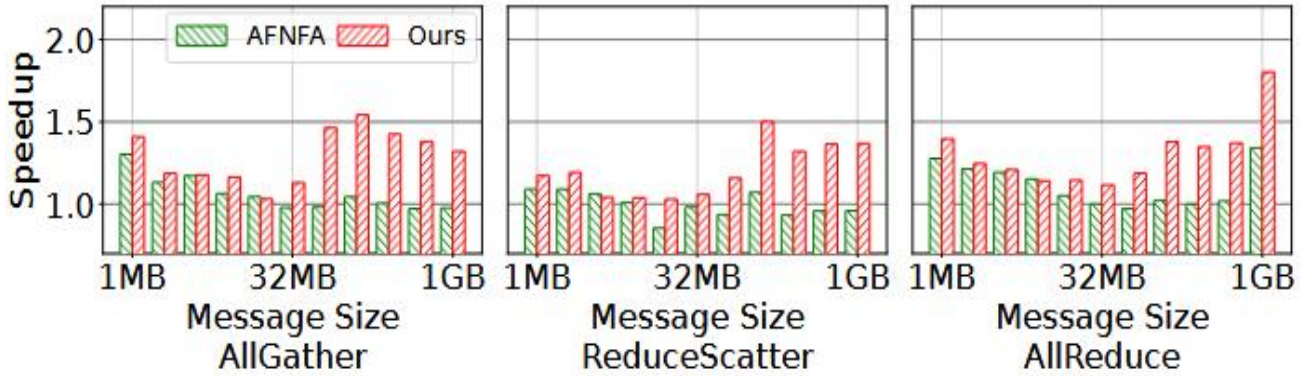
(b) *ReduceScatter*

无计算干扰场景：

- 在 PCIe 环境下，AutoCCL 带宽平均提升 22% 至 28%
- 在已被高度优化的 NVLink 环境下，AutoCCL 的优势更为显著，带宽平均加速高达 1.38倍

Evaluation

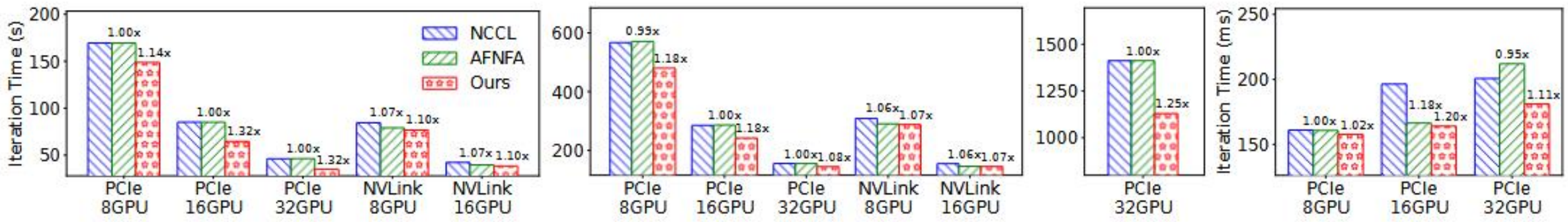




有计算干扰的真实通信性能：

- 性能优势扩大： AutoCCL相较于NCCL的带宽加速比提升至1.29倍至1.50倍 。
- 离线调优失效： AFNFA在此场景下表现极差，证明其离线模型无法适应真实复杂的动态干扰 。
- 强大的抗干扰能力： 经过AutoCCL调优后，在有干扰环境下的最优带宽，非常接近无干扰环境下的最优带宽。

Evaluation



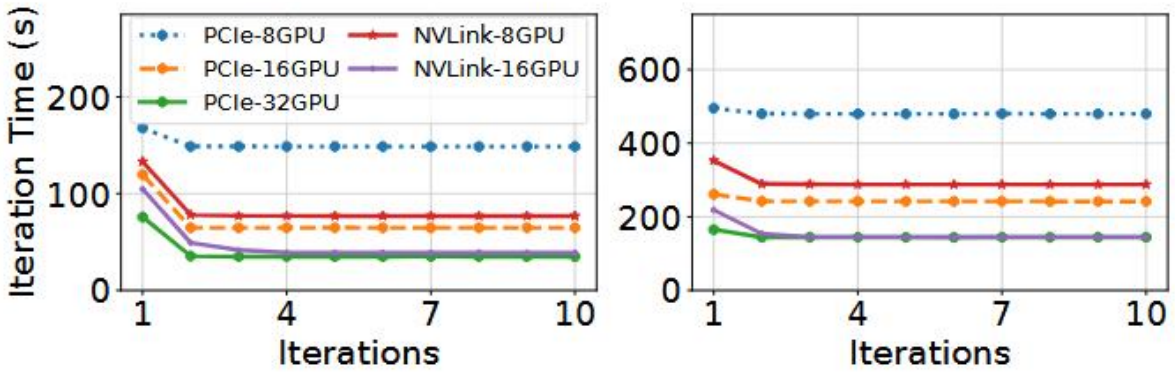
(a) Phi-2-2B (b) Llama-3.1-8B (c) Yi-1.5-34B (d) VGG-19-0.14B

Figure 11: Training iteration time for different models between NCCL, AFNFA, and AutoCCL, with speedup annotated

端到端训练性能结果：

- 在所有硬件平台和DNN模型上，AutoCCL都缩短了单次训练的迭代时间。
- 在PCIe环境下增益最大： AutoCCL在通信瓶颈更显著的PCIe集群上表现尤为出色。其在线调优方法能有效抵御计算干扰。在最佳情况下，AutoCCL可将训练迭代时间缩短超过32% 。
- 在NVLink环境下增益适中： 在通信速度极快、计算成为主导瓶颈的NVLink集群上，优化通信对整体性能的提升空间有限（缩短10%左右） 。

Evaluation



(a) Phi-2-2B (b) Llama-3.1-8B

在线调优效率：快速收敛

对于大型语言模型 (LLMs):

- 由于Transformer模型在单次迭代内会产生数千甚至上万次重复的通信操作，这为调优算法提供了海量的学习样本，因此，AutoCCL仅需几个训练迭代即可快速收敛并找到最优配置 。

对于小型模型 (VGG-19):

- 尽管VGG-19在单次迭代内的重复通信次数较少，但其迭代本身的速度非常快。实验表明，AutoCCL为VGG-19找到最优配置的时间不超过10分钟 。

Thinking

- **文章有什么问题，基于这个 paper 还能做什么?**

参数选择：

- 问题：论文提到，为保证系统稳定性、避免死锁，AutoCCL有意避开了对某些传输层特定参数的调优，可能意味着放弃了部分潜在的性能提升空间。
- 可以将AutoCCL与更智能的故障恢复机制相结合，更“激进”地探索那些有风险但可能带来更高收益的参数组合。

对非迭代或短时任务的调优：

- 问题: AutoCCL的核心优势在于利用DNN训练的海量迭代来分摊和隐藏在线调优的开销 。对于迭代次数较少或通信模式不固定的任务，在线调优的开销可能会变得不可忽略。
- 可以研究迁移调优是否可行。建立一个跨任务的知识库，在一个任务上学到的最优配置知识，是否可以通过迁移或少量样本微调的方式，快速应用到新任务中。

Thinking

- **Paper 的 idea 能不能应用在自己的工作上面?**
  - AutoCCL的在线、自适应调优范式对于依赖静态配置文件的、参数复杂的分布式系统，都可以引入类似的在线调优循环，让系统根据真实负载进行自我优化。

- **这个 paper 能不能泛化?**
  - 从“预测式”到“自适应”优化: 在复杂动态系统中，直接在线测量并做出适应的“自适应”方法，比试图建立精确模型的“预测式”方法更为有效，且适应复杂场景 ，这一理念可以泛化到动态系统的性能调优问题上。

Q & A

Presenter: Yujie Chen

2024.9.20