未来硬件上面向未来 Transformer 的计算与通信扩展

Suchita Pati
spati@cs.wisc.edu 威斯康星大学麦迪逊分校 超威半导体公司

Shaizeen Aga
shaizeen.aga@amd.com 超威半导体公司

Mahzabeen Islam
mahzabeen.islam@amd.com 超威半导体公司

Nuwan Jayasena
nuwan.jayasena@amd.com 超威半导体公司

Matthew D. Sinclair
sinclair@cs.wisc.edu 威斯康星大学麦迪逊分校 超威半导体公司

摘要

神经网络模型的扩展在机器学习问题上带来了显著的质量提升。然而，这种扩展也使人们愈发依赖高效的分布式训练技术。因此，与其他分布式计算场景一样，了解随着模型规模扩大和硬件不断发展，计算与通信将如何相对扩展至关重要。通过深入研究回答这一问题，可以更好地指导未来系统的设计，使其能够高效训练未来的大型模型。

因此，本研究对未来硬件上的未来 Transformer 模型的计算与通信（**计算与通信**）扩展进行了全面的多维分析（算法层面、实证层面和硬件演进）。首先，我们的算法分析表明，随着模型规模扩大，计算通常比通信更具优势。然而，由于内存容量的扩展速度慢于计算，这些趋势正面临考验。接下来，我们通过实证研究未来模型在未来硬件上的计算与通信扩展情况，量化这一优势。为避免在多种设置下对大量 Transformer 模型进行性能分析，我们提取执行区域，并利用算子模型预测成本。这样便能准确研究各种（数百种）未来模型／硬件场景（误差为 < 15%），并将性能分析成本降低 2100×。我们的实验表明，随着模型和硬件演进，通信将占运行时间的很大一部分（40–75%）。此外，在当今模型中常被重叠计算掩盖的通信，在未来更大的模型中往往无法被掩盖。总的来说，本研究凸显了通信在模型规模扩大过程中将发挥日益重要的作用，并探讨了有助于应对这一问题的技术和即将出现的新兴技术。

1 引言

近年来，机器学习（ML）和深度神经网络（DNNs）改变了社会，在语音识别 [73]、图像分类 [26, 33, 36, 61, 66, 67]、机器翻译 [25]、自主智能体 [37]、和语言处理 [12, 16, 48]等任务上的准确性也显著提升。这种变革性影响得益于以下三者形成的良性协同：(1) 更高效的硬件，(2) 更大的数据集，以及 (3) 算法进展（包括模型规模的指数级增长）；而这些进展又进一步受益于更高效的硬件和更大的数据集。然而，模型规模的增长速度远快于 (1000×) GPU 内存的增长速度 (5×) [52]。随着模型规模扩大，计算需求也不断增加，这使得人们愈发依赖分布式

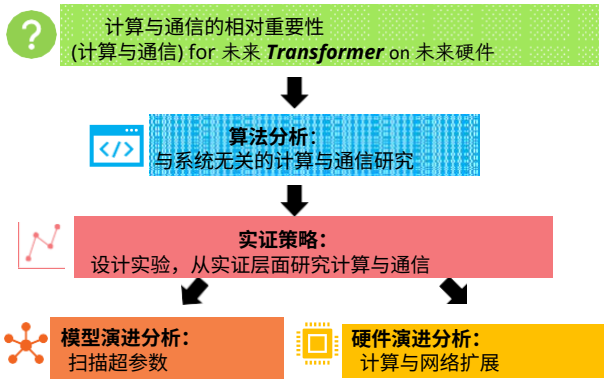


图1：计算与通信分析概览。

训练来实现高效的大规模训练：多个加速器（例如 GPU）协同训练一个深度神经网络模型，以利用多个加速器的总内存容量及其计算能力。因此，理解随着深度神经网络规模扩大、硬件不断演进，分布式训练中的计算和通信将如何相对扩展，至关重要。

本文开展了这项分析，并将其称为**计算与通信**，旨在更好地指导未来系统的设计，使其能够高效训练未来的大型模型。该分析针对未来硬件上的未来Transformer模型，开展了全面的多轴（算法层面、实证层面、硬件演进）计算与通信分析。尽管如今各种各样的深度神经网络模型都在使用，但Transformer已经成为一种候选通用架构[11]，能够处理跨多个模态的多项任务（例如文本、视觉和音频）。此外，正如近期研究人员训练单个 1.2B Transformer执行机器人学、仿真环境以及视觉和语言领域数百项任务所展示的那样，Transformer在通用人工智能方面也取得了长足进展[55]。因此，鉴于Transformer的普及程度、日益增长的重要性和能力，我们将重点放在Transformer上（其他深度神经网络将在第6.4节讨论）。

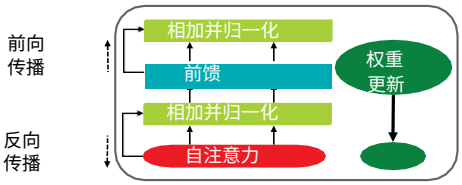
图 1 展示了我们方法的概览。我们首先对Transformer模型中的计算操作和通信操作进行算法分析。这项分析提供了不依赖具体系统的计算与通信扩展视角，这一点尤为重要，因为系统／基础设施能力种类繁多，涵盖从少量独立加速器到配备先进互连网络的加速器集群。此外，这项分析也为实证研究奠定了坚实基础。

计算与通信之间的关系，且不会产生显著开销。我们的算法分析表明，计算操作的复杂度通常高于通信量（数据大小）。我们将此称为计算相对于通信的优势 。以计算为主导的执行配置通常具有优势，因为：(a) 传统上，尤其是在加速器领域，人们对计算（每秒浮点运算次数）扩展的关注远高于对通信（带宽）扩展的关注；(b) 算法／系统优化通常会通过重叠通信与有效计算来提升性能。因此，计算的优势也有助于隐藏通信成本。在串行执行或计算与通信重叠的情况下，计算都具有这种优势；这两种情况都出现在分布式训练中。然而，从模型扩展和内存容量趋势的角度来看这些观察结果时，我们发现这种优势正面临挑战。

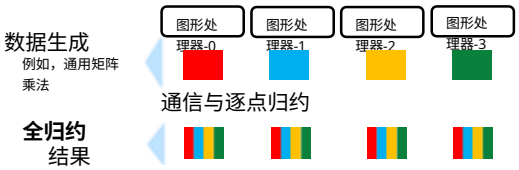
为了解计算相对于通信的优势可能如何受到未来模型和未来硬件的影响，我们对计算与通信扩展进行了实证研究。这种方法面临若干挑战，包括需要研究许多模型/硬件演进情景，而每种情景都会产生大量性能分析成本。我们提出的实证策略通过以下方式应对这些挑战：(a) 设计受算法分析指导的受控实验，(b) 仅执行特定的感兴趣区域（ROI），以降低性能分析成本，以及 (c) 为训练操作建立算子级模型，并证明这些模型能够准确捕捉不同超参数下的运行时趋势。这些方法使我们能够以2100× 更低的性能分析成本研究数百种未来模型/硬件情景。此外，鉴于实证策略具有普适性，我们还讨论了如何将计算与通信扩展分析应用于未来的 Transformer模型。

由实证策略指导的实验支持了我们从算法分析中得出的结论。具体而言，我们发现计算相对于通信的优势正面临压力：未来 Transformer 的训练时间中，最多有 50% 将用于通信数据。此外，如今可以重叠或隐藏的通信，在未来模型中可能超过计算时间，进一步提高通信所占的比例。而且，如果过去计算能力相对于通信带宽的扩展趋势持续下去，那么通信将在未来系统中成为更大的瓶颈（占训练执行时间的> 75%）。我们的工作主要贡献如下：

- 随着模型规模不断扩大、训练日益分布式化，深入理解计算与通信的扩展规律对于有效设计未来系统至关重要。因此，我们从多个维度（算法层面、实证层面、硬件演进）全面分析了未来硬件上未来 Transformer模型的计算与通信扩展。
- 通过算法层面/与系统无关的分析，我们表明，尽管无论在串行还是重叠场景中，计算都比通信更具优势，但模型的扩展速度超过了内存容量的增长速度，正在削弱这一优势。
- 接下来，我们设计了一种策略，用于对未来模型的计算与通信进行实用的实证研究。该策略提取执行感兴趣区域，并准确地（误差为 <15%）预测未来模型的操作时间，从而降低性能分析成本（比执行所有模型快 2100× 倍）。



(a) 计算：Transformer层



(b) 通信：全归约

图 2：分布式 Transformer 组件 **ts.**

- 由实证策略指导的实验表明，随着模型规模扩大，通信将发挥越来越重要的作用：训练时间中用于通信的比例最高可达 50%。此外，如果过去计算与网络的扩展趋势持续下去，通信最终可能成为分布式训练中的关键瓶颈：训练时间中用于通信的比例最高可达 75%，而一些此前隐藏的通信成本也将显现。
- 我们将讨论应对这一新兴通信挑战的技术与即将推出的技术（第 5 节）。

总体而言，若 Transformer 不发生剧烈演变，我们的研究凸显了随着模型规模扩大，通信所占的比重日益增加。

2 背景与动机

在此，我们简要概述 Transformer模型 [68], 及其分布式训练，以及开展计算与通信扩展研究的必要性。

2.1 Transformer：未来模型的构建模块

Transformer [68] 已成为适用于广泛任务和领域的通用架构。近期一项分析了不同模态 Transformer 相关论文的研究显示，越来越多的解决方案（例如，文本领域占 41%，图像领域占 22%）以 Transformer 作为基础模型 [11]。这些 Transformer 的基本构建模块是编码器层或解码器层。如图 2a 所示，这些层包含注意力子层和全连接（FC）子层，两者后面都接有残差连接和层归一化。这些子层体现为矩阵乘法运算（通用矩阵乘法），后接少量逐元素运算；这些运算通常会与通用矩阵乘法融合 [18, 21, 64, 70] 。编码器层和解码器层类似，但解码器的注意力机制通用矩阵乘法输入经过掩码处理，这会导致推理时的计算行为不同，但不影响训练（这是我们的关注重点）。

Transformer模型的发展主要集中在改变层类型（编码器与解码器）、增加层宽度和/或增加层数上。所有Transformer模型皆是如此；从BERT [16] （3亿个参数）开始，直至其最

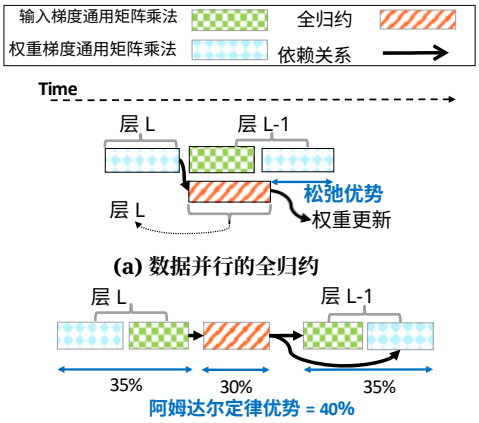


图 3：分布式环境中的全归约。

其近期后继模型 MT-NLG（5400 亿个参数），以及介于两者之间的许多其他模型 [12, 15, 34, 39, 48, 60, 65, 74]。因此，尽管Transformer模型的规模已大幅增长，其基本计算组成部分却大体保持不变。因此，由于这些模型采用相似的架构，只是超参数不同，我们以 BERT架构为基线，并调整超参数来研究规模更大的未来 Transformer模型。不过，我们用于算法分析和实证分析的方法也可扩展到其他深度神经网络（参见第 6.4 节）。

2.2 大规模训练的分布式训练

大多数 Transformer模型采用分布式训练技术，并协同使用多个加速器（例如 GPU）来训练单个 Transformer模型。训练大型模型通常必须分布模型，因为模型所需的内存容量超过了单个设备的容量。此外，多个设备的总计算能力可通过并行处理大型输入数据集来加速训练。因此，随着 Transformer模型及其数据集（通常是大型未标注文本语料库）的规模增长了几个数量级（第 3.5 节），分布式技术不仅已成为事实上的标准，而且所需的设备数量也越来越多。未来模型的分布式训练规模还将继续扩大（第 4.3.2 节）。尽管我们侧重于训练，但我们的分析也适用于推理（参见第 6.3 节）。

2.3 分布式训练技术及相关通信

虽然存在许多不同的分布式技术，但它们都涉及设备之间的通信，用于传输（例如，流水线并行 [28]）、归约（例如，张量 [60] 与数据并行）、聚合（例如，基于ZeRO的优化 [52]）或交换（例如，专家并行 [31]）信息。这些通信模式通常由一组集合通信操作处理，例如全归约、全收集和全交换。不过，我们重点关注全归约——这是两种最有效且应用最广泛的分布式技术（第3节.1）、数据并行和张量并行，接下来我们将对此进行讨论。我们还将在第6节.1讨论其他形式的通信所带来的影响。

2.3.1 全规约通信类型。如图2b所示，全归约（AR）集合通信会对所有参与设备生成的数据进行规约，并将规约后的数据广播给这些设备。针对不同的系统拓扑，AR 也有不同的优化实现。虽然 AR 集合通信保持不变，涉及

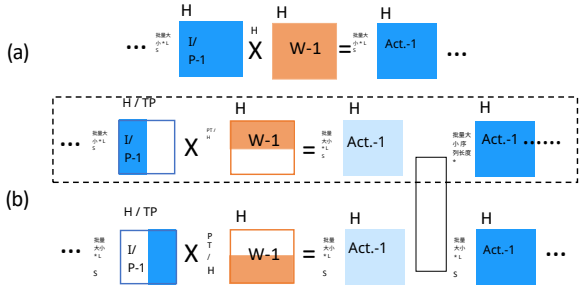
在数据并行（DP）和张量并行（TP）设置中，它既涉及通信也涉及计算（例如逐元素求和），但其使用方式以及对整体训练行为的影响各不相同。

2.3.2 数据并行（DP）与松弛优势。数据并行有效地
通过在所有设备上复制模型，并让每个设备处理互不重叠的输入集，来增大训练输入批量大小。然而，
这要求模型副本保持同步，因此设备必须在反向训练过程中对权重梯度执行全归约。在数据并行训练中，如图 3a 所示，一层的梯度全归约可以与另一层的梯度计算异步进行。因此，相关通信有可能与计算重叠并被其隐藏。然而，只有当计算的执行时长等于或超过通信时长时，才能实现完全重叠。我们将重叠执行的计算与通信之间的这一差值称为计算松弛优势，如图 3a 所示。

2.3.3 张量/水平并行（TP）与阿姆达尔定律优势。TP 通过将模型拆分到多个设备上，有效提升模型可用的内存容量（如图 4 所示）。它将模型层拆分到多个设备上，使每个设备都持有并运行每一层参数的一个子集。然而，这种切分会导致每个设备在训练的前向（和反向）传播过程中只生成部分层激活值（和误差）（图 4(b) 中的浅蓝色矩阵），因此需要通过全归约在所有设备间进行归约，以生成最终的层输出（图 4(b) 中的深蓝色矩阵）。此外，一层的前向和反向执行依赖于另一层对激活值和误差进行全归约。因此，与 DP 不同，在 TP 中，计算与通信并非异步，通信处于模型执行的关键路径上（图 3b）。如图 3b 所示，我们将执行计算与串行化通信之间的差异称为计算的阿姆达尔定律优势。

2.4 为什么要研究计算与通信扩展性的演变

尽管通信是分布式训练所必需的，但当通信处于关键路径上时，可能会导致计算资源闲置，并限制吞吐量随设备数量增加而扩展。因此，了解随着模型规模扩大和硬件演进，计算与通信如何相对扩展至关重要。与加速器设计人员长期关注的系统计算吞吐量不同，网络带宽的扩展幅度并不大（例如，12× 计算性能提升与2× 网络带宽提升相比[32]）。如果计算持续以更快的速度扩展，同时通信量不断增加，未来系统将无法高效训练未来的大规模模型。为解决这一问题，我们对未来硬件上的未来模型的计算与通信扩展进行多轴分析（算法层面、实证层面、



Fig图 4：层操作，(a) 原始方式，或采用数据并行 (b)　采用张量并行。

硬件演进）。这种多管齐下的分析既能为未来的系统设计提供信息，也能为其指明方向，从而更好地支持未来模型的大规模训练。

3　计算与通信：算法分析

对计算与通信进行算法分析十分重要，因为这为我们对未来模型得出有意义的结论奠定了坚实基础。此外，正如我们在第4节所展示的，这有助于制定实证策略，利用现有硬件研究未来硬件上的模型扩展。同样重要的是，这种分析不依赖于特定的系统与基础设施，因此能够提供通用的计算与通信扩展视角，确保所得结论无论针对何种系统都具有参考价值。

3.1　研究的分布式技术

我们研究了采用数据并行（DP）和张量并行（TP）设置的Transformer（见第2.1节）。虽然训练 Transformer 还会用到其他分布式技术及其组合，但 DP 和 TP 是最常见的 [13, 60]。随着模型规模扩大以提升准确性（第2.1节），训练数据集的规模也随之增大，以跟上模型扩展的步伐。在这些情况下，DP 至关重要，它可以将大型数据集分配到多个设备上，分而治之。同样，TP 对于训练无法在单个设备上容纳的大型模型也不可或缺。此外，TensorFlow 和 PyTorch 等主流深度神经网络框架都大力支持 DP 和 TP。因此，我们重点研究 DP 与 TP 的结合使用，并在第6.1节讨论其他机制。

3.2　重要超参数

模型组件的大小及其成本由模型的超参数 [47]决定。如图4(a)所示，在 Trans-
影响权重大小的关键超参数
激活值包括：层宽度或隐藏维度（H）、输入
批量大小（B）和输入序列长度（SL）。尽管 Transformer 训练期间还会调整其他超参数（例如层数、注意力头数、学习率），但它们不会直接影响运算的规模。因此，我们使用 H，B、和 SL（列于表 1）以算法层面且不依赖硬件的方式分析计算与通信（第 3.3 节）。

除了这些超参数之外，分布式设置也会影响运算规模。在数据并行（DP）中，由于模型只是

首字母缩略词	定义	首字母缩略词	定义
B	批量大小	H	隐藏维度
SL	序列长度	TP	张量并行度

表 1：超参数

	BERT [16]	T5 [50]	GPT-2[48]	Mega.-语言模型[60]	T-NLG [42]	GPT-3[12]	MT-NLG[63]	PaLM [13]
Year	’18	’19	’19	’19	’20	’20	’21	’22
层数	24	24	48	74	78	96	105	118
H	1K	1K	1600	3K	4256	12K	20K	18K
#注意力头	16	128	25	24	28	96	128	48
大小（批量大小）	0.34	11	1.54	8.3	17	175	530	540
Type	编码器。	编码器-解码器。	解码器。	Dec.	解码器。	解码器。	解码器。	Dec.
SL	512	512	1K	1K	1K	2K	2K	2K
全连接层维度	4K	4K	6400	12K	17024	48K	80K	72K

表2：不同的NLP模型超参数。

复制时，操作大小不受影响。相反，在张量并行（TP）中，操作会被切分，如图4(b)的虚线框内所示。因此，我们的算法分析也考虑了 TP 度（我们用 TP 同时指张量并行及张量并行度），即模型所跨越的设备数量。

3.3　计算方面的阿姆达尔定律优势

如第 2.3.2 节和第 2.3.3 节所述，采用数据并行和张量并行的分布式设置中存在两种以全归约形式进行的通信。这里我们考虑与张量并行相关的通信，它位于模型执行的关键路径上（如图5(b)所示），下文称为串行化通信。为了评估计算的相对优势，即阿姆达尔定律优势（第 2.3.3 节），我们分析 Transformer层中所有计算和通信操作的相对占比。计算包括代表 Transformer 关键子层（注意力机制和全连接层，第 2.1 节）的矩阵乘法（通用矩阵乘法），以及构成其余激活函数和归一化子层的其他逐元素运算与归约运算（图5(b)中的 1a）。然而，现代 Transformer 实现通常会将 [18, 21, 64, 70] 非通用矩阵乘法运算与前面的通用矩阵乘法融合，以最大限度地提高片上数据复用（图5(b)中的 1b）。因此，我们的算法分析仅考虑作为主要计算的通用矩阵乘法。由于通用矩阵乘法通常是计算受限的，我们将其算法成本评估为所执行运算（乘法运算和加法运算）的总数：$2 * M * N * K$（其中 M、N 和 K 是其矩阵维度，根据图4所示的模型超参数推导得出）。在一个 Transformer层中，有三组关键的通用矩阵乘法；[47], 其计算复杂度（采用张量并行）如
下方方程（以及图5(b)②的方程）。

全连接层通用矩阵乘法运算。$= 2 * (4 * H * H/TP * SL * B)$

$$= O(H^2 * SL * B/TP) \quad (1)$$

$$\text{Attention GEMM Ops.} = 2 * (H/TP * SL * SL * B)$$

$$= O(H * SL^2 * B/TP) \quad (2)$$

$$\text{Linear GEMM Ops.} = 3 * 2 * (H/TP * H * SL * B)$$

$$= O(H^2 * SL * B/TP) \quad (3)$$

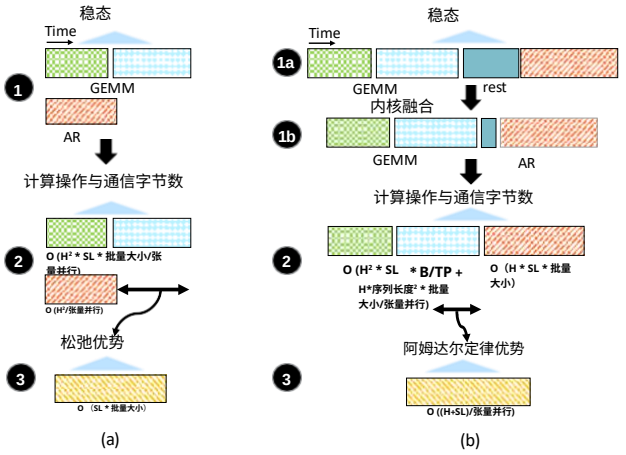


图5：计算相对于（a）重叠通信的余量，以及相对于（b）串行化通信的优势。

结合上述公式，可得

$$\text{Overall Compute Ops.} = \mathrm{O}\,(H * SL * B/TP * (H + SL)) \tag{4}$$

对于串行化通信，我们考虑需要通过全归约进行归约的数据总字节数。串行化全归约运算会减少层生成的激活值和误差。因此，它们的大小是通用矩阵乘法输出矩阵大小的倍数（取决于精度），即 $M * N$，也可以用超参数表示（见图4）。在一个 Transformer 层中，有四个此类串行化的全归约操作，其复杂度均如公式5所示（图5(b)中的2）。然后，利用公式4和公式5，我们可以得出公式6中的计算操作数与通信字节数之比，该比值表示计算相对于通信的**阿姆达尔定律优势**（图5(b)中的3）的复杂度。

$$\text{Overall Communication Bytes} = (precision/8) * (H * SL * B) = \mathrm{O}\,(H * SL * B) \tag{5}$$

$$\textit{Amdahl's law edge} = \mathrm{O}\,((H^2 * SL * B/TP) + (H * SL^2 * B/TP)) / \mathrm{O}\,(H * SL * B) = \boldsymbol{O((H + SL)/TP)} \tag{6}$$

这带来两点影响。首先，给定这些超参数的取值（第 3.5 节和表1），由于 $(H + SL)$ 始终大于 TP，计算（操作数）相较于通信（字节数）具有优势。其次，如果 TP degree 在不同 Transformer模型之间的增幅大于 $(H + SL)$ 的增幅，则通信所占比例会增加，反之亦然。

3.4 计算的余量优势

与我们对串行化通信的分析类似，对于 DP 的重叠通信（如图5(a)所示），我们从算法层面对每个 Transformer 子层中的计算操作与重叠通信操作的相对开销进行分析，并评估计算隐藏通信的能力（第2.3.2节）。对于计算，我们考虑用于计算反向传播权重的通用矩阵乘法

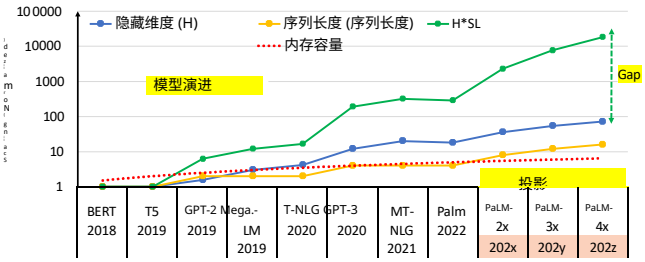


图6：模型和设备内存容量趋势。

梯度（权重梯度，WG）和误差（输入梯度，IG）；对于通信，我们考虑权重梯度的大小（图中以 1 表示图5(a)）。方程7和8（图5(b)中的 2）展示了全连接子层中计算操作和通信字节数的示例。方程9（图5(b)中的 3）使用方程7和8表示通用矩阵乘法执行的运算次数与通信的全归约字节数总量之间的比值，以及计算的**余量**复杂度（即隐藏通信的能力）。

$$\text{FC WG + FC Error GEMM Ops.} = 4 * (4 * H * H/TP * SL * B) = \mathrm{O}\,(H^2 * SL * B/TP) \tag{7}$$

$$\text{Communication bytes} = (precision/8) * (4 * H * H/TP) = \mathrm{O}\,(H^2/TP) \tag{8}$$

$$\textit{Slack advantage} = \mathrm{O}\,(H^2 * SL * B/TP)/\mathrm{O}\,(H^2/TP) = \boldsymbol{O(SL * B)} \tag{9}$$

这一 $SL * B$ 因子为计算操作提供了额外余量，以隐藏通信字节的成本。然而，减小输入大小（$SL * B$）会减少余量，并可能使通信成本显现出来。此外，由于 Transformer 的各层具有相同的复杂度，这些公式和比值适用于所有层。因此，这一观察结果对所有 Transformer 反向传播子层均成立。

3.5 模型扩展对计算优势与余量的压力

我们在第 3.4 节和第 3.3 节中的算法分析表明，计算相较于串行化通信具有（阿姆达尔定律意义上的）优势，并且拥有足以隐藏重叠通信的余量。然而，其程度这种优势和余量会因超参数而异：随着 H 或 SL 增加，优势会扩大；随着 TP 增加，优势则会缩小。反之，随着 B 或 SL增加，余量会扩大。近年来，H 和 SL 显著增加（见表2）。如图6所示，预计这些趋势会持续下去，因为较大的 H 和 SL 与模型质量提升密切相关 [49]。然而，增加 H 会使模型参数呈二次方增长。同样，增加 SL 会使激活值大小线性增长，从而增加 Transformer 的内存需求。我们将这两者的乘积 $H * SL$用作衡量内存需求扩展的指标。图6还显示，如果设备内存容量继续按线性趋势增长，模型未来的内存需求与可用容量之间的间隔只会进一步扩大。因此，采用更小的 B和更大的 TP 已势在必行（详见第 4.3.2 节和图9(b)），以降低并分散内存

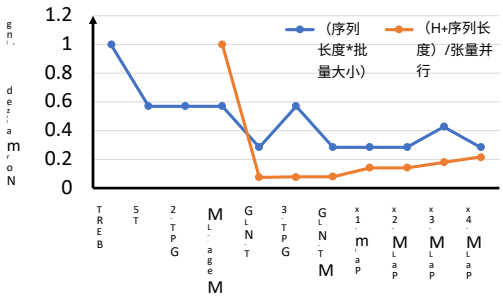


图7：余量和优势的算法扩展。

压力（图6中的间隔）。如果 B 和 TP 的增长超过 H 和 SL的相应增幅，那么由此产生的算法层面优势比（即 $(H+SL)/TP$）和余量（即 $SL*B$）可能会降低，从而使更多通信暴露在关键路径上。我们在图7中展示了这种扩展趋势；该图绘制了所有研究的 Transformer 的计算松弛和计算相对于通信的优势，并以 BERT 的对应值为基准进行归一化。由于 B 显著下降（=1），计算松弛减少了 ~75%。同样，由于所需 TP，增加，计算优势下降了 ~80%。因此，尽管计算在算法层面一直优于通信，但模型扩展和内存容量趋势正在削弱这一优势。

此外，算法分析并未考虑执行某项操作或传输一个字节的成本。因此，单个 Transformer 的计算操作与通信字节之比并不能直接转化为执行时间比，而且计算实际上可能并无优势或余量较小。我们将在第4节中对此进行详细探讨。尽管如此，我们的算法分析仍揭示了随着 Transformer 不断演进，这种优势和余量如何增减。

4 计算与通信：实证分析

到目前为止，我们的分析表明，计算在算法层面优于通信。然而，随着模型演进，这一优势正面临挑战（第3节）。我们通过实证分析，以执行时间为指标量化这一优势。但由于全面的实证研究成本高昂，我们转而提出一种基于算法分析的策略，利用现有硬件预测未来硬件上任意未来模型的计算与通信比较。

4.1 实证分析面临的挑战

实证分析必须经过精心设计，因为模型演进会导致需要考虑的情景（超参数）激增，进而需要开展大量实验。考虑硬件演进时（由于硬件参数众多），这一问题会进一步加剧。因此，在设计实验以研究模型与硬件演进时，仔细确定关注变量至关重要。此外，即使有条不紊地探索超参数和硬件参数，性能分析成本仍可能非常高，尤其是在需要对完整训练迭代进行性能分析的情景中。因此，要对未来模型和硬件的计算与通信扩展进行实证研究，仔细考察变量空间、密切关注如何控制性能分析开销，以及设计高保真度的模型与硬件演进方案，都是至关重要的。

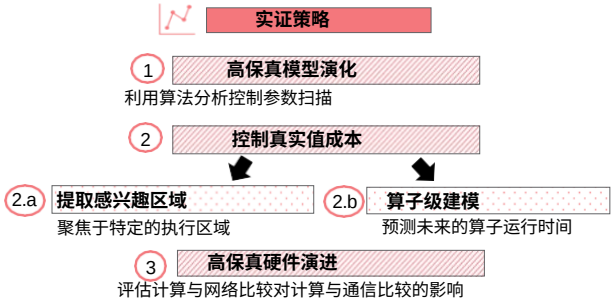


图8：拟议的实证策略——解决方案列表。

4.2 拟议的实证策略

接下来，我们将讨论实证策略的各个组成部分（图8中的步骤），以应对上述挑战。

4.2.1 高保真模型演化（步骤 1）

为了有效研究未来模型的计算与通信扩展，必须仔细考量模型演进。表 1 突出了 Transformer模型的重要超参数。此外，第 3.5 节表明，从历史上看，为了提高准确率，模型一直在扩展隐藏维度（H）和序列长度（SL）。其他超参数（批量大小 B 和张量并行度 TP）取决于底层系统（包括计算能力和内存容量）。对这一超参数空间进行简单但穷尽的探索，有助于忠实地研究模型演进。然而，即使排除不切实际的配置（例如，张量并行度较低时使用大模型和大批量大小），这类探索仍需要运行数量庞大且不切实际的实验。

我们通过立足于算法分析来克服这一挑战。具体而言，我们利用第3节中确定的计算与通信扩展比率来设计受控实验。对于通信与计算重叠的场景（例如数据并行），由于算法层面的计算与通信比率为 O ($SL*B$)，我们重点针对不同的隐藏维度（H）取值，扫描 $SL*B$，以研究计算松弛优势如何随未来模型扩展。不过，这仍需要对若干不同的（来自 H，和 $SL*B$）迭代进行性能分析。此外，对于串行化通信（例如张量并行），由于计算与通信比率为 O $((H+SL)/TP)$，我们只能将批量大小 B作为变量。我们还提出了其他策略来简化这一探索（第 4.2.2 节）。

4.2.2 驯服真实成本（步骤 2）

虽然算法分析有助于缩减搜索空间，但仍需采用其他方法来降低性能分析成本。因此，我们运用应用理解，在可能的情况下提取执行中的特定片段。当需要完整的迭代时间时，我们依靠高保真算子级模型，在不实际运行模型的情况下，有效预测不同 Transformer 配置的运行时间。下面将进一步说明这些策略。**感兴趣区域（ROI）提取（步骤 2a）**为了研究通信与计算重叠的场景（例如，数据并行）下计算与通信的扩展性，我们发现，只需提取未来模型中会出现的特定计算片段（例如，通用矩阵乘法）和通信片段（例如，全归约），并对其进行性能分析即可。

参数 / 设置	值
H	1K、2K、4K、8K、16K、32K、64K
{B}, {SL}	{1,4}, {1K, 2K, 4K, 8K}
{TP degree}, {DP degree}	{ 4, 8, 16, 32, 64, 128, 256}, {任意}

表3：所研究模型的参数和设置。

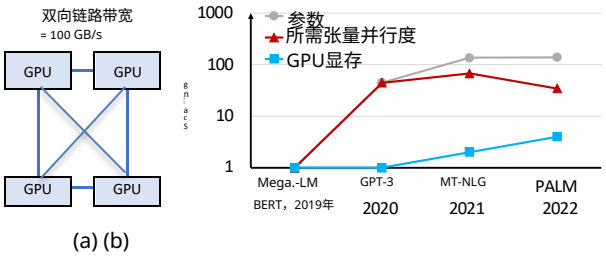
仅在硬件上执行这些区域。这些受控的
这些实验不仅有助于我们研究随着模型规模扩大和硬件不断演进，计算松弛如何变化，还能避免为所有感兴趣的配置运行完整训练迭代所产生的成本。

算子级模型（步骤 2b）要研究通信与计算串行执行（例如张量并行）的场景下计算与通信扩展的情况，执行感兴趣区域并不容易。为了量化计算相较于通信所享有的阿姆达尔定律优势，有必要研究完整的训练迭代。为降低随之而来的性能分析成本，我们发现，构建高保真算子级模型并综合其结果，有助于预测整个网络的行为。具体而言，对于 Transformer 层执行过程中每次训练迭代都会重复执行的每个算子（例如通用矩阵乘法、层归一化等），我们通过算法分析确定影响其执行时间的超参数。我们还在现有硬件上对所有运算进行性能分析，同时独立改变各项超参数。这有助于我们确定每种运算的执行时间与超参数之间的相关性。进而，我们可以预测不同超参数取值下每个算子的执行时间。这样一来，无需付出高昂的性能分析成本，我们便能预测各种超参数取值下 Transformer 的行为。我们对算子级模型的评估（第4节.3.8）表明，这些模型可靠，能够准确捕捉运算行为随超参数变化的情况。

4.2.3 高保真硬件演进（步骤 3）

最后，与模型演进类似，严谨地设定硬件参数搜索空间同样至关重要。因此，我们确定了影响计算与通信扩展的关键因素：计算吞吐量（FLOPS）、网络带宽和内存带宽。在这些因素中，我们重点关注前两项。尽管这三项因素都会影响通信性能，但高效的通信原语（例如全归约）实现通常采用流水线方式。因此，它们能够将内存访问与网络传输重叠起来；由于网络传输通常占主导地位，内存带宽的影响相对较小。此外，虽然计算操作同时取决于计算 FLOPS 和内存带宽，但 Transformer 的关键操作（例如通用矩阵乘法）通常是计算受限的（例如，Gshard [35] 报告称，峰值 FLOPS 利用率为 >85%），且内存带宽利用率较低 [47]。因此，我们重点关注计算 FLOPS 和网络带宽，尤其是根据不同厂商 GPU 的历史数据得出的二者相对扩展比率（详见第4节 4.3.6）。

4.2.4 相较于穷举式性能分析的优势。我们的实证策略降低了执行和性能分析成本（第4.3.8节）。首先，我们的算法分析确定了需要遍历的超参数子集，从而限制了需要考虑的超参数组合（$SL*B$，而不是分别遍历SL 和B）。其次，算子级



(a) (b)

图9：系统：(a) 4-GPU 节点 (b) 随模型规模变化的张量并行扩展。

模型可以仅通过执行和性能分析单次迭代，预测许多（196种）不同配置下的串行化通信。最后，聚焦于特定的感兴趣区域有助于避免为重叠通信执行端到端迭代。

4.3 实验分析的观察结果

接下来，我们将介绍根据实证策略开展实验所得的观察结果。

4.3.1 系统配置。我们在一台配备 AMD RyzenTM ThreadripperTM CPU 和四个 AMD InstinctTM MI210 加速器（GPU）[8]（见图9(a)）的系统上运行实验，每个加速器配有 64GB 高带宽内存。这些 GPU 通过 AMD Infinity FabricTM 链路实现全互连，双向链路带宽为 100GB/s。这些链路构成多个环形拓扑，提供峰值为 150GB/s 的环形全归约带宽。最后，我们的软件栈采用 AMD 开源 ROCmTM 平台 5.2 [6] 版本，搭配 PyTorch v1.7、rocBLAS [5] BLAS 库，以及 RCCL [3] 通信集合库。

4.3.2 模型与训练集群配置。如表3所示，为了研究一系列 Transformer（表2、图6），我们考虑了多种超参数组合和分布式设置。

模型配置（H， B， SL）：Transformer 模型扩展通常涉及扩展H 和SL [49]。因此，我们考虑过去五年中各模型所体现的H 和SL 取值范围，并将其扩展，以预测未来五年的模型。此外，如第 3.5 节所述，超参数（H 和SL）与内存容量扩展之间的巨大间隔，迫使我们采用较小的B 和较大的TP 度数（下文称为 TP）。因此，大多数现代大型模型（例如，MT-NLG [63] 和 PALM [13]）已采用较小的B 值 1，且该值无法进一步降低。因此，我们也考虑较小的B 值。

训练集群配置（张量并行度、DP 度数）：我们还研究了一系列TP 和 DP 度数：

张量并行度：我们根据现代 Transformer 设置确定合适的 TP 范围。我们从规模最大的模型之一入手：参数量为 3.9B 的 Megatron-LM 模型（Mega.-LM_BERT）。这是首个已知的公开使用张量并行、并行度为八的 Transformer。要估算未来 Transformer 的 TP，我们必须考虑设备内存容量和模型规模。假设 Mega.-LM_BERT 需要使用八台设备的容量（=$base_TP$），那么我们根据较大模型与 Mega.-LM_BERT 的模型规模之比（p），按比例扩大 $base_TP$，以估算该模型的 TP。为考虑同一时期设备内存容量可能增加的情况，我们再将预测的 TP 除以该时期的内存容量扩展比（s）。因此，所需的张量并行度为 $base_TP * (p/s)$。图9(b)展示了 TP 扩展值（p/s），用于

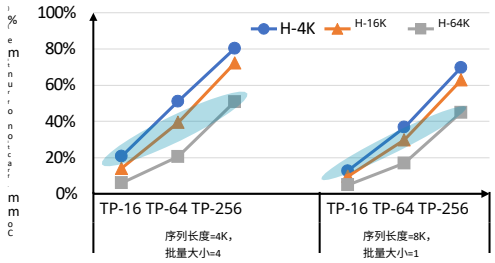


图 10：串行通信时间占比。

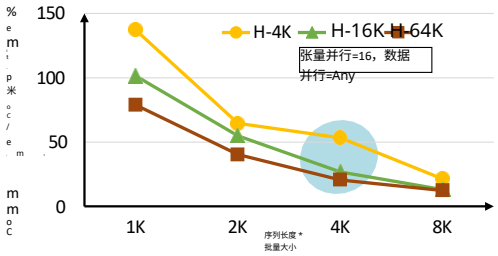


图 11：重叠通信时间占计算时间的百分比。

自 Mega.-LM_BERT 以来的 Transformer。因此，*TP* 需要扩大 40-60× 倍，所需的 *TP* 度为（∗8）~250-550。尽管随着模型规模扩大，过去几年间 *TP* 有所提升，但要实现如此大规模的 *TP*。，仍需在互连技术方面取得重大创新。此外，还可以借助流水线并行来控制 *TP*。。因此，我们研究了最高达 256 的一系列 *TP* 值。

DP 度数：我们的数据并行实证分析在很大程度上不受 *DP* 度数的影响。与张量并行不同，计算 FLOPS 和整体通信量并不由 *DP* 度数决定。此外，虽然我们采用四个图形处理器（$N = 4$）的设置，但这也为更大规模的设置提供了合理但偏保守的通信时间估计，因为在设备数量较多时，全归约（环形）流量的扩展幅度较小（$(N-1)/N \sim 1$，适用于较大的 N））。不过，设备数量增加也会提高设备间的同步开销，使实际通信时间略高。此外，数据并行训练通常部署在大规模、多节点且往往异构的系统上，使用较慢的节点间链路进行通信。由于我们无法使用这类机器，因此改用节点内链路乐观地估算通信时间，并在第 4.3.7 节讨论节点间链路的影响。

4.3.3 性能分析配置。对于重叠通信分析，如第 4.2.2 节所述，我们从训练迭代中提取相关区域（计算和通信操作），并针对所考虑的所有超参数组合，仅执行这些相关区域（表3）。尽管这些操作在实际中会并发执行，我们仍将其分开执行和研究，以避免共享资源导致的干扰减速，并了解它们各自的最佳特性。对于串行通信分析，我们首先对单个图形处理器上的 BERT [16] 训练迭代进行性能分析，作为基线。接下来，我们使用算子级模型（第 4.2.2 节）预测数百种 Transformer 配置的训练运行时间。最后，我们使用 rocProf [4] 测量 GPU 内核执行时间。

4.3.4 阿姆达尔定律优势分析。我们采用第 4.2 节所述的实证策略、第 3.5 节观察到的超参数趋势，以及第 4.3.2 节估算的张量并行值，接下来预测串行化通信相对于计算所占的比例。图 10 展示了在不同 *H*, *SL* 和 *TP* 值下，Transformer 训练时间中用于通信的比例。我们考察的 *H*, *SL* 和 *TP* 值涵盖中型 Transformer（~T-NLG [42]）、当今最大的 Transformer 模型之一（~PALM [13]），以及一个大型的未来 Transformer。

对于固定的 *H* 和 $SL * B$（一条线），通信占比会随着 *TP* 度数的增加而上升。相反，在固定 *TP* 的情况下，随着 *H* 或 *SL* 增大，通信占比会下降。这些趋势与我们的算法性结论（第 3.3 节）一致。此外，通信所占比例相当可观，并且会随着模型规模扩大而增加。根据我们在第 4.3.2 节中的讨论，不同规模的模型需要不同的 *TP* 值进行训练。对于 $H = 4K$ 的模型（例如 T-NLG），*TP* 度数为 16 可能就足够了；但对于更大的模型，则必须提高该值（例如，在 PALM-1x 中，*H* 为 16K 时，*TP* 为 64）。图 10 用蓝色标出了这些参数组合，显示通信占比会随着模型规模扩大而上升——对于 $H = 64K$（PALM-3x）模型，通信时间可占执行时间的 50%，相当可观。这一趋势也与我们的算法性结论（第 3.3 节）一致：在 *SL* 保持不变且 *H* 和 *TP* 的扩展幅度相近时，$(H + SL)/TP$ 的分母增长幅度更大，导致计算的阿姆达尔定律优势减弱。

4.3.5 松弛优势分析。利用我们的实证策略（第 4.2 节）和超参数趋势（第 3.5 节），我们还估算了通信与计算重叠的时间比例。这有助于估算计算松弛优势隐藏通信成本的程度，以及这种松弛如何扩展。此外，无论 *DP* 的程度如何，这些估算都成立。图 11 显示，随着 *SL* 与*B* 的乘积（图 11 中的 $SL * B$）增大，重叠时间会减少，这与我们在第 3.4 节得出的算法层面结论相似。此外，在较小的 *H* 中，计算与通信重叠的比例更高，导致剩余松弛更少。我们的算法分析未考虑这一点，因为这是硬件执行带来的现象。较小的 *H* 以及由此产生的较小通信量，无法像较大规模那样充分利用设备的网络带宽容量。因此，通信成本呈次线性增长，直到网络带宽饱和。然而，计算操作规模足够大，能够使计算 FLOPS 达到饱和。因此，较小规模下较慢的通信会产生更大的重叠，并留下更少的计算松弛。

此外，通信重叠比例通常很高；在 *H*, *SL* 和 *B* 取值范围内，固定 *TP* 度数为 16 时，该比例介于 17% 到 140% 之间，且不受 *DP* 度数影响。具体而言，蓝色高亮区域显示，对于常见的 $SL * B$ 值 4K（涵盖多种模型），通信耗时占计算时间的 20%–55%，留给计算的余量较小。此外，这一比例很可能高得多，因为重叠的通信通常发生在大型多节点配置中，其网络链路速度比我们研究的高带宽节点内链路更慢。

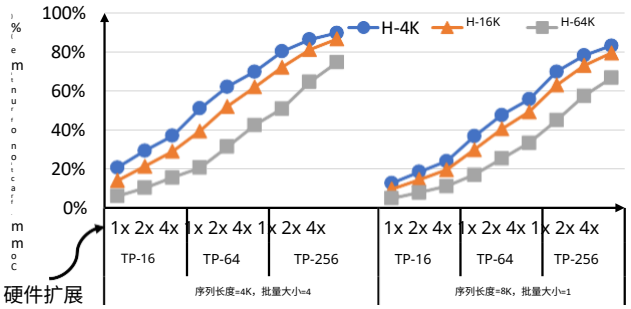


图12：硬件演进对串行通信时间占比的影响。

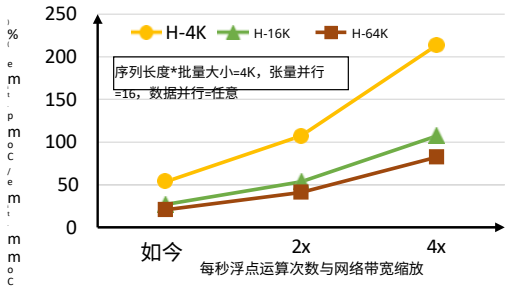


图13：硬件演进对重叠通信的影响，以计算时间的百分比表示。

4.3.6 未来硬件分析。到目前为止，我们估算了在当前系统上训练 Transformer 时计算与通信的成本。然而，硬件不断演进可能会改变这些估算，并转移应用瓶颈。因此，接下来我们根据过去的硬件趋势，估算未来系统中的计算与通信成本，以帮助指导未来系统设计。为此，我们首先估算计算 FLOPS 相对于网络带宽的变化比例，并将其称为 FLOPS 与带宽之比。该比值因 GPU 世代和厂商而异。2018 年至 2020 年间，计算 FLOPS 分别增长了 ~5× [43, 45] 和 ~7× [2, 7]，而相应的网络带宽仅增长了 ~2× [43, 45] 和 ~1.7× [2, 7]。这意味着计算 FLOPS 的增长相对网络带宽快了 ~ 2 – 4×。我们使用这些 FLOPS 与带宽的相对比值，对第 4.3.4 节和第 4.3.5 节估算的计算时间进行缩放，并预测由此产生的松弛优势以及相对于通信的阿姆达尔定律优势。请注意，降低精度会进一步使计算 FLOPS 的增长幅度远超网络带宽，从而使该比值大幅提高。我们将在第 6.2 节进一步讨论这一点。

如图12所示，在 2× 和 4× 每秒浮点运算次数与带宽缩放下，对于第4.3.4节中的配置，串行通信开始主导 Transformer训练的执行，其占比范围分别从20-50%上升至30-65%和40-75%。同样，计算加速也会减少甚至消除用于重叠通信的计算余量。图13显示，在 2× 和 4× 每秒浮点运算次数与带宽缩放下，重叠的通信量相当于计算时间的50-100%和80-210%；在许多情况下（当 >= 100%时），通信会暴露出来（即处于关键路径上）。此外，在节点间配置中，这些通信占比还会增加（见第4.3.7节）。因此，如果硬件演进继续呈现类似趋势，通信将成为Transformer训练中的关键瓶颈。

4.3.7 端到端计算与通信对比案例研究：结合串行化与重叠通信。接下来，我们研究对一个大型未来 Transformer 模型同时采用张量并行和数据并行的综合影响（如图14所示）。我们发现，47% 的时间用于串行化通信，9% 的时间用于重叠通信。由于后者完全被独立的反向传播通用矩阵乘法计算所隐藏，因此最终处于关键路径上的通信占比为 47%。

较低的节点间通信带宽和干扰导致的减速也会影响重叠通信（~8× [53]）。前者之所以相关，是因为数据并行的重叠通信可能需要通过节点间链路传输。后者之所以相关，是因为并发执行时，计算与持续时间较长的通信可能相互干扰，从而导致通信减速[53] 。图14中的第三种情形考虑了这些影响。图中显示，面向数据并行的通信不再能够完全隐藏。因此，随着面向张量并行的通信被串行化，而面向数据并行的通信仅部分重叠，总通信将成为未来 Transformer 训练中更大的瓶颈。

4.3.8 评估算子级模型。研究所有模型/配置和系统设置下计算与通信的实际硬件执行情况，可以更准确地分析通信相对于计算的扩展情况。尽管可以通过感兴趣区域提取来研究数据并行中的重叠通信，但要找到端到端模型分解以评估串行化通信，成本可能很高或不切实际（设置和执行所有模型可能很繁琐）。因此，我们使用算子模型（第4.3.8节）对端到端训练进行建模，并评估这种方法的有效性和优势：

准确性：为评估算子级模型的有效性，我们分别扫描超参数和数据大小，将运算及通信的归一化预测运行时间与在图形处理器上测得的时间进行比较。图15(a)显示，随着 *SL* 扫描，按线性比例缩放 Transformer通用矩阵乘法的运行时间能够准确捕捉其行为。同样，该图还显示，随着层宽度或 *H* 的增加，按二次比例缩放通用矩阵乘法的运行时间能够捕捉其变化趋势。总体而言，模型预测通用矩阵乘法运行时间的误差为 ~ 15%。同样，图15(b)显示，我们能够准确地对层归一化的运行时间进行建模；随着 *SL* 和 *H* 变化，其运行时间呈线性变化（几何平均误差仅为 ~ 7%）。最后，图15(c)显示，在扫描归约数据大小时，我们能够准确地建模全归约通信时间的变化趋势（几何平均误差仅为 ~ 11%）。不过，运行时间的个别误差，尤其是在使用较小运算规模进行预测时，可能并不总是很小。这是因为算子效率会随着规模增大而提升（例如，由于每秒浮点运算次数、内存或网络利用率更高），因此其运行时间不一定会如预期般增长。通用矩阵乘法等复杂运算还会针对不同规模采用不同的内核实现，这可能妨碍其随参数理想地线性或二次缩放。虽然使用更大的基线模型（从而使用更大的运算规模）或许能减小误差，但图15中的趋势仍然成立。因此，我们的关键结论不受影响。

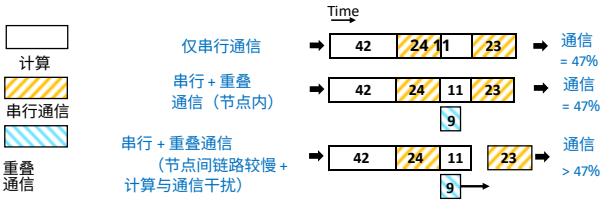


图14：计算与通信对比案例研究总览。设置：H=64K、批量大小 B=1、序列长度 SL=4K、张量并行度=128、浮点运算与带宽缩放比例=4倍。

性能分析加速：最后，无需实际执行数百种不同配置（这些配置由表3中的参数值组合而成），只需使用算子级模型以及单个基线模型的 Transformer 执行时间分解，即可进行全面研究，同时节省大量性能分析时间和精力。具体而言，我们的策略避免了执行 ~198 个不同的 Transformer模型（其中一些执行成本很高），使性能分析开销降低了三个数量级以上（2100×）。此外，为估算重叠计算与通信开销，我们无需执行端到端迭代，具体来说，无需执行前向传播。这使性能分析速度提升了 1.5× 倍。因此，我们的算法分析和实证策略为我们节省了大量性能分析时间和资源。

5 加速通信

我们的分析表明，通信正逐渐成为分布式训练中一个显著的瓶颈。因此，我们预计，未来的系统设计者将特别关注峰值计算能力与网络带宽的扩展比例 [52]：鉴于通信在总运行时间中所占比例不断上升，若没有其他深度神经网络创新，网络能力的扩展速度将与计算能力相当，甚至更快。

然而，即使网络带宽按适当比例扩展，通信仍面临其他挑战。现代系统中的通信通常由加速器自身负责协调，这会（a）消耗计算资源，并且（b）访问加速器内存；两者都可能代价高昂，并造成内存争用。此外，由于通信抽象的存在，通信集合操作（例如全归约）的实现与应用中的其他计算或集合通信相互独立，从而阻碍了并行化和重叠等优化。因此，下面我们将讨论一些有望应对这些挑战的技术和即将出现的新技术，以及仍需进一步研究的挑战。

技术1 – 通信卸载：一些技术会将通信从加速器（如图形处理器）卸载到专用协处理器（例如 ASIC、FPGA、DPU）[9, 53]，这些协处理器专门用于加速通信原语。此外，这些加速器还能释放或减少加速器用于任何独立计算的计算、缓存和内存资源。不过，这些方法会增加编程复杂度，也会增加芯片面积、功耗和碳排放成本。

技术二 – 网络内处理 (PIN)：同样，PIN 可增强现有网络交换机，使其执行集合通信 [23, 32, 38]。因此，它能带来2× 有效网络带宽方面的优势。对于全归约，加速器只需将数据发送到交换机

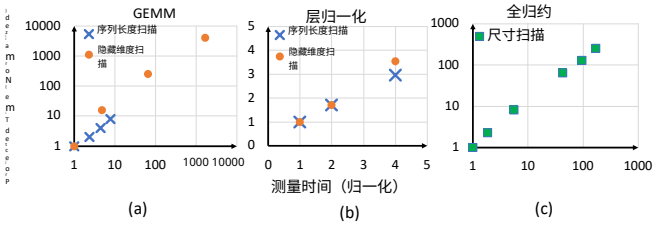


图15：算子级建模的有效性。

交换机对数据进行归约，并将结果返回给加速器。相比之下，带宽最优环形全归约 [10] 会通过网络传输两倍的数据。不过，PIN 仅适用于配备交换机的拓扑结构，而这类拓扑并不常见。

技术3 – 并行计算与通信：虽然上述技术能够加速通信，但任何串行化通信都可能导致设备吞吐量扩展不佳。因此，打破通信抽象并针对数据生成与通信的流水线处理和重叠进行优化的技术，对于扩展大语言模型至关重要 [20, 30, 72]。此类技术可以打破通信与计算（或其他通信原语）之间的粗粒度屏障，从而启动细粒度通信，使其与数据生成（例如计算）并行进行，并在很大程度上隐藏通信开销。不过，如果通信并且计算位于同一加速器上。

未来技术– 存内处理（PIM）：为应对计算对内存带宽日益增长的需求，存内处理（PIM）是一种很有前景的范式。PIM 的不同实现形式将计算单元布置得更靠近内存，从而使这些单元获得比加速器更高的内存带宽。近期已出现多种 PIM 商业实现 [59, 62]，，展现出这一范式的潜力。总体而言，PIM 有助于通信。例如，支持存内原子操作可以减少全归约原语中归约计算所需的内存流量。这也能减少在加速器上执行的通信与计算之间的内存干扰。更广泛地说，有效融合计算与通信，并结合 PIM 等技术，有助于克服上述挑战，同时不受拓扑结构和额外硅片开销的限制。

6 讨论

此处我们将讨论我们的计算与通信分析和/或方法论如何适用于其他机器学习场景和设置。

6.1 超越数据与张量并行

鉴于数据并行和张量并行在分布式训练中十分常见且无处不在，我们的分析主要聚焦于它们。不过，我们的分析也可以轻松扩展，以考察其他分布式技术带来的通信。

6.1.1 混合专家（MoE）。随着模型不断扩展，其计算成本也在迅速增加。因此，近年来，混合专家（MoE）已成为一种广受欢迎的范式，能够在不显著牺牲准确性的情况下降低计算成本。混合专家模型采用条件计算：与稠密机器学习模型不同，后者会针对所有输入激活整个网络；

混合专家模型根据输入，以稀疏方式激活大型网络中的子网络。这使混合专家模型能够在不按比例增加计算成本的情况下提升模型容量。因此，基于Transformer的混合专家模型[19, 51]在保持甚至超越稠密模型准确性的同时降低计算成本，展现出巨大的潜力。

我们提出的分析适用于基于Transformer的混合专家模型。与其稠密模型对应体类似，混合专家模型依赖数据并行和张量并行，而我们对此进行了深入研究。此外，混合专家模型还采用专家并行，这会在关键路径上增加更多串行化通信（全交换）。我们针对串行化通信提出的算法分析（第3节）和实证策略（第4节），均可扩展以纳入专家并行所引发的通信。

最后，混合专家模型每个输入所需的计算量也可能逐渐减少。这一优化进一步提高了混合专家模型中通信所占的比重，也进一步凸显了我们工作的核心论点：未来的系统设计必须仔细考量通信及其加速方式。

6.1.2 流水线并行。流水线并行将模型沿水平方向拆分为多个分区，并将每个分区分配给不同的设备。因此，该技术会进一步增加关键路径上的通信，因为必须在流水线分区之间传递输入和输出。与混合专家模型类似（如上所述），虽然可以将这类通信纳入我们的分析，但出于若干原因，本文不考虑流水线并行。首先，常规流水线并行通常会在计算过程中产生空闲气泡，可通过微批处理等技术加以解决[28]。而微批处理又需要较大的批大小，从而降低内存效率并损害训练收敛。因此，我们专注于更为普遍的数据并行与张量并行相结合的分布式训练方式。

6.1.3 大容量系统内存。以往的技术[52, 56]会将模型状态（参数、激活值等）放在系统内存（例如，CPU连接的DDR内存、NVMe内存）中，而不只是依赖加速器连接的内存（例如，图形处理器显存）。因此，当技术利用系统内存时，加速器内存容量就不再那么容易成为瓶颈，使得使用相同数量的加速器也能训练更大的模型。不过，这些场景既涉及加速器间通信（这是我们的关注重点），也涉及在CPU内存/NVMe内存与加速器内存之间移动数据所需的通信。此外，必须谨慎设计软件，以确保系统内存与加速器内存之间的数据移动不会落入关键路径。这颇具挑战性，因为存放在系统内存中的数据必须在使用前及时调入加速器内存[29, 57, 71]。

我们重点关注加速器间通信，因为它通常位于关键路径上（例如张量并行），并且会直接影响训练吞吐量。此外，我们研究的分布式训练技术是最常用的。最后，尽管对于给定规模的模型，可以通过使用系统内存来减少加速器间通信（以CPU与加速器之间的通信取代加速器间通信），但由于单个或少量加速器的整体计算能力有限，这样做的收益可能不大。

6.2 数字格式

深度神经网络（DNN）在训练和推理中也经常使用低精度数值格式来表示激活值和权重[41, 58]。低精度之所以流行，是因为它能降低通信和计算成本。例如，随着表示数据所用的位数减少，加速器（如GPU）的峰值计算能力通常会提高（例如，我们研究的MI210 GPU的FP16吞吐量约为FP32的4×）。同样，低精度也会减少从内存读取以及在加速器之间传输的数据字节数。不过，由于“计算与通信”的分析和方法基本不受数值格式影响，因此应同样适用于采用其他数值格式的训练。此外，虽然位数降低时计算能力可能呈二次或更快增长（例如FP16与FP32的峰值计算能力），但通信字节数只会线性增长（例如，从FP16到FP32为2×）。因此，“计算与通信”的关键结论也适用于这些其他格式。

6.3 模型推理

虽然我们重点分析训练中的计算与通信扩展性，但在此简要讨论计算与通信扩展性对推理的影响。训练包括在网络各层进行前向传播和反向传播，而推理只在网络中进行前向传播。此外，与训练不同，为了适应计算能力和能源受限的场景，推理中通常会采用量化[22]和蒸馏[27]等技术，以避免分布式计算及其伴随的通信。在必须进行分布式计算的场景中（例如混合专家推理[51]，推荐模型[46]），会采用我们已研究过的数据并行和张量并行等技术。在此类场景中，计算与通信的分析也可扩展至分布式推理。

6.4 Transformer之外

正如我们在第2.1节所述，尽管我们因Transformer的广泛应用和部署而将其作为重点，但我们提出的方法也可以迁移至其他深度神经网络，和／或扩展至其他深度神经网络。具体而言，我们关于ROI提取和算子级建模的见解可以轻松迁移至其他模型。同样，我们的工作所提出的、基于算法分析的实证策略也可以扩展至其他深度神经网络。

7 相关工作

7.1 深度神经网络特征分析

深度神经网络，尤其是Transformer，是推动系统优化的重要应用领域。因此，已有多项工作对其进行了基准测试和特性分析[1, 24, 40, 54, 69, 75–77, 77]。然而，与我们的工作不同，这些研究聚焦于单设备深度神经网络执行中的计算瓶颈，因此并未分析多设备分布式环境中产生的通信成本。相反，我们着重分析通信相对于计算操作的成本，并指出未来需要更多以通信为重点的创新。

7.2 研究并加速通信 其他工作研究和/或优化了分布式环境中的通信 [14, 32, 47, 53, 72]。然而，与我们的工作不同，这些研究没有考察通信成本以及相应优化带来的收益如何随不同的Transformer、硬件能力和分布式技术而变化。我们展示了通信成本在这些不同场景下的变化情况，以及何时会成为瓶颈。尽管有些工作 [17, 44] 考察了扫描部分超参数对吞吐量的影响，但它们要么没有深入分析其行为特征，要么仅针对单设备进行研究。

8 结论

包括 Transformer 在内的深度神经网络是一个重要的应用领域，正在推动现代硬件的系统需求。Transformer 近年来取得的快速进展，得益于以下三者之间的良性协同：（1）更高效的硬件；（2）更大的数据集；以及（3）算法进展（包括模型规模的指数级增长），而这些进展又进一步受益于更高效的硬件和更大的数据集。然而，模型及其数据集的增长速度远超内存容量的提升速度，从而威胁到这一良性循环，并使日益分布式的训练成为必要。为理解其影响，我们针对Transformer模型的计算与通信（**计算与通信**）扩展开展了多轴（算法层面、实验、硬件演进）分析。我们的算法分析提供了与系统无关的计算与通信扩展视角，并指出，尽管计算曾占据相对于通信的优势，但未来计算能力、内存容量和模型规模的发展趋势很可能很快使通信成为主导因素。随着硬件演进，我们还通过实证研究了面向未来 Transformer模型的计算与通信。通过提取特定的关注区域，并进一步对未来算子运行时间建模，我们能够以2100× 更低的性能分析成本研究数百种未来 Transformer/硬件场景。此外，我们的实验验证了，随着模型规模扩大，在分布式训练设置中，通信将发挥越来越大的作用（40-75%）。总体而言，我们的多轴分析清楚表明，需要有效扩展通信能力，使其至少达到与计算扩展相匹配的水平；同时，还存在若干必须解决的遗留挑战，才能实现未来模型的高效分布式训练。

参考文献

[1] Robert Adolf、Saketh Rama、Brandon Reagen、Gu-yeon Wei 和 David Brooks。2016年。Fathom：现代深度学习方法的参考工作负载。载于 IEEE国际工作负载特征分析研讨会（IISWC）。IEEE，美国华盛顿特区， 1 10 10 1109 2016 7581275 –。 [2] 2018 50 https://doi.org/。/IISWC。。AMD。。AMD INSTINCTTM MI 加速器。https://www.amd.com/en/products/professional-graphics/instinct-mi50。[3] AMD。2018年。AMD 的 ROCm通信集合库。“https://github.com/ROCmSoftwarePlatform/rccl/wiki”。[4] AMD。2019年。AMD ROCm性能分析器。“ https://rocmdocs.amd.com/en/latest/ROCm_ Tools/ROCm-Tools.html”。[5] AMD。2019年。AMD 的 BLAS 库。“ https://github.com/ROCmSoftwarePlatform/rocBLAS”。[6] AMD。2019年。ROCm：开放式GPU计算的新时代。“https://rocm.github.io/”。[7] AMD。2020年。AMD InstinctTM MI100 加速器。“ https://www.amd.com/en/products/server\protect\discretionary{\char\hyphenchar\font}{}{}accelerators/instinct\protect\discretionary{\char\hyphenchar\font}{}{}mi100”。

[8] AMD. 2022. AMD INSTINCTTM MI210 加速器。https://www.amd.com/en/products/server-accelerators/amd-instinct-mi210. [9] AMD. 2023. 分布式服务卡（DSC）。https://www.amd.com/system/files/documents/pensando-dsc-200-product-brief.pdf. [10] 百度. 2017. 百度 All-Reduce. https://github.com/baidu-research/baidu- allreduce.[11] Nathan Benaich 和 Ian Hogarth. 2022. 《2022 年人工智能现状报告》。https://www. stateof.ai/. [12] Tom Brown、Benjamin Mann、Nick Ryder、Melanie Subbiah、Jared D Kaplan、Prafulla Dhariwal、Arvind Neelakantan、Pranav Shyam、Girish Sastry、Amanda Askell、Sandhini Agarwal、Ariel Herbert-Voss、Gretchen Krueger、Tom Henighan、Re won Child、Aditya Ramesh、Daniel Ziegler、Jeffrey Wu、Clemens Winter、Chris Hesse、Mark Chen、Eric Sigler、Mateusz Litwin、Scott Gray、Benjamin Chess、J ack Clark、Christopher Berner、Sam McCandlish、Alec Radford、Ilya Sutskever 和 Dario Amodei。2020. 语言模型是少样本学习者。载于《神经信息处理系统进展》（NeurIPS，第33卷），H. Larochelle、M. Ranzato、R. Hadsell、M. F. Balcan 和 H. Lin（编）。Curran Associates, Inc.，美国纽约州雷德胡克，1877–1901。https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf [13] Aakanksha Chowdhery、Sharan Narang、Jacob Devlin、Maarten Bosma、Gaurav Mishra、Adam Roberts、Paul Barham、Hyung Won Chung、Charles Sutton、Sebastian Gehrmann 等。2022. Pal m：借助路径扩展语言建模。arXiv预印本 arXiv:2204.02311（2022）。[14] Meghan Cowan、Saeed Maleki、Madanlal Musuvathi、Olli Saarikivi 和 Yifan Xiong。2023. MSCCLang：微软集合通信语言。载于第28届 ACM 编程语言与操作系统架构支持国际会议论文集，第2卷。502–514。[15] Zihang Dai、Zhilin Yang、Yiming Yang、Jaime Carbonell、Quoc V. Le 和 Ruslan Salakhutdinov。2019. Transformer-XL：超越固定长度上下文的注意力语言模型。CoRR 1901.02860（2019）。[16] Jacob Devlin、Ming-Wei Chang、Kenton Lee 和 Kristina Toutanova。2019. BERT：用于语言理解的深度双向 Transformer 预训练。载于2019年北美计算语言学协会分会：人类语言技术会议（NAACL-HLT）论文集，Jill Burstein、Christy Doran 和 Thamar Solorio（编）。美国计算语言学协会， 4171 4186 10 18653 1 19 1423 –。 [17] 2021 https://doi.org/。/v /n - Dave Dice 和 Alex Kogan. . 优化 CPU 上 Transformer 的推理性能。CoRR abs/2102.06621（2021）。[18] Izzat El Hajj、Juan Gomez-Luna、Cheng Li、Li-Wen Chang、Dejan Milojicic 和 Wen-mei Hwu。2016. KLAP：通过内核启动聚合与提升优化动态并行。载于第49届 IEEE/ACM 微体系结构国际研讨会（MICRO）。 1 12 10 1109 2016 7783716 –。 [19] 2021 https://doi.org/。/MICRO. . William Fedus、Barret Zoph 和 Noam Shazeer. . Switch Transformers：以简单高效的稀疏性扩展至万亿参数模型。CoRR abs/2101 03961 2021 2101 03961 2101 03961 . ( ). arXiv: . [20] https://arxiv.org/abs/ . Yangyang Feng、Minhui Xie、Zijie Tian、Shuo Wang、Youyou Lu 和 Jiwu Shu。2023. Mobius：在商用 GPU 服务器上微调大规模模型。载于第28届 ACM 编程语言与操作系统架构支持国际会议论文集，第2卷。489–501。[21] Jan Fousek、Jiři Filipovič 和 Matuš Madzin。2011. 面向并行映射的 CUDA-GPU 内核自动融合。SIGARCH Comput. Archit. News（2011年12月），98–99。[22] Amir Gholami、Sehoon Kim、Zhen Dong、Zhewei Yao、Michael W. Mahoney 和 Kurt Keutzer。2021. 高效神经网络推理量化方法综述。https://doi.org/ 10 48550 2103 13630 . /ARXIV. . Richard L. [23] Graham、Devendar Bureddy、Pak Lui、Hal Rosenstock、Gilad Shainer、Gil Bloch、Dror Goldenerg、Mike Dubman、Sasha Kotchubievsky、Vladimir Koushnir、Lion Levi、Alex Margolin、Tamir Ronen、Alexander Shpiner、Oded Wertheim 和 Eitan Zahavi。2016. 可扩展分层聚合协议（SHArP）：一种高效数据归约的硬件架构。载于2016年第一届 HPC 通信优化国际研讨会（COMHPC）。1–10。https://doi.org/ 10 1109 2016 006 . /COMHPC. . Udit Gupta、Samuel Hsia、[24] Vikram Saraph、Xiaodong Wang、Brandon Reagen、Gu-Yeon Wei、Hsien-Hsin S. Lee、David Brooks 和 Carole-Jean Wu。2020. Deep-RecSys：用于优化端到端大规模神经推荐推理的系统。载于第47届ACM/IEEE计算机体系结构年度国际研讨会（ISCA）。 982 995 10 1109 45697 2020 00084 –。 [25] https://doi.org/ . /ISCA . . Hany Hassan Awadalla、Anthony Aue、Chang Chen、Vishal Chowdhary、Jonathan Clark、Christian Federmann、Xuedong Huang、Marcin Junczys-Dowmunt、Will Lewis、Mu Li、Shujie Liu、Tie-Yan Liu、Renqian Luo、Arul Menezes、Tao Qin、Frank Seide、Xu Tan、Fei Tian、Lijun Wu、Shuangzhi Wu、Yingce Xia、Dongdong Zhang、Zhirui Zhang 和 Ming Zhou。2018. 中文到英文的自动新闻翻译达到人类水平。[26] Kaiming He、Xiangyu Zhang、Shaoqing Ren 和 Jian Sun。2015. 用于图像识别的深度残差学习。CoRR abs/1512.03385（2015）。arXiv:1512.03385 http://arxiv.org/abs/ 1512 03385 . Geoffrey [27] Hinton、Oriol Vinyals 和 Jeff Dean。2015. 神经网络中的知识蒸馏。https://doi.org/10.48550/ARXIV.1503.02531

[28] Yanping Huang、Youlong Cheng、Ankur Bapna、Orhan Firat、Mia Xu Chen、Dehao Chen、HyoukJoong Lee、Jiquan Ngiam、Quoc V. Le、Yonghui Wu 和 Zhifeng Chen。2019. GPipe：使用流水线并行训练巨型神经网络。Curran Associates Inc., 美国纽约州雷德胡克。[29] Yuki Ito、Haruki Imai、Tung Le Duc、Yasushi Negishi、Kiyokuni Kawachiya、Ryo Matsumiya 和 Toshio Endo。2019. 面向大型神经网络的基于性能分析的核外混合方法：海报。载于第24届并行编程原理与实践研讨会论文集（华盛顿特区）*(PPoPP '19)*。美国计算机协会，美国纽约州纽约，399–400。https://doi.org/10 1145 3293883 3298790 . / . Abhinav Jangda、Jun Huang、Guodong Liu、Amir [30] Hossein Nodehi Sabet、Saeed Maleki、Youshan Miao、Madanlal Musuvathi、Todd Mytkowicz 和 Olli Saarikivi。2022. 打破分布式机器学习工作负载中计算与通信的抽象屏障。载于第 27 届 ACM 编程语言与操作系统架构支持国际会议论文集。402–416。[31] Young Jin Kim、Ammar Ahmad Awan、Alexandre Muzio、Andres Felipe Cruz Salinas、Liyang Lu、Amr Hendy、Samyam Rajbhandari、Yuxiong He 和 Hany Hassan Awadalla。2021. 面向多任务多语言模型的可扩展高效混合专家训练。https://doi.org/10 48550 2109 10465 . /ARXIV. . Benjamin Klenk、[32] Nan Jiang、Greg Thorson 和 Larry Dennison。2020. 用于加速共享内存多处理器集合通信的网络内架构。载于第47届ACM/IEEE计算机体系结构年度国际研讨会（ISCA）。IEEE、IEEE计算机学会，美国华盛顿特区，996–1009。[33] Alex Krizhevsky、Ilya Sutskever 和 Geoffrey E. Hinton。2012. 使用深度卷积神经网络进行 ImageNet 分类。载于第25届神经信息处理系统国际会议论文集——第1卷（内华达州太浩湖）*(NIPS'12)*。Curran Associates Inc., 美国，1097–1105。http://dl.acm.org/citation.cfm?id =2999134 2999257 . Zhenzhong Lan、Mingda Chen、Sebastian [34] Goodman、Kevin Gimpel、Piyush Sharma 和 Radu Soricut。2019. ALBERT：用于语言表示自监督学习的轻量级 BERT。载于第七届国际学习表示会议（ICLR）论文集。OpenReview.net，17页。[35] Dmitry Lepikhin、HyoukJoong Lee、Yuanzhong Xu、Dehao Chen、Orhan Firat、Yanping Huang、Maxim Krikun、Noam Shazeer 和 Zhifeng Chen。2020. Gshard：通过条件计算和自动分片扩展巨型模型。arXiv预印本 arXiv:2006.16668（2020）。[36] Min Lin、Qiang Chen 和 Shuicheng Yan。2013. 网络中的网络。CoRR abs/1312 4400 2013 1312 4400 1312 4400 . ( ). arXiv: . http://arxiv.org/abs/ . [37] Shih-Chieh Lin、Yunqi Zhang、Chang-Hong Hsu、Matt Skach、Md E. Haque、Lingjia Tang 和 Jason Mars。2018. 自动驾驶的架构影响：约束与加速。载于第二十三届编程语言与操作系统架构支持国际会议论文集（美国弗吉尼亚州威廉斯堡）（ASPLOS）。ACM，美国纽约州纽约，751–766。https://doi.org/10 1145 3173162 3173191 . / . Shuo Liu、Qiaoling Wang、Junyi [38] Zhang、Wenfei Wu、Qinliang Lin、Yao Liu、Meng Xu、Marco Canini、Ray CC Cheung 和 Jianfei He。2023. 面向分布式训练的网络内聚合与传输透明性。载于第28届 ACM 编程语言与操作系统架构支持国际会议论文集，第3卷。376–391。[39] Yinhan Liu、Myle Ott、Naman Goyal、Jingfei Du、Mandar Joshi、Danqi Chen、Omer Levy、Mike Lewis、Luke Zettlemoyer 和 Veselin Stoyanov。2019. RoBERTa：一种稳健优化的 BERT 预训练方法。arXiv:1907 11692 [ ] . cs.CL Peter Mattson、[40] Christine Cheng、Cody Coleman、Greg Diamos、Paulius Micikevicius、David A. Patterson、Hanlin Tang、Gu-Yeon Wei、Peter Bailis、Victor Bittorf、David Brooks、Dehao Chen、Debojyoti Dutta、Udit Gupta、Kim M. Hazelwood、Andrew Hock、Xinyuan Huang、Bill Jia、Daniel Kang、David Kanter、Naveen Kumar、Jeffery Liao、Guokai Ma、Deepak Narayanan、Tayo Oguntebi、Gennady Pekhimenko、Lillian Pentecost、Vijay Janapa Reddi、Taylor Robie、Tom St. John、Carole-Jean Wu、Lingjie Xu、Cliff Young 和 Matei Zaharia。2019. MLPerf训练基准。CoRR abs/1910.01500（2019），14页。arXiv:1910.01500 http://arxiv.org/abs/1910 01500 . Paulius Micikevicius、Dusan Stosic、Neil Burgess、Marius [41] Cornea、Pradeep Dubey、Richard Grisenthwaite、Sangwon Ha、Alexander Heinecke、Patrick Judd、John Kamalu、Naveen Mellempudi、Stuart Oberman、Mohammad Shoeybi、Michael Siu 和 Hao Wu。2022. 深度学习的 FP8格式。https://doi.org/10.48550/ ARXIV.2209.05433[42] 微软。2020. Turing-NLG：微软的170亿参数语言模型。《微软研究博客》1，8（2020）。https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/ [43] NVIDIA。2018. NVIDIA TESLA V100 GPU加速器。https: // images.nvidia.com/content/technologies/volta/pdf/tesla-volta-v100- datasheet-letter-fnl-web.pdf. [44] NVIDIA。2020. NVIDIA FasterTransformer。"https://github.com/NVIDIA/FasterTransformer/"。[45] NVIDIA。2021. NVIDIA A100 Tensor Core GPU。https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet-us-nvidia-1758950-r4-web.pdf. [46] Jongsoo Park、Maxim Naumov、Protonu Basu、Summer Deng、Aravind Kalaiah、Daya Khudia、James Law、Parth Malani、Andrey Malevich、Satish Nadathur、Juan Pino、Martin Schatz、Alexander Sidorov、Viswanath Sivakumar、Andrew Tulloch、Xiaodong Wang、Yiming Wu、Hector Yuen、Utku Diril、Dmytro Dzhulgakov、Kim Hazelwood、Bill Jia、Yangqing Jia、Lin Qiao、Vijay Rao、Nadav Rotem、Sungjoo Yoo 和 Mikhail Smelyanskiy。2018。Facebook 数据中心的深度学习推理：特征分析、性能优化与硬件影响。https://doi.org/10 48550 1811 09886 。/ARXIV。. Suchita Pati、Shaizeen Aga、Nuwan Jayasena 和 Matthew D [47] Sinclair。2022。揭开 BERT 的神秘面纱：系统设计影响。载于《2022年 IEEE 国际工作负载特征分析研讨会（IISWC）》。IEEE，296–309。[48] Alec Radford、Jeff Wu、Rewon Child、David Luan、Dario Amodei 和 Ilya Sutskever。2019。语言模型是无监督多任务学习器。《OpenAI 博客》1, 8（2019）。[49] Colin Raffel、Noam Shazeer、Adam Roberts、Katherine Lee、Sharan Narang、Michael Matena、Yanqi Zhou、Wei Li 和 Peter J. Liu。2019。借助统一的文本到文本 Transformer 探索迁移学习的极限。https: //doi.org/10 48550 1910 10683 。/ARXIV。. Colin Raffel、Noam Shazeer、Adam [50] Roberts、Katherine Lee、Sharan Narang、Michael Matena、Yanqi Zhou、Wei Li、Peter J Liu 等。2020。借助统一的文本到文本 Transformer 探索迁移学习的极限。《机器学习研究期刊》21, 140（2020），1–67。[51] Samyam Rajbhandari、Conglong Li、Zhewei Yao、Minjia Zhang、Reza Yazdani Aminabadi、Ammar Ahmad Awan、Jeff Rasley 和 Yuxiong He。2022。DeepSpeed-MoE：推进混合专家推理与训练，赋能下一代 AI 规模扩展。https://doi.org/10 48550 2201 05596 。/ARXIV。. Samyam Rajbhandari、Olatunji Ruwase、Jeff [52] Rasley、Shaden Smith 和 Yuxiong He。2021。ZeRO-Infinity：突破超大规模深度学习的 GPU 内存墙。载于《国际高性能计算、网络、存储与分析会议论文集》（密苏里州圣路易斯）（SC *'21*）。美国计算机协会，纽约州纽约，美国，论文编号 59，共 14 页。https://doi.org/10 1145 3458817 3476205 。/。Saeed Rashidi、Matthew Denton、Srinivas Sridharan、[53] Sudarshan Srinivasan、Amoghavarsha Suresh、Jade Nie 和 Tushar Krishna。2021。实现分布式深度学习训练平台中的计算与通信重叠。载于《2021年 ACM/IEEE 第48届年度国际计算机体系结构研讨会（ISCA）》。IEEE，540–553。[54] V. J. Reddi、C. Cheng、D. Kanter、P. Mattson、G. Schmuelling、C. Wu、B. Anderson、M. Breughe、M. Charlebois、W. Chou、R. Chukka、C. Coleman、S. Davis、P. Deng、G. Diamos、J. Duke、D. Fick、J. S. Gardner、I. Hubara、S. Idgunji、T. B. Jablin、J. Jiao、T. S. John、P. Kanwar、D. Lee、J. Liao、A. Lokhmotov、F. Massa、P. Meng、P. Micikevicius、C. Osborne、G. Pekhimenko、A. T. R. Rajan、D. Sequeira、A. Sirasao、F. Sun、H. Tang、M. Thomson、F. Wei、E. Wu、L. Xu、K. Yamada、B. Yu、G. Yuan、A. Zhong、P. Zhang 和 Y. Zhou。2020。MLPerf 推理基准测试。载于 ISCA。446–459。[55] Scott Reed、Konrad Zolna、Emilio Parisotto、Sergio Gomez Colmenarejo、Alexander Novikov、Gabriel Barth-Maron、Mai Gimenez、Yury Sulsky、Jackie Kay、Jost Tobias Springenberg、Tom Eccles、Jake Bruce、Ali Razavi、Ashley Edwards、Nicolas Heess、Yutian Chen、Raia Hadsell、Oriol Vinyals、Mahyar Bordbar 和 Nando de Freitas。2022。通用智能体。（2022）。https://doi.org/10.48550/ARXIV.2205.06175[56] Jie Ren、Samyam Rajbhandari、Reza Yazdani Aminabadi、Olatunji Ruwase、Shuangyan Yang、Minjia Zhang、Dong Li 和 Yuxiong He。2021。ZeRO-Offload：让十亿规模模型训练普及化。https://doi.org/10.48550/ARXIV.2101.06840[57] Minsoo Rhu、Natalia Gimelshein、Jason Clemons、Arslan Zulfiqar 和 Stephen W. Keckler。2016。VDNN：用于可扩展、内存高效神经网络设计的虚拟化深度神经网络。载于《第49届年度 IEEE/ACM 国际微体系结构研讨会》（中国台湾台北）（MICRO-49）。IEEE 出版社，论文编号 18，共 13 页。[58] Bita Rouhani、Daniel Lo、Ritchie Zhao、Ming Liu、Jeremy Fowers、Kalin Ovtcharov、Anna Vinogradsky、Sarah Massengill、Lita Yang、Ray Bittner、Alessandro Forin、Haishan Zhu、Taesik Na、Prerak Patel、Shuai Che、Lok Chand Koppaka、Xia Song、Subhojit Som、Kaustav Das、Saurabh Tiwary、Steve Reinhardt、Sitaram Lanka、Eric Chung 和 Doug Burger。2020。借助微软浮点数，在云规模下突破窄精度推理的极限。载于《第34届神经信息处理系统国际会议论文集》（加拿大不列颠哥伦比亚省温哥华）*(NIPS'20)*861，11 。Curran Associates Inc.，美国纽约州雷德胡克，论文页码。[59] 三星。2022。采用 HBM-PIM 的 GPU。https://semiconductor.samsung.com/news-events/tech-blog/samsung-electronics-semiconductor-unveils-cutting-edge-memory-technology-to-accelerate-next-generation-ai/。[60] Mohammad Shoeybi、Mostofa Patwary、Raul Puri、Patrick LeGresley、Jared Casper 和 Bryan Catanzaro。2019。Megatron-LM：使用模型并行训练数十亿参数语言模型。CoRR abs/1909.08053（2019）。arXiv:1909.08053 [cs.CL]

[61] Karen Simonyan 和 Andrew Zisserman。2014年。用于大规模图像识别的超深度卷积网络。CoRR abs/1409.1556（2014年）。arXiv:
1409 1556 1409 1556 。http://arxiv.org/abs/。SK海力士。。
[62] 2022
GDDR-PIM。
https://news.skhynix.com/sk-hynix-develops-pim-next-generation-ai-accelerator/。
[63] Shaden Smith、Mostofa Patwary、Brandon Norick、Patrick LeGresley、Samyam Rajbhandari、Jared Casper、Zhun Liu、Shrimai Prabhumoye、George Zerveas、Vijay Korthikanti 等。2022。使用 DeepSpeed 和 Megatron 训练 Megatron-Turing NLG 530B：大规模生成式语言模型。arXiv 预印本 arXiv:2201.11990（2022）。[64] Matthias Springer、Peter Wauligmann 和 Hidehiko Masuhara。2017。动态类型语言中的模块化基于数组的 GPU 计算。载于《第4届 ACM SIGPLAN 数组编程库、语言与编译器国际研讨会论文集》。48–55。[65] Yu Sun、Shuohuan Wang、Yukun Li、Shikun Feng、Hao Tian、Hua Wu 和 Haifeng Wang。2019。ERNIE 2.0：用于语言理解的持续预训练框架。CoRR 1907.12412（2019）。[66] Christian Szegedy、Wei Liu、Yangqing Jia、Pierre Sermanet、Scott Reed、Dragomir Anguelov、Dumitru Erhan、Vincent Vanhoucke 和 Andrew Rabinovich。2015。通过卷积不断深入。载于《计算机视觉与模式识别》（CVPR）。http://arxiv.org/abs/
1409 4842 。Christian Szegedy、Vincent Vanhoucke、
[67]
Sergey Ioffe、Jonathon Shlens 和 Zbigniew Wojna。2015。重新思考用于计算机视觉的 Inception 架构。CoRR abs/
1512 00567 2015 1512 00567 1512 00567 。（）。arXiv:。
[68]
http://arxiv.org/abs/。Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N. Gomez、Lukasz Kaiser 和 Illia Polosukhin。2017。注意力机制即你所需。载于《第31届神经信息处理系统国际会议论文集》（NeurIPS）。6000–6010。[69] Snehil Verma、Qinzhe Wu、Bagus Hanindhito、Gunjan Jha、Eugene B John、Ramesh Radhakrishnan 和 Lizy K John。2020。揭秘 MLPerf 训练基准测试套件。载于 ISPASS。IEEE，24–33。[70] Guibin Wang、YiSong Lin 和 Wei Yi。2010。内核融合：提升多线程 GPU 能效的有效方法。载于《2010年 IEEE/ACM 绿色计算与通信国际会议暨网络、物理与社会计算国际会议论文集》。344–350。[71] Linnan Wang、Jinmian Ye、Yiyang Zhao、Wei Wu、Ang Li、Shuaiwen Leon Song、Zenglin Xu 和 Tim Kraska。2018。Superneurons：用于深度神经网络训练的动态 GPU 显存管理。载于《第23届 ACM SIGPLAN 并行编程原理与实践研讨会论文集》（奥地利维也纳）*(PPoPP '18)*。美国计算机协会，纽约州纽约，美国，41 53 10 1145 3178487 3178491 −。
[72]
https://doi.org/。/。Shibo Wang、Jinliang Wei、Amit Sabne、Andy Davis、Berkin Ilbeyi、Blake Hechtman、Dehao Chen、Karthik Srinivasa Murthy、Marcello Maggioni、Qiao Zhang 等。2022。通过分解在大型深度学习模型中实现通信与依赖计算重叠。载于《第28届 ACM 编程语言与操作系统体系结构支持国际会议论文集》第1卷。93–106。[73] Wayne Xiong、、Xuedong Huang、Frank Seide、和 Andreas Stolcke。2017。迈向会话语音识别的人类水平。IEEE/ACM 音频、语音与语言处理汇刊 25（2017年9月），2410–2423。[74] Zhilin Yang、Zihang Dai、Yiming Yang、Jaime Carbonell、Ruslan Salakhutdinov 和 Quoc V. Le。2020。XLNet：通用自回归预训练，实现语言理解。CoRR 1906.08237（2020）。[75] Ali Hadi Zadeh、Zissis Poulos 和 Andreas Moshovos。2019。深度学习语言建模工作负载：图形处理器上的时间都花在了哪里。载于《IEEE 国际工作负载特征分析研讨会》（IISWC）。IEEE、IEEE计算机学会，美国华盛顿特区，131–142。[76] Bojian Zheng、Abhishek Tiwari、Nandita Vijaykumar 和 Gennady Pekhimenko。2018。EcoRNN：在 GPU 上高效计算 LSTM 循环神经网络训练。arXiv 预印本 arXiv:1805.08899（2018）。[77] Hongyu Zhu、Mohamed Akrout、Bojian Zheng、Andrew Pelegris、Amar Phanishayee、Bianca Schroeder 和 Gennady Pekhimenko。2018。TBD：深度神经网络训练的基准测试与分析。载于《IEEE 国际工作负载特征分析研讨会》（IISWC）。IEEE 出版社，美国华盛顿特区，共 13 页。