DenseLLM部署的并行化策略：应对应用特定的权衡与瓶颈

Burak Topcu
bvt5283@psu.edu 宾夕法尼亚州立大学 美国宾夕法尼亚州斯泰特学院

Musa Oguzhan Cim
mtc5693@psu.edu 宾夕法尼亚州立大学 美国宾夕法尼亚州斯泰特学院

Poovaiah Palangappa
Poovaiah.Palangappa@amd.com 超威半导体 美国加利福尼亚州圣何塞

Meena Arunachalam
M
eena.Arunachalam@amd.com 超威半导体 美国俄勒冈州波特兰

Mahmut Taylan Kandemir
mtk2@psu.edu 宾夕法尼亚州立大学 美国宾夕法尼亚州斯泰特学院

arXiv:2603.05692v1 [cs.DC] 5 Mar 2026

摘要

生成式人工智能领域的突破推动了大语言模型（LLM）应用的爆发式增长，这些应用的工作负载本质上由经由 Transformer 架构进行的一系列推理组成。在这一迅速扩张的生态系统中，稠密大语言模型——即每次生成词元时都会激活所有模型参数的模型——构成了先进的基于专家的变体的基础。稠密模型凭借强大的泛化能力、可扩展性、易于微调以及对多种任务的适应性，至今仍占据主导地位。在大语言模型推理系统中，性能主要由延迟、响应时间和吞吐量（即单位时间内生成的词元数）来衡量。延迟与吞吐量天然相互关联：优化其中一项往往要以牺牲另一项为代价。此外，当稠密模型参数超过设备内存容量时，批处理策略和并行策略配置至关重要，并会显著影响延迟和整体系统吞吐量。本文（i）研究了两个具有代表性的稠密大语言模型——Llama-3.1-70B 和 Llama-3.1-405B 的工作负载，尤其关注节点内并行化方案；（ii）分析输入特征、批处理和并行策略如何影响延迟灵活性以及延迟与吞吐量权衡；（iii）识别有助于指导设计决策的关键性能瓶颈，以满足服务级别协议（SLA）并维持推理质量。实证评估表明，张量并行（TP）有助于实现延迟目标，而流水线并行（PP）则更适合面向吞吐量的应用。我们指出，通过控制 TP 和 PP 的并行度进行混合使用，可以调节延迟与吞吐量之间的相互作用。

关键词

大语言模型推理、模型并行、延迟与吞吐量的相互作用

1 引言

生成式人工智能的迅速普及，尤其是通过大语言模型（LLM）实现的普及，深刻改变了现代生活的诸多方面，简化了各种现实任务，例如代码生成 [54, 61], 文档摘要 [85],对话代理 [22], 甚至临床诊断 [28, 77]。越来越多的实证证据表明，较大的模型，尤其是那些拥有更长上下文窗口的模型，更善于捕捉自然语言 [9, 30, 39, 80]固有的复杂、层次化且高度非线性的结构。尽管近期混合专家（MoE）及其他稀疏架构方面的进展，通过选择性激活部分参数提升了计算效率，但所有参数都参与每个词元生成的稠密模型仍不可或缺。它们不仅构成大规模预训练和微调的基础，也是剪枝 [20], 量化 [83], 和专家开发 [23, 43] 等下游变换的基础。此外，稠密架构固有的可扩展性、稳健性和通用性，使其持续成为现代生成式人工智能系统的基石。

当前流行的先进稠密大语言模型，例如 GPT-4 [56], 、GPT-3 [10], 、Llama3.1 [25], 、Mistral Large 2 [75], 以及部分通义千问-2.5 [62] 模型等变体，拥有数千亿个模型参数，并支持长达数万词元的上下文长度。然而，在通过大型模型为长上下文提供推理的同时为一批请求生成词元，需要相当大的内存来存储模型权重以及运行时键值（KV）缓存；该缓存会随自回归解码阶段动态增长，因而常常超出单个 GPU 设备的内存容量。例如，通过 Llama3.1-405B 模型 [25] 以 FP8 量化提供推理服务的系统，仅容纳模型权重就需要 ~405GB 内存空间；单个 GPU——即便是 MI325x [3]或 H200 [15]等尖端服务器级型号——也因物理内存不足而无法以 8 位或更高量化精度服务该模型。因此，出于推理目的部署此类模型通常需要尖端计算基础设施，这类基础设施一般配备多 GPU 或 TPU 以及它们之间可扩展的互连，这往往会使大语言模型推理服务的部署变得极为昂贵。

为应对大语言模型推理服务资源密集型的特性，近期研究提出，当 GPU 内存容量不足时，将模型参数和 KV缓存卸载至主机内存，并在需要时将其取回 [5, 70]。尽管这种方法降低了推理服务的成本，但由于主机与加速器之间引入了通信开销，该方法往往难以同时实现高吞吐量和低延迟。为进一步缓解内存压力，量化、压缩和剪枝已成为颇具前景的替代方案，使大型模型及其不断增长的 KV缓存占用能够适应更受限的内存空间。

量化 [18, 71] 降低模型参数和 KV缓存条目的数值精度（例如 FP6 和 FP4 [76]），使一个或更少的加速器即可在精度损失极小的情况下执行推理。另一方面，剪枝 [6, 50, 81] 旨在识别模型参数或 KV缓存中的某一区域，例如部分 Transformer 块或 KV缓存条目的部分片段，在对推理输出影响最小的情况下将其移除，并将精度权衡控制在可接受范围内。此外，压缩 [24, 38, 41] 通过在推理期间对模型或 KV缓存进行统一或选择性地压缩与解压缩，缓解了内存压力，但代价是增加了额外的计算开销。尽管这些技术能够有效缓解服务大型模型所面临的资源限制，但在不牺牲服务质量（QoS）的情况下，它们都无法完全满足现代大语言模型服务工作负载对严格延迟和吞吐量的要求。

相比之下，并行化技术通过将模型权重和 KV缓存分布到多个 GPU 上，缓解了内存压力，从而创造了在不损害服务质量的情况下提升延迟和吞吐量的巨大机会。常见的模型并行化策略，例如张量并行（TP） [72] 和流水线并行（PP） [29], , 采用不同的方法对大语言模型推理进行并行化，因此呈现出各异的权衡。例如，TP 和 PP 分别将内部 Transformer 层分片，并将 Transformer 块划分到多个设备上；二者都能支持部署受单设备内存限制的大型模型，降低每台设备的内存需求，并使系统能够处理较大的批次。我们认为，大语言模型推理系统中的并行化技术是实现高吞吐量或满足延迟敏感型大语言模型应用需求，同时确保推理质量的最有前景的途径。然而，不同并行化方案之间的相互作用、并行化 Transformer 工作负载所导致的不同资源使用情况、模型和各异的输入特征与批处理偏好，以及这些因素对延迟和吞吐量的综合影响，仍未得到充分探索，阻碍了面向大语言模型应用的高性能推理服务的实现。

据我们所知，尚无先前研究系统地评估各种并行策略在不同系统和应用约束下的性能，也未识别这些策略中出现的关键瓶颈。因此，本研究的首要目标是借助内部模拟器，深入分析数据 [67] 并行和模型并行技术（例如张量 [72] 并行和流水线 [29] 并行）及其混合组合所带来的机遇与挑战。在本研究中，我们：(i) 描述Transformer工作负载在不同并行化方案和深度下如何映射到配备GPU的节点系统；(ii) 在使用Llama 3.1–70B和3.1–405B模型提供推理服务时，量化不同并行化配置、批处理策略和输入特征下的延迟与吞吐量趋势；以及(iii) 找出能够实现严格延迟要求和高吞吐量的最佳并行化设置，同时诊断推理工作负载行为和系统能力方面制约进一步提升的关键局限性。最后，我们还将(iv)讨论研究发现和见解对于多节点系统、稀疏模型和基于专家的模型的更广泛适用性，以及更先进的并行化形式（例如专家并行 [69], ）如何影响此类场景下稠密模型的延迟和吞吐量目标。本文的主要**贡献**可概括如下：

- 我们展示了典型稠密大语言模型 Transformer 工作负载的内核级分解，并说明了在节点环境中，如何将 Transformer 层映射到 GPU 上，以及不同并行化方案、其混合变体和扩展深度的影响。

- 我们量化了批处理和并行化偏好，以及不同模型和输入特征所引起的延迟与吞吐量变化。为此，我们采用了具有代表性的短上下文和长上下文数据集、系统所能容纳的所有批次大小，以及涵盖混合变体且并行度各异的多种并行化策略。

- 我们分析了延迟灵活性以及延迟与吞吐量之间的权衡，以满足广泛的应用目标，并找出阻碍应用性能进一步提升的局限性。

本文其余部分安排如下：第 2 节概述大语言模型推理工作流，介绍目标评估模型，并回顾相关研究，从而说明对并行技术进行详细评估的必要性。第 3 节验证硅芯片与内部模拟器之间的执行相关性；第 4 节详细介绍并行化策略，并探讨这些策略在多 GPU 系统中的工作负载映射。第 5 节全面评估不同并行化、批处理、模型和输入数据集配置下的延迟与吞吐量。最后，第 6 节讨论我们的分析对多节点系统和基于专家的模型的适用性，第 7 节则总结研究的关键见解和揭示的前景趋势。

2 背景与相关工作

2.1 大语言模型

大语言模型（LLM）是基于海量文本语料库预训练的深度神经网络，由一系列*Transformer* 块构成。每个 Transformer 块都是核心计算单元：它通过注意力和前馈运算捕捉词元之间的上下文关系。发出推理请求后，模型分两个阶段处理请求：（i）**预填充阶段**，并行处理所有输入词元；（ii）**解码阶段**，逐个自回归生成新词元，直至生成序列结束（EoS）词元。在这两个阶段中，批次中的每个请求都会依次通过整个 Transformer 块堆栈，因为每个块都要使用前一个块的输出。

Transformer 块主要由 QKV投影、旋转位置编码（RoPE）、注意力、输出投影、残差加法与归一化层，以及前馈网络（FFN）组成；前馈网络可能包含多个 FFN 子网络（即专家）。在注意力层之前，QKV投影层会将输入嵌入转换为查询（Q）、键（K）和值（V）向量分量。注意力 层 [78] 会计算注意力分数，并将其转换为概率分布，以表示词元之间的两两相关性。随后，输出投影层会将注意力的输出映射回隐藏维度。前馈网络

块通常是单层或双层的门控前馈网络，其中线性投影之间包含非线性激活函数 [68] （基于专家的模型包含多个前馈网络 [69]）；它通过全连接深度神经网络学习复杂的上下文模式。最后，残差加法层会捕捉注意力层或前馈网络层之后的梯度变化，而归一化层则会在输出传递给前馈网络或下一个 Transformer 块之前使其保持稳定。

2.2 Llama 3.1-70B/-405B 模型概述

Llama 3.1 [25] 代表着当前开放权重大语言模型的先进水平，其性能可与领先的专有模型相媲美，同时完全开放，便于开展可复现的实验和系统研究。实际上，它由一系列预训练模型和微调模型组成，参数规模分别为 8B、70B 和 405B；在多项基准测试任务中，其服务质量与 GPT-4、GPT-3.5 和 Mixtral-8×22B Turbo 等当前领先模型相当 [16]。Llama 3.1 模型支持 128K 的上下文长度，能够理解显著更长的输入，并在多语言理解、推理、数学、编程和摘要方面展现出强大能力。表 1 总结了 70B 和 405B 版本的规格，这些规格决定了影响延迟和吞吐量的关键推理工作负载特征，同时也涉及不同的输入条件和系统配置。与 405B 版本相比，70B 版本的隐藏维度和中间维度（即前馈网络层中的升维投影）更小，注意力头和 Transformer 块也更少。

表 1：Llama 3.1-70B/-405B 的架构规格。

	70B 模型	405B模型
Transformer 块数量	80	126
隐藏维度大小	8192	16384
中间维度大小	28672	53248
注意力头数量	64	128
注意力头维度	128	128
参数总数	70B	405B

2.3 相关工作

控制延迟与吞吐量的设计参数：大多数推理系统由一台主机和一组加速器组成 [70, 84] 。

主机负责与外部组件通信，并运行大语言模型服务软件基础设施，管理推理执行的元数据和控制流，同时向加速器分发内核；互连的加速器（通常为 GPU）则执行核心推理计算。实现应用目标面临的主要挑战，源于词元生成的自回归特性，以及模型参数、KV缓存和中间激活占用的大量内存。因此，高效的内存管理、批处理策略、调度策略和并行化方案构成了推理系统设计的关键要素，会影响服务延迟和系统吞吐量。

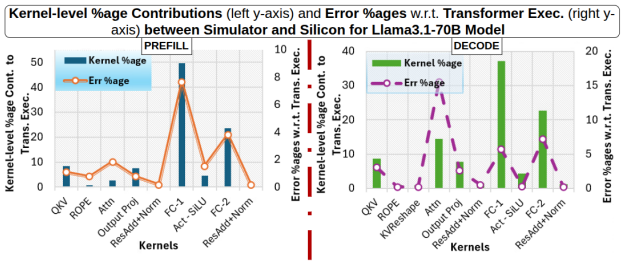
为降低大语言模型的内存需求，已有多种方法被提出，例如模型压缩 [24, 38, 41], 、量化 [18, 71], 和剪枝 [6, 50, 81],，以减小模型参数的占用以及KV缓存的占用。此外，PagedAttention [45]借鉴操作系统的分页机制，解决了内存碎片问题；vAttention [60]则进一步利用逻辑内存的连续性，最大限度地降低KV缓存运行时的内存分配开销，而KV缓存会在推理阶段不断增长。此外，ORCA [84]提出了迭代式和选择性批处理，在连续批处理的基础上采用更复杂的设计，以便在新请求到达时启动其推理。相比之下，动态批处理 [27, 58] 提供了更细粒度的资源利用率和服务等级协议控制方式，并支持面向内存和应用的批处理机会。此外，基于排序的调度等调度策略可以预测输出长度的相对排名 [21], ；推测解码 [47], 则可并行估计多个输出词元。这些机制能够调节服务等级协议并提升整体系统吞吐量。此外，DistServe [88] 重新设计了推理工作负载的系统级调度，将预填充阶段和解码阶段分配到不同的加速器上（即分离式预填充），从而消除预填充与解码之间的干扰。ChunkedPrefill [1] 专注于平衡预填充阶段和解码阶段的资源利用率；为此，它提出将较长的预填充动态交错划分为多个分块，并在交错的预填充分块之间调度解码，以提升延迟和吞吐量目标。当前的调度器（如POD-Attention和TensorRT-LLM [19, 37], ）甚至会将分块预填充和解码阶段共同纳入批次，以实现计算资源和内存的最佳协同利用。

大型模型部署与性能控制的并行化方法：数据并行(DP) [44, 67] 在每个推理服务实例上放置一份模型参数副本，不会产生加速器间通信开销，但会占用加速器内存存放模型权重，从而限制批次规模。当模型能够装入 GPU 显存，且有足够空间容纳较大的批次时，DP 可扩展系统规模，并能有效优化对延迟要求不严格的应用的系统吞吐量。张量并行 (TP) [72] 将 Transformer 层切分到多个加速器上，使各加速器并行处理同一层，并通过全归约 操作汇总 Transformer 中的部分矩阵。尽管全归约计算会带来开销，TP 仍可通过为预填充阶段和解码阶段分配更多计算单元来降低延迟，同时也能为无法装入单个加速器的模型以及更大的请求批次提供更多内存空间。相比之下，流水线并行 (PP) [29, 35] 根据并行化深度，将 Transformer 块划分到多个加速器上；每个加速器对应一个流水线阶段，并运行一组 Transformer。与 TP 在多个加速器上并发处理同一批次不同，PP 可在推理阶段让多个批次在不同流水线阶段并发处理，从而提升推理阶段的系统吞吐量，而非带来延迟收益。

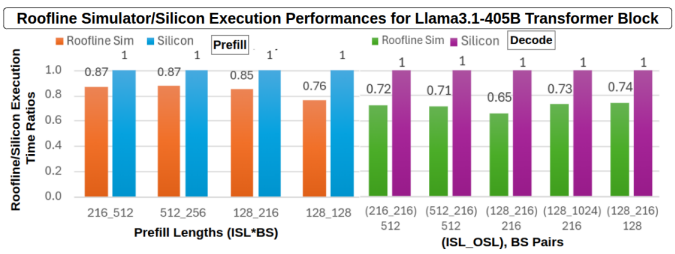
除 TP 和流水线并行等常见的模型并行方法外，已有研究还提出了其他并行化方案，尤其是上下文 [64, 82], 专 [33, 46, 63], 家或混合使用多种并行技术 [53,87], ，以解决内存瓶颈，并满足大语言模型驱动的应用的服务级别协议要求。与此同时，



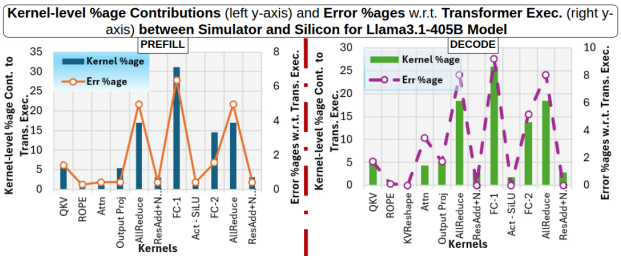
(a) 相对于硅芯片，Llama 3.1-70B模型预填充和解码的模拟器延迟归一化结果。硅芯片上的 Llama 3.1-70B模型。



(b) 各内核对 Transformer 执行的贡献，以及内核误差百分比相对于 Transformer 执行时长的占比。



(c) 相对于硅芯片，Llama 3.1-405B模型预填充和解码的模拟器延迟归一化结果。



(d) 各内核对 Transformer 执行的贡献，以及内核误差百分比相对于 Transformer 执行时长的占比。

图 1：图 (a、c) 展示了 预填充 和 解码 阶段的模拟器 Transformer 遍历时间，并将其归一化为硅芯片上的时间，适用于 *Llama 3.1-70B/-405B* 模型。图 (b) 和 (d) 分解了硅芯片上预填充和解码阶段 Transformer 块内的内核执行时间，并在右侧 y 轴上展示了各内核执行时间相对于 Transformer 执行时间的比较误差。

先前的研究，例如 [17, 48, 65], , 评估了性能一般的 GPU 互连，并发现拓扑偏好、小芯片设计、算法选择和通信协议会显著影响底层通信原语，进而影响并行化推理服务的性能。近期研究既利用现有的网络技术进展，也探索更先进的大语言模型部署并行化策略，旨在控制系统栈不同组件间的加速器间通信开销 [73, 86]。例如，Ulysses [36] 和 RingAttention [49] 采用序列并行，通过对输入序列进行分区并将各段分发到多个加速器，来处理长上下文推理所需的大型 KV缓存。TokenRing [79] 进一步改进了这些方法，有效地重叠计算与通信，从而避免计算单元闲置并提高其利用率。

现有文献大多侧重于优化训练目标，以缓解训练大规模模型所需的大量资源。然而，大语言模型训练过程涉及双向传递、动态权重更新以及不同的 KV缓存需求，从根本上不同于推理。因此，需要专门设计适用于大语言模型推理服务的并行化策略。与我们工作最相关的两项研究是：(i) DeepSpeed- I nference [5], 为推理服务提供延迟与吞吐量导向的控制；(ii) [59], 对推理延迟进行解析建模，并全面分析了在延迟约束严格的情况下，针对长序列在 TPU-v4 上进行张量并行部署的表现。尽管大语言模型驱动的应用和各种大语言模型推理服务已得到广泛采用，据我们所知，文献尚未充分探讨在不同条件下延迟灵活性以及高吞吐潜力的边界，这些条件包括不同的批次大小、输入和模型特征，以及并行化策略。我们的研究旨在填补这一空白，通过使用具有代表性的数据集，并采用不同模型规模和量化级别，全面考察并行化偏好、批次大小及序列长度变化对推理延迟和系统吞吐量的影响。

3 内部模拟器及其验证

在本研究中，我们的首要目标是分析在稠密大语言模型上为来自代表性数据集的多样化提示词提供推理服务时，不同批处理和并行化策略所涉及的延迟–灵活性权衡及延迟–吞吐量权衡。要探索这些权衡，需要全面评估数百种批处理策略、输入特征、模型与硬件配置以及并行化方案的组合。为使这项探索切实可行，我们采用内部模拟器来估算关键推理延迟指标，例如首令牌生成时间（TTFT）和每输出令牌时间（TPOT），并据此推导吞吐量值。这种方法使我们能够在实际的时间和资源限制内开展系统性的大规模实验。

我们的模拟器通过分析建模，对不同系统配置、推理模型和输入特征下的大语言模型工作负载进行端到端推理。具体而言，模拟器基于注意力和通用矩阵乘法等内核的独立表示，对 Transformer 模块进行建模，其中包括这些内核的先进版本（即分组查询注意力 [2]），并采用与实际模型架构相同的可变超参数维度。模拟器通过级联与实际大语言模型层数相同的 Transformer 层，实现了完整的表示能力。此外，它还以虚拟方式模拟配备多 GPU 的节点、集群（即多节点），

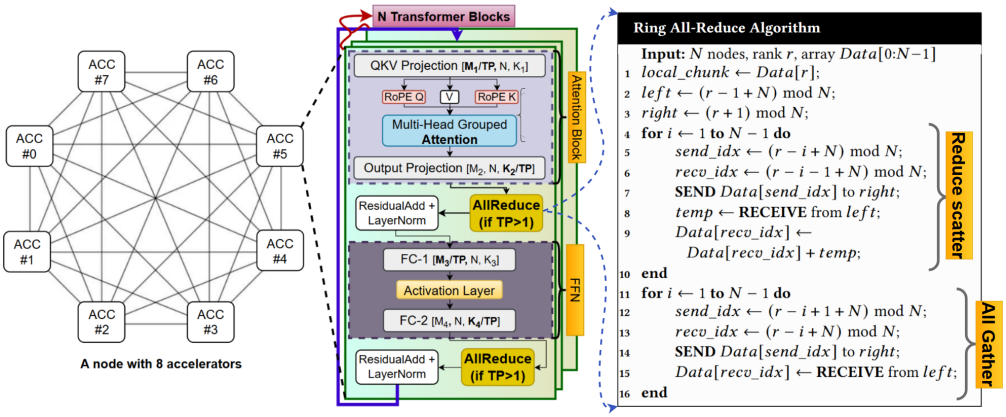


图 2：大语言模型推理中张量并行工作流程的示意概览。

节点内部及节点之间的互连，以及高层级的计算和内存层次结构，包括内存带宽和读写延迟，从而在计算受限或内存受限条件下准确模拟目标系统。

图 1a 展示了模拟器对 Llama 3.1–70B FP8 Transformer 前向传播所得的归一化执行时间（相对于 MI325x GPU [3]）。结果涵盖了多种输入序列长度（ISL）、输出序列长度（OSL）和批次大小（BS），如 x轴所示。如图所示，对于预填充和解码阶段（分别以橙色和绿色柱表示），模拟器能够准确捕捉 MI325x GPU 上的 Transformer 执行情况，准确率分别超过 90% 和 86%。此外，图 1b 展示了各个内核对 Transformer 执行的贡献，以及模拟器与实际硬件测得的 Transformer 总遍历时间之间的误差。尽管模拟器在预测单个内核运行时间时可能存在轻微误差，但这些偏差对我们的研究目的并不重要。关键在于，模拟器能够可靠地捕捉不同批处理策略、输入特征和系统配置下的整体延迟与吞吐量趋势，而这正是本研究的主要关注点和要求。

为扩展模拟器对多 GPU 上运行的更大规模模型的验证，我们针对采用 *TP8* 并行化（即并行度为 8 的张量并行）的 Llama 3.1-405B FP8 模型进行了类似的相关性分析。图 1c 和 1d 展示了在不同输入序列长度、输出序列长度和批处理配置下，预填充和解码阶段的 Transformer 执行及内核整体执行情况；比较对象是模拟器与配备八块 MI325x GPU 的节点配置，并对结果进行了归一化。对于四种不同的预填充长度，模拟器都复现了硬件上的预填充执行行为，平均误差为 17.25%，略高于单 GPU 验证中观察到的误差。此外，解码阶段的模拟精度低于预填充阶段，这反映了深度并行化带来的复杂性增加。深度并行化方案导致的轻微精度下降，可归因于 GPU 之间的动态电压频率变化。这些变化会造成内核执行差异，延迟全归约同步；此外，部分内核在实际硬件上的性能也并非最优。

此外，在两个推理阶段中，FC-1、FC-2 和全归约内核主导了 Transformer 执行，而这些主导内核的精度偏差叠加后，会降低模拟器与硅芯片端到端执行结果之间的相关性。例如，在 Llama 3.1-405B TP8 配置下预测解码推理过程时，误差范围不断扩大，主要是由 FC-1 和两个全归约内核的累积误差造成的。尽管如此，如上所述，我们的模拟器仍能非常有效地识别各种并行策略、批处理配置、不同输入条件及各类模型特征下的吞吐量和延迟趋势。

4 并行化策略

本节深入分析了我们工作中研究的并行化策略，并阐明了在采用这些策略部署大语言模型时，Transformer 层以及加速器在节点环境中的结构映射。

4.1 张量并行

图 2 展示了张量并行（TP）工作流程概览：一个节点配备 8 个加速器，并通过网格互连提供对等连接（P2P）。TP 将模型权重和注意力头均匀切分，并将 KV 缓存条目分布到各个加速器上，从而支持更大的批次，并可通过无法装入单个加速器内存的模型进行推理。在 Transformer 块中，通用矩阵乘法运算（例如 FC-1 和 FC-2）的维度为 $[M, N, K]$compute $C = \alpha AB + \beta C$，其中 α 和 β 是标量系数，$A \in R^{M \times K}$ 和 $B \in R^{K \times N}$ 是输入矩阵，输出矩阵为 $C \in R^{M \times N}$。为展示 TP 中典型的张量分片方式，我们对 QKV 投影和 FC-1 通用矩阵乘法进行切分，它们各自具有不同的维度 M，K、按行，其中 N 在预填充和解码阶段分别对应预填充长度或批量大小。在此方案中，每个加速器分别执行维度为 $[M_1/TP,\ N,\ K_1]$ 和 $[M_3/TP,\ N,\ K_3]$ 的 QKV 投影层和 FC-1 层，其中 TP 表示 TP 的深度。类似地，我们按列切分输出投影和 FC-2 通用矩阵乘法，得到的维度分别为 $[M_2, N, K_2/TP]$ 和 $[M_4, N, K_4/TP]$，如图 2 的 Transformer 块所示。

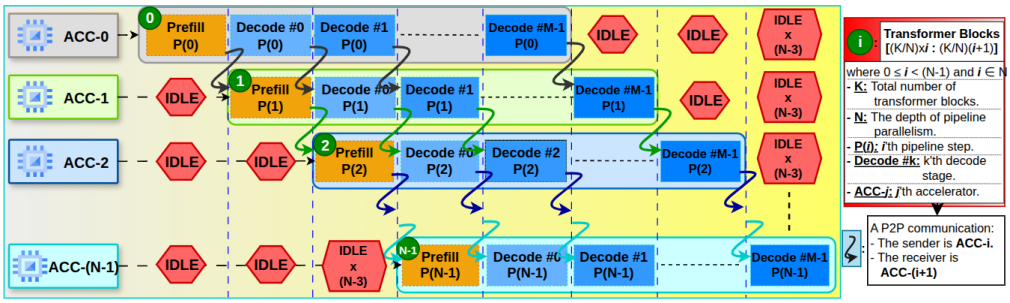


图 3：流水线并行及其在大语言模型推理中的工作流分配示意图。

如前所述，系统会将注意力头均匀分配给参与运算的加速器。例如，$TP = 4$ 将 Llama 3.1-70B 模型的 64 个注意力头分为四组，基础设施将第一组（共 16 个注意力头）分配给 ACC-0，其余 48 个注意力头分别分配给 ACC-1、ACC-2 和 ACC-3。TP 要求在残差加法层之前，通过全归约运算汇总每个加速器上注意力模块和前馈网络模块的部分输出。环形算法是 TP 中执行全归约运算的常用首选方法，如图 2 所示。该算法包含两个关键阶段——规约散射 和全收集 [26]——通过发送 & 接收 运算处理加速器间通信，并执行必要的求和（第 9 行）。因此，参与 TP 的加速器组会同步执行 Transformer 块的前向传播，为一批请求生成输出词元，如中间子图所示。

对于多个批处理请求，尽管张量并行需要承担全归约层带来的通信开销，但将注意力和前馈神经网络模块分散到各个加速器上，可以加快 Transformer 块的前向传播。对于预填充阶段这一计算受限工作负载 [88],，增大张量并行规模可缩短核心 Transformer 层的执行时间，因为规模更大的张量并行方案会调用更多加速器，从而投入更多计算资源来处理同等数量的工作负载。此外，由于模型参数经过分片，用于存放模型参数的内存空间会减少，节省出的内存可用于扩大 KV 缓存，从而在推理期间形成更大的批次。因此，张量并行能够应对解码阶段的内存受限特性，并有望更充分地利用计算单元。然而，在规模更大的张量并行方案中，参与设备间通信的加速器越多，全归约操作中涉及的单元也越多，这可能会限制扩大张量并行规模所带来的延迟收益。对于给定批量大小，尽管规模更大的张量并行部署会使每次发送和接收步骤传输的数据更少（因为通用矩阵乘法和注意力工作负载的分区更小），但这会导致规约散射和全收集步骤的循环迭代次数增加（第 4 行和第 11 行），从而增加通信步骤。因此，并行化深度、底层全归约算法、批处理、互连拓扑和链路带宽，都是决定服务不同模型和输入提示时可实现延迟上限的关键因素。

尽管张量并行为推理服务系统提供了延迟灵活性，但其对吞吐量（每秒输出词元数）的影响会因模型规模和并行深度而有所不同。举例来说，部署参数规模为405GB的Llama 3.1-405B FP8 模型的系统，对于两个数据并行 $TP = 2$系统，其KV缓存为2 x (2 x 256GB − 405GB) ≈ 214GB，而对于 $TP = 4$ 系统则为4 x 256GB − 405GB ≈ 619 GB，这些系统主要由四块MI325x GPU组成，每块具有256GB的设备内存。在所述条件下，$TP = 4$配置提供的KV缓存容量是两路数据并行 $TP = 2$ 系统的2.89倍，同时经历六次发送与接收步骤（即每次规约散射和全收集各三次），比 $TP = 2$ 多四次。此外，当以高吞吐量为目标时，$TP = 4$ 系统由于批量大小大得多，在每个通信步骤中迁移的数据量大于 $TP = 2$ 系统，从而导致显著的全规约延迟。因此，模型规模、并行度以及输入特征决定了KV缓存行为，它们是实现满足延迟与吞吐量需求的系统设计的关键。在我们的评估中，我们在节点环境中对不同规模的TP进行了延迟灵活性以及延迟与吞吐量的相互作用的实证分析。

4.2 流水线并行

与 TP 不同，TP 会沿 Transformer 层进行层内分片，例如对通用矩阵乘法和注意力核进行分片；流水线并行（PP）则根据并行化深度，将 Transformer 块分配到各个加速器上（即层间拆分）。图 3 展示了用于大语言模型推理的 PP 部署概览。假设一个采用 $PP = N$ 配置的系统，通过 K 个 Transformer 为某个稠密模型提供推理服务，其中 Transformer 块被划分为 K/N组，每组映射到一个加速器。在全部 N 个流水线阶段中，每个流水线阶段 P（i）——如图 3 所示，在 ACC-i 上运行 $i \in [0, N-1]$——负责处理跨越以下范围内 Transformer 层的请求：[（K/N）x i：（K/N）x（$i+1$）],，其中，描述分配给各流水线阶段的 Transformer 层范围的标注位于图示右侧。预填充和解码均从运行在 ACC-0 上的阶段 P（0）开始执行推理，通过范围为 [0： K/N]的 Transformer 块处理推理工作负载。阶段 P（0）完成后，服务框架会将第 K/N个块的激活值从 ACC-0 传输到 ACC-1，如图 3 中从 P（0）指向 P（1）的最左侧黑色箭头所示；随后，推理在流水线阶段 P（1）继续进行，经过跨越 [K/N：（K/N）x 2]区间的 Transformer 层。当 P（1）执行

已在 P（0）中完成的批次操作时， P（0）会开始处理一批新的请求，或为已经经过 P（$N-1$）步骤的批次生成下一个词元。通过这一流程，PP 使推理系统能够同时对多个批次进行并行流水线执行。

尽管流水线并行方案减少了参与并行化的加速器所分配到的 Transformer 块数量，但与张量并行不同，它不会影响每次 Transformer 遍历的执行时间，因为用于执行单个 Transformer 块的计算资源保持不变。相反，由于 Transformer 层采用分布式结构，它能够缓解每个加速器的内存压力，从而支持更大的 KV缓存，并使每个加速器能够形成更大的批次，进一步实现推理阶段对多个批次的同时处理。虽然更深的流水线并行配置会增加流水线阶段之间的点对点通信，但只要微批次（即每个加速器处理的一批请求）大小相同，尽管其中包含不同的请求，节点间传输的激活数据量就相同，因此不同点对点数据传输的发送和接收开销也完全一致。因此，通过在流水线阶段间分布 Transformer 层来缓解内存压力，并不能带来任何延迟收益。此外，更深的流水线并行方案可能不利于推理服务的延迟目标，因为它会使更广泛的流水线阶段之间产生更多点对点通信。

不过，通过流水线并行（PP）将 Transformer 块及其参数集合分布到各个加速器上，可以让更大的微批次在不同流水线阶段上并发运行，从而形成更大的全局批次（即系统当前正在处理的批次总数），因此有望提升吞吐量。在使用 $PP=2$ 和 $PP=4$ 设置为 Llama 3.1-405B FP8 模型提供服务时，每个加速器分别管理 ~202.5GB 和 ~101.25GB 的模型参数。因此，在采用 $PP=2$和 $PP=4$ 设置时，每个加速器分别有 256GB -202.5GB ≈ 53.5GB 和 256GB -101.25GB ≈ 154.75GB 的可用空间用于 KV缓存；两种系统均运行在四个 MI325x GPU 上。也就是说，与 $PP=2$ 方案相比， $PP=4$ 方案可用于存储大 2.89 倍的 KV缓存；在通过四个 MI325x GPU 部署时，该方案能够运行显著更大的微批次。因而，更深的 PP 方案使系统能够处理更大的批次，并通过消除内存受限的解码阶段中受限的计算利用率，对整体系统吞吐量产生积极影响。此外，PP 引入的点对点通信开销通常远小于 TP 引入的全归约内核所带来的影响，因为流水线阶段之间的传输——每次传输的代价都低于一次全归约操作——在 Transformer 之间发生 $PP_{depth}-1$ 次，而全归约内核发生的次数是 Transformer 块总数的两倍。因此，使用 PP 为稠密大语言模型提供服务，虽然由于显著增大的全局批次大小以及点对点通信带来的开销，可能会增加整体延迟，但能够提升系统吞吐量。下文将阐明节点内不同 PP 深度下的延迟与吞吐量权衡，并对 PP 与 TP 进行比较分析。

4.3 混合张量并行与流水线并行

混合 PP 和 TP 部署会将 Transformer 块分布到多个加速器集合中，每个集合负责一个流水线阶段，而 PP 的深度决定集合的总数。在每个流水线阶段内，一段 Transformer 层（其大小等于 Transformer 层总数除以 PP 深度）会在参与 TP 的加速器之间进行切分。例如，当 Llama 3.1-405B 模型采用 $TP=2$， $PP=2$ 并行化设置时，PP 阶段——$P(0)$ 和 $P(1)$——各自在两个加速器上运行，并对由 63 个 Transformer 块组成的推理进行计算；如表 1 所述，该模型共包含 126 个 Transformer 块。此外，每个 P（i）内的 $TP=2$ 设置在 $i\in\{0,1\}$ 的情况下，会将内部 Transformer 内核及其对应的张量切分为两部分，使每个 P（i）阶段中的一个加速器计算一半的内部 Transformer 内核，例如计算维度为 $[M_1/2, N, K_1]$ 的 QKV 投影通用矩阵乘法，但仅处理 63 个 Transformer 块。因此，混合使用 PP 和 TP 有望在增大 PP 深度的同时实现较高吞吐量，并保持较低的推理延迟以及 TP 的规模。我们将在第 5 节中分析不同混合 TP 和 PP 配置下的延迟与吞吐量权衡。

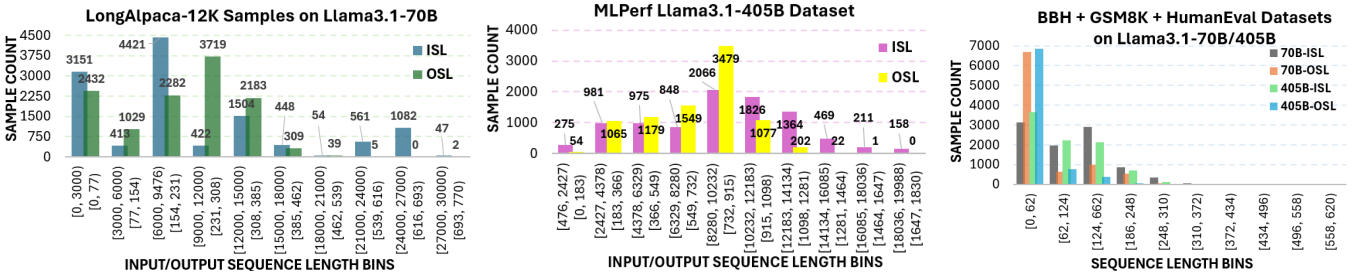
5 实验评估

5.1 实验设置

模型： [illegible] 系列，具有 8B、70B 和 405B 版本、支持最长 128k 词元上下文长度的模型，在通识知识和推理任务方面具有竞争力；这类任务需要较长的上下文记忆，例如摘要和编程 [57]。如表 1 所示，70B 和 405B 变体分别具有 64 个和 128 个查询头，且都采用分组的 8 个键值头（即分组查询注意力 [2]）；其头维度为 128，因此隐藏维度分别为 8192 和 16384。Transformer 网络的高层结构如图 2 所示，70B 和 405B 版本分别由 80 个和 126 个块组成。

数据集： 考虑到 Llama 3.1-70B 和 -405B 模型通常需要处理各种上下文长度的推理，我们使用代表长、短输入特征的数据集进行推理模拟。使用 *BIG-Bench Hard*（BBH） [74], 人工评估 [12], 和 *GSM8K* [14]数据集时，与这两种模型交互所涉及的序列长度都相对较短。BBH 数据集提供样本，用于分析高级推理、理解能力和多步逻辑能力；GSM8K 则通过高质量的小学数学题，衡量大语言模型在多步算术和推理方面的表现。人工评估提供提示词，用于评估编程任务的功能正确性。

另一方面，*LongAlpaca* [13] 和 *MLPerf* [52] 数据集在分别通过 Llama 3.1-70B 和 -405B 模型进行推理时，会产生更长的序列。LongAlpaca 用于评估模型在长上下文推理和指令遵循任务上的能力，也可用于微调模型。MLPerf 数据集从 Longbench [8], GovReport [34], 和 Ruler [31] 数据集中精心挑选了 8313 条提示，作为推理评估的基准数据集。LongAlpaca 和 MLPerf 推理数据集通过构建更长的提示词，旨在测试模型的上下文扩展



(a) 使用 Llama 的 LongAlpaca 数据集，包含 12K 个样本；(b) 使用 Llama 3.1-70B模型的 MLPerf 数据集，包含 8313 个样本；(c) 使用 Llama 3.1-405B 模型的 BBH、GSM8K 和 HumanEval 数据集组合。

图 4：Llama 3.1-70B 和 405B模型上 LongAlpaca、MLPerf 以及 BBH、GSM8K 和 HumanEval 数据集组合的输入和输出序列长度分布。

能力，这些模型具有更大的上下文窗口，可用于摘要和代码理解等现实任务。图 4 展示了各数据集的序列长度特征分布，其中 ISL 和 OSL 分别表示通过目标模型进行推理时的输入序列长度和输出序列长度。

表 2：通过目标大语言模型进行推理时，各评估数据集的平均输入和输出序列长度。

	数据集	ISL	输出序列长度	总计
70B	LongAlpaca [13]	9092	208	**9350**
	BBH、GSM8K、人工评估 [12, 14, 74]	106	26	132
405B	MLPerf [52]	9428	684	**10112**
	BBH、GSM8K、人工评估	89	20	110

我们使用 LongAlpaca，以及由 BBH、人工评估和 GSM8K 组成的组合数据集，在长、短序列输入特征下，通过 Llama 3.1-70B 模型提供推理服务，以分析并行化性能。为使用更大的模型进行类似分析，我们使用 MLPerf 和相同的组合数据集，确定不同 Llama 3.1-405B 推理设置下的延迟与吞吐量表现。在实验中，我们为每个单独的数据集代入平均输入序列长度和输出序列长度数值（如表 2 所示），以便在延迟和吞吐量分析中引入具有代表性的输入特征。

表 3：GPU 规格。

GPU	HBM Mem.	架构	精度支持	互连 Connect BW
MI325x [3]	256GB	CDNA-3	64、32、16、和 8 位	128GB/s 双向
MI355x [4]	288GB	CDNA-4	64、32、16、8 和 4 位	rect 153.6GB/s 双向

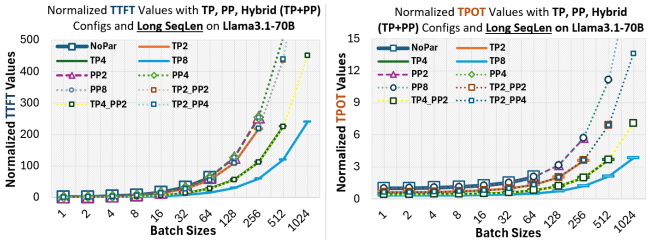
推理配置：我们的模拟仿效 AMD Instinct GPU 节点平台，该平台配备八块 MI325x [3] 或 MI355x [4] GPU，并在 GPU 对之间采用全互连。表 3 总结了目标平台的关键规格，尤其是 GPU HBM 容量和所支持的精度。有关架构拓扑和计算单元的更多详情，请参阅相应的参考数据手册。

评估指标：大语言模型驱动的应用范围广泛，带来了多样化的性能目标，从严格的延迟要求，到均衡的延迟与吞吐量权衡，再到纯粹的吞吐量优化，具体取决于工作负载涉及的是实时、交互式多用户任务，还是离线批处理 [51]。因此，首令牌时间（TTFT）和每个输出令牌时间（TPOT）指标用于评估推理延迟，而每秒总输出令牌数（TPS）指标更适合用于评估吞吐量。在这方面，我们的实验通过表 2 中列出的数据集，在分别配备 MI325x 和 MI355x GPU 的节点上，针对 8 位 Llama 3.1-70B 和 4 位 Llama 3.1-405B 量化模型模拟推理，量化了各种并行化与批处理配置下的延迟和吞吐量。

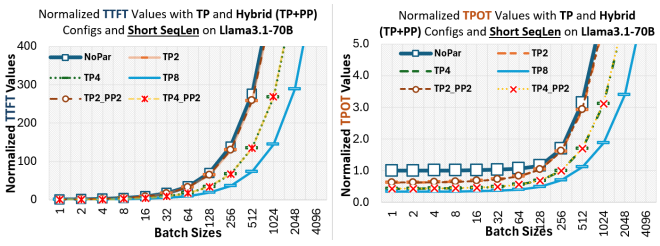
5.2 实验结果

5.2.1 延迟灵活性分析。

Llama 3.1-70B。 图 5a 和图 5b 展示了通过不同的 Llama 3.1-70B 部署配置提供推理服务时的首令牌生成时间和每输出令牌时间，包括在配备 MI325x GPU 的节点内采用的并行化方案；测试使用了分别代表长序列和短序列的两个数据集。如第 3 节所述，由于具体数值可能因推理服务软件、系统设置和内核实现而略有不同，我们以批量大小为 1 的非并行情况（标记为 NoPar）为基准，对首令牌生成时间和每输出令牌时间进行归一化，以分析延迟目标方面的趋势。如图 5a 所示，对于所有批次大小，TP8 （即 $TP = 8$ ）在首令牌生成时间和每输出令牌时间这两项延迟指标上均优于其他可行的并行化设置。同样，TP4 和 $TP4_PP2$ （即 $TP = 4$ 和 $PP = 2$）的并行化深度均低于 TP8，但与所有仅采用流水线并行、采用较低张量并行度的张量并行与流水线并行混合配置，以及非并行配置相比，它们的首令牌生成时间和每输出令牌时间都更短。因此，提高张量并行度（例如从 TP4 提高到 TP8）能够增强延迟灵活性。例如，使用批量大小 256 为 LongAlpaca 数据集提供服务时，与 TP4 和 TP2 相比，TP8 的（预填充、解码）执行速度分别快（1.87×, 1 67× 3 61× 3 01× .）和（. , .）；并行化深度低于 2 的配置无法组成包含 256 个提示词的批次，因为 KV缓存



(a) 对于 LongAlpaca 数据集。



(b) 对于 BBH、GSM8K 和人工评估数据集。

图 5：归一化的首令牌生成时间　以及通过 Llama 3.1-70B 为 (a) 和 (b) 中的数据集提供推理服务时的每输出令牌时间值。

会超出 GPU显存。在参与张量并行的 GPU 之间分片 Transformer 中的注意力和前馈网络模块，可以为每次 Transformer 遍历分配更多计算单元，从而加快推理速度，尤其是预填充阶段，因为该阶段主要受计算能力限制。而流水线并行会以层间方式将按顺序执行的 Transformer 块分布到多个 GPU 上，使多个批次能够并发处理，而不是为单次 Transformer 遍历分配更多计算资源；因此，它无法降低首令牌生成时间和每输出令牌时间。

批处理偏好和提示序列长度会影响首令牌生成时间和每输出令牌时间延迟，以及并行化方式。预填充和解码工作负载的大小（决定其执行时长）取决于预填充长度、批量大小以及模型本身；其中，预填充长度对应于批量大小 × 输入序列长度，而解码工作负载大小主要由批量大小决定。如图 2 所示，Transformer通用矩阵乘法中的维度 N 分别对应预填充阶段和解码阶段的预填充长度与批量大小。例如，较小的批次大小会直接减少预填充和解码阶段的 Transformer 遍历工作负载，从而相应缩短首令牌生成时间和每输出令牌时间；这两种延迟在图 5a 和图 5b 的 x轴上越往左越低。此外，与较小输入序列长度值对应的较短输入提示会减少预填充工作负载并降低首令牌生成时间延迟。例如，在 TP8 配置且批量大小为 256 时，为 BBH、人工评估和 GSM8K 数据集组合提供推理服务，其生成首个令牌（即预填充）的速度比为 LongAlpaca 数据集中的提示提供服务快 1.63 倍。然而，输入序列长度和输出序列长度都不会直接影响每输出令牌时间值，因为解码阶段每一步生成的词元数量等于批量大小。提示序列长度影响每输出令牌时间的间接途径是内存访问延迟：即使批量大小相同，较长的提示也需要将更大的 KV缓存加载到核心中，但它们对每输出令牌时间的影响仍然相对较小。

Llama 3.1-405B。 为了研究更大模型的延迟灵活性——与较小模型相比，这类模型可用于KV缓存的显存明显更少——我们进一步探索了使用Llama 3.1-405B模型，通过MLPerf推理数据集中的提示词提供推理服务的情况。405B参数量在4位量化下占用 ~202.5GB，可容纳于一块MI355x GPU显存中，并有 ~85GB空间可用于KV缓存。正如通过Llama 3.1-70B分析所探讨的那样，仅采用流水线并行的方案无法降低首令牌生成时间和每输出令牌时间延迟，因此我们分享了张量并行、张量并行与流水线并行混合以及不采用并行化配置下的延迟指标，如图6所示。与Llama3.1-70B推理延迟分析中的结论一致，更深的张量并行方案（例如TP4

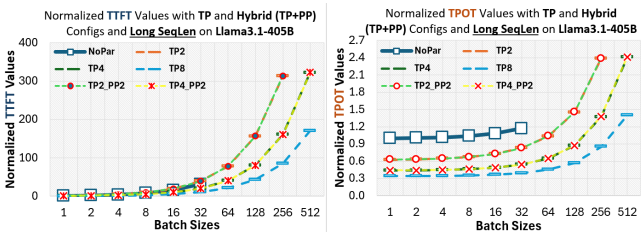
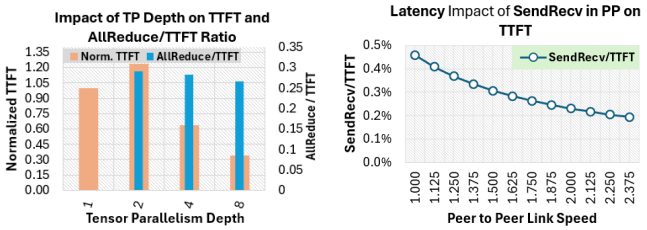


图6：采用不同并行化技术与并行深度、不同批次大小时，Llama 3.1-405B 推理服务的归一化首令牌生成时间和每输出令牌时间值。

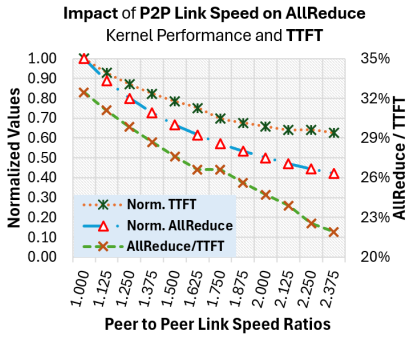
或TP8）能够大幅降低通过405B模型进行推理时的首令牌生成时间和每输出令牌时间值。对于批量大小为256的情况，需要整体并行化深度至少为2，才能容纳相应的KV缓存；与TP4、TP4_PP2和TP2相比，TP8完成预填充和解码阶段的速度分别快(1.89×, 1.61×)、(1.90×, 1.62×)和(3.67×, 2.81×)。TP4配置的延迟略优于张量并行与流水线并行混合的TP4_PP2配置，其根本原因在于流水线并行中的点对点通信会引入延迟。此外，更大的模型权重占用了有限的KV缓存空间，从而限制了允许的最大批量大小，并最终降低计算利用率。因此，与70B版本相比，405B模型拥有更多Transformer块和更大的张量内核，在相同批量大小和相近序列长度下表现出更长的首令牌生成时间和每输出令牌时间。

节点内通信的延迟开销。 为研究不同 TP 规模下节点内通信的延迟开销，我们使用 Llama 3.1-405B模型，以批量大小为 32 在节点上为 MLPerf 数据集执行推理，并测量各种 TP 配置下的首令牌生成时间和全归约时长。图 7a 分别展示了相对于 TP1（非并行）情况归一化后的首令牌生成时间和全归约时长，以及相应 TP 方案的首令牌生成时间延迟。TP8 和 TP4 方案分别将首令牌生成时间缩短了 68% 和 38%，但全归约操作的开销导致 TP2 的预填充速度比 TP1 更慢。尽管更深层的 TP 方案通过提高聚合有效带宽容量加速了全归约内核， [42], 全归约与首个令牌时间之比几乎没有变化，这表明无论 TP 规模如何，这些方案都会带来不可避免的延迟开销。此外，为分析链路速度对全归约和首令牌生成时间的影响，我们将聚合链路速度从 256 GB/s 调整至 608 GB/s，在使用相同的

模型、数据集和批量大小，但采用 TP8 配置的情况下执行推理。图 7c 展示了不同聚合链路速度下归一化后的首令牌生成时间和全归约时长（左侧 Y 轴），以及全归约与首个令牌时间之比（右侧 Y 轴）。我们发现，将链路速度提高一倍可使首令牌生成时间缩短 34%，并使全归约与首个令牌时间之比降低 7.7%。因此，在更深层的 TP 配置中，改善首令牌生成时间和每输出令牌时间的延迟，不仅取决于更快的互连，也取决于是否具备充足的计算资源。



(a) 张量并行规模与延迟之间的权衡 (b) 流水线并行中链路速度与首令牌生成时间及全归约延迟之间的权衡。



(c) 链路速度与首令牌生成时间及全归约延迟之间的关系。

图 7：张量并行规模和互连链路速度对张量并行与流水线并行中的首令牌生成时间、全归约时长及点对点通信时长的影响。

最后，我们研究通过发送和接收操作进行流水线间阶段通信对延迟的影响。图7b显示了使用配置为PP8的405B模型、批量大小为512为MLPerf 数据集提供推理服务时，点对点通信时间占首令牌生成时间（TTFT）的比例。我们发现，即使点对点链路速度为32 GB/s（即数据手册标称值的五分之一），通信对TTFT造成的延迟开销也不到0.5%。由于PP中的流水线间传输总次数为 PP_{depth} - 1次，例如PP8为7次，因此远少于全归约操作的总次数，其对TTFT和每输出令牌时间（TPOT）的延迟影响都很小。这也解释了为何在我们对Llama 3.1-70B和-405B进行延迟分析时，PP2、PP4、PP8与不采用并行化方案的情况下，TTFT和TPOT呈现出非常相似的趋势。鉴于目前所有目标物理节点系统都采用全互连网络进行GPU通信，我们对互连对延迟的影响的研究仅聚焦于链路带宽和并行化深度。因此，我们不再进一步分析不同网络拓扑可能如何影响并行化性能。

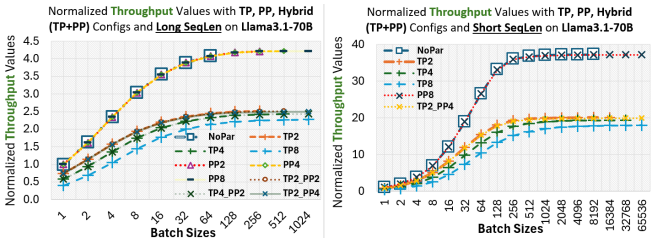
5.2.2 延迟与吞吐量的相互作用及吞吐量趋势。

图8a和图8b分别展示了搭载MI325x/MI355x GPU的节点中，采用Llama3.1-70B/-405B模型的推理系统在为LongAlpaca和MLPerf 数据集，以及合并后的BBH、GSM8K和人工评估数据集提供服务时达到的每秒令牌数（TPS）。我们根据上一节分析的延迟测量结果，使用下方公式计算每秒令牌数。

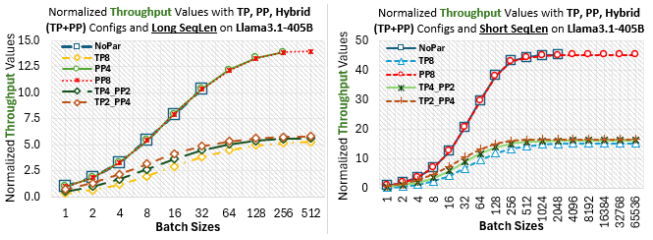
$$\text{TPS} = \frac{\text{Total Output Tokens}}{\text{Total Time (s)}} = \frac{G_{BS} \times OSL \times N_{DP-System}}{Lat_{Pref} + OSL \times Lat_{Dec}}$$

where
G_{BS} : Global batch size
OSL : Output sequence length
$N_{DP-System}$: Number of data-parallel systems
Lat_{Pref} : Average prefill latency
Lat_{Dec} : Average decode latency

采用 TP 时的吞吐量趋势。 如图8a和图8b所示，与流水线并行相比，张量并行和张量并行与流水线并行混合配置会降低每秒令牌数。尽管张量并行通过将注意力和前馈神经网络模块的计算负载分片到多个 GPU 上，缩短了这些模块的执行时间，但也引入了全归约操作的开销。虽然更大的 TP 规模通过使用更多 GPU，缩短了令牌通过 Transformer 块的时间，但对于相同的批量大小，延迟降低幅度与并行化规模并不相称，导致系统每秒令牌数下降。也就是说，由于全归约操作的开销，从无并行化配置切换到任意规模的张量并行配置时，延迟降低幅度并不与 TP 规模的增长率成反比；图7a展示了全归约操作对不同 TP 规模下首令牌生成时间的影响。



（a）左图和右图分别显示为 LongAlpaca 数据集以及 BBH、GSM8K 和人工评估数据集的组合提供推理服务时的输出每秒令牌数。



（b）为 MLPerf 推理服务时的输出每秒令牌数，以及 BBH、GSM8K 和人工评估数据集组合的输出每秒令牌数，分别如左图和右图所示。

图8：通过 Llama 3.1-70B（a）和 -405B（b）模型进行推理服务的归一化输出每秒令牌数。

流水线并行下的吞吐量趋势。 采用流水线并行方案部署 Llama 3.1-70B 和 -405B 模型，可通过将纳批大小[1] 增大至饱和水平来提升两种模型的系统吞吐量；在该水平下，解码操作与预填充一样变为纯粹的计算受限。在流水线并行中将 Transformer 分布到各流水线阶段，可减轻每块 GPU 的模型权重负担，并因 KV缓存可用空间增加而打破各解码阶段的内存受限行为，从而允许更大的纳批次，进而提高计算利用率。举例来说，在未采用任何模型并行的节点上运行的推理系统，每块 GPU 最多只能将 MLPerf 数据集中的 32 条提示词组成批次，此时所有 GPU 以数据并行方式通过 Llama 3.1-405B 模型提供推理服务；而 PP4 和 PP8 配置则分别将最大纳批大小提高到 256 和 512，如图8b所示。

尽管 PP8 支持将批量大小扩大 16 倍，但与唯一的数据并行场景（即不涉及张量并行或流水线并行）相比，其每秒令牌数仅提升 1.35× 。与批量大小的显著增长相比，每秒令牌数提升相对较小，这表明较大的批量大小会使解码工作负载进入计算受限程度更高的状态，而预填充本来就处于这种状态。若不按比例分配额外的计算资源，扩大计算受限工作负载会同时增加首令牌生成时间和每输出令牌时间，从而增大每秒令牌数公式的分母。例如，在 PP8 配置中，平均预填充时长和解码时长分别比仅采用数据并行的方案高 16.03× 和 4.24× 。此外，对于由 MLPerf 数据集表示的摘要任务，输入序列长度远大于输出序列长度，因此平均预填充延迟在推理服务中占主导地位，也主导了每秒令牌数公式的分母。因此，在服务输入序列长度远大于输出序列长度的数据集时，增加流水线并行深度和微批次大小所带来的每秒令牌数增幅，小于服务输入序列长度与输出序列长度相近的数据集时的增幅。因此，采用 PP8 方案、批量大小为 1024 的 Llama 3.1-70B，在为 LongAlpaca 数据集和组合数据集执行推理时，相较于不采用模型并行、批量大小为 1 的场景，每秒令牌数分别提高 4.2× 和 37× 。此外，采用 PP8、批量大小为 512 的 405B 版本，在 MLPerf 数据集和组合数据集上的每秒令牌数，相较于不采用模型并行、批量大小为 1 的系统，分别提升 13.8× 和 44× 。

6 讨论

更复杂模型的性能趋势。鉴于混合专家（MoE）模型日益普及，将我们的并行化策略分析扩展到涵盖这些架构将大有裨益。在混合专家 Transformer 中，位于注意力块之后的门控层会从全连接网络池中选择最适合的前 k 个专家，并将前向传播路由至这些专家。对于采用 TP 方案部署的混合专家模型，参与计算的图形处理器会在本地计算门控分数，而用于激活每个单独令牌对应专家的路由决策，则通过额外的全互连通信获得。不过，核心注意力块和专家层会在参与 TP 的 GPU 之间进行分片。因此，与稠密大语言模型一样，TP 可能为混合专家模型提供显著的延迟灵活性，尽管混合专家 Transformer 会产生额外的全互连通信开销。此外，PP 会将 Transformer 块分配到各个 GPU 上，这与混合专家模型中更新后的 Transformer 正交，因此有望提升系统吞吐量，尤其适用于几乎无法装入设备内存的模型。

多节点系统内的性能趋势。 确定模型并行技术在多节点配置中的性能趋势，将进一步有助于在更大规模上探索最优的大语言模型推理部署。 [32] 探究了节点内与节点间配置的通信性能，其中节点内各 GPU 之间具有 150 GB/s 的全互联链路，而节点间配置则依赖 DragonFly [40] 拓扑，以 25 GB/s 的带宽连接各节点。研究表明，环形全归约算法在节点内配置中的性能快 6-8× 。因此，将张量并行规模扩大到多节点规模，预计会显著放大全归约内核对首令牌生成时间和每输出令牌时间的延迟贡献，使节点间通信成为大语言模型推理的主要延迟瓶颈，正如训练中的情况 [7]。同样，多节点系统中的流水线并行会增加流水线阶段之间所经历的延迟，从而对延迟目标产生负面影响。此外，升高的推理延迟会增大吞吐量公式的分母，由于多节点系统中与模型并行相关的通信负担加重，会略微损害系统吞吐量。

新颖见解。熟悉第 4 节并行化部署讨论的读者可以推断，TP 提供了延迟灵活性，而 PP 提高了吞吐量，却没有延迟收益。除了这些直观的推论之外，我们还观察到，TP 将更多计算资源分配给一次 Transformer 遍历，使首令牌生成时间几乎呈线性下降；但由于预填充相比解码更加依赖计算，TP 对每输出令牌时间的改善幅度较小。对于更大的 TP 规模，全归约会成为这两项延迟指标上的显著延迟瓶颈，尤其是在多节点场景下。我们还发现，随着解码在更深层次的并行化下变得完全受计算限制，系统吞吐量会达到饱和，因此 PP 带来的吞吐量扩展有限。其根本原因在于，与 TP 不同，PP 会保持分配给一次 Transformer 遍历的计算资源不变，而是允许增加每 GPU 批次大小。然而，在完全受计算限制的推理工作负载中使用更大的批次，会使 PP 配置下的首令牌生成时间和每输出令牌时间延迟按比例增加，导致输出每秒令牌数达到饱和。

7 结语

本文通过在内部模拟器中配置多种推理设置，全面量化了大语言模型驱动的应用目标——延迟和吞吐量——在现有并行化策略、其混合用法及不同深度、批处理偏好和序列长度下的表现。模拟器部署了用于核心数据集的稠密大语言模型——Llama 3.1-70B 和 405B——涵盖长短序列。我们的分析得出结论：

[1]全局批次大小指系统当前主动服务的提示词总数，而纳批次大小和微批次大小则分别表示推理阶段每 GPU 与每节点的批次。

与流水线并行（PP）和不采用模型并行的方案相比，张量并行（TP）显著降低了首令牌生成时间（TTFT）和每输出令牌时间（TPOT）的延迟，而更深的张量并行还可进一步提升延迟调节的灵活性。我们还展示了不同张量并行度下，TP 所引入的全归约内核对 TTFT 的影响，并揭示了聚合链路速度变化时的延迟表现。此外，我们的分析表明，在所有批次大小下，PP 的吞吐量都高于 TP；在较大的批次下，其吞吐量也超过了唯一的数据并行基线，这主要得益于其更大的 KV缓存容量，而流水线间通信所增加的延迟远低于 TP 引入的全归约操作。因此，不同规模的 TP 可灵活满足延迟目标，而 PP 更适用于面向吞吐量的场景。由此，混合使用 TP 和 PP 可实现可调节的平衡：通过张量并行度控制延迟，通过流水线并行深度扩展吞吐量。

参考文献

[1] Amey Agrawal、Ashish Panwar、Jayashree Mohan、Nipun Kwatra、Bhargav S. Gulavani 和 Ramachandran Ramjee。2023年。SARATHI：通过将解码与分块预填充相结合，实现高效的大语言模型推理。arXiv:2308.16369 [cs.LG] https://arxiv.org/abs/2308 16369 。Joshua Ainslie、James Lee-Thorp、Michiel de Jong、[2] Yury Zemlyanskiy、Federico Lebrón 和 Sumit Sanghai。2023年。GQA：从多头检查点训练广义多查询变换器模型。arXiv:2305.13245 [cs.CL] https://arxiv.org/abs/2305 13245 。AMD。AMD InstinctTM MI325x。[3] 2024 https://www.amd.com/en/products/ accelerators/instinct/mi300/mi325x.html 访问日期：2025年5月6日。[4] AMD。2025年。AMD InstinctTM MI350X GPU。https://www.amd.com/en/products/ accelerators/instinct/mi350/mi350x.html。访问日期：2025年9月2日。[5] Reza Yazdani Aminabadi、Samyam Rajbhandari、Ammar Ahmad Awan、Cheng Li、Du Li、Elton Zheng、Olatunji Ruwase、Shaden Smith、Minjia Zhang、Jeff Rasley 和 Yuxiong He。2022年。DeepSpeed-inference：在前所未有的规模下实现变换器模型的高效推理。载于国际高性能计算、网络、存储与分析大会论文集（得克萨斯州达拉斯）*(SC '22)*。IEEE出版社，第46篇文章，15页。[6] Sotiris Anagnostidis、Dario Pavllo、Luca Biggio、Lorenzo Noci、Aurelien Luc- chi 和 Thomas Hofmann。2024年。面向高效且可解释的自回归变换器的动态上下文剪枝。arXiv:2305.15805 [cs.CL] https://arxiv.org/abs/2305 15805 。Quentin Anthony、Benjamin Michalowicz、Jacob Hatef、[7] Lang Xu、Mustafa Abdul- jabbar、Aamir Shafi、Hari Subramoni 和 Dhabaleswar Panda。2024年。揭示分布式变换器模型的通信特征。arXiv:2408.10197 [cs.DC] https://arxiv.org/abs/2408.10197[8] Yushi Bai、Xin Lv、Jiajie Zhang、Hongchang Lyu、Jiankai Tang、Zhidian Huang、Zhengxiao Du、Xiao Liu、Aohan Zeng、Lei Hou、Yuxiao Dong、Jie Tang 和 Juanzi Li。2024年。LongBench：用于长上下文理解的双语多任务基准。载于计算语言学协会第*62*届年会论文集（第*1*卷：长篇论文），Lun-Wei Ku、Andre Martins 和 Vivek Srikumar（编）。计算语言学协会，泰国曼谷，3119 3137 10 18653 1 2024 172 – 。https://doi.org/ 。/v / .[9] acl-long。Tom B. Brown、Benjamin Mann、Nick Ryder、Melanie Subbiah、Jared Kaplan、Prafulla Dhariwal、Arvind Neelakantan、Pranav Shyam、Girish Sastry、Amanda Askell、Sandhini Agarwal、Ariel Herbert-Voss、Gretchen Krueger、Tom Henighan、Rewon Child、Aditya Ramesh、Daniel M. Ziegler、Jeffrey Wu、Clemens Winter、Christopher Hesse、Mark Chen、Eric Sigler、Mateusz Litwin、Scott Gray、Benjamin Chess、Jack Clark、Christopher Berner、Sam McCandlish、Alec Radford、Ilya Sutskever 和 Dario Amodei。2020年。语言模型是少样本学习者。arXiv:2005 14165 [ ] 2005 14165 . cs.CL https://arxiv.org/abs/ 。Tom [10] B. Brown、Benjamin Mann、Nick Ryder、Melanie Subbiah、Jared Kaplan、Prafulla Dhariwal、Arvind Neelakantan、Pranav Shyam、Girish Sastry、Amanda Askell、Sandhini Agarwal、Ariel Herbert-Voss、Gretchen Krueger、Tom Henighan、Rewon Child、Aditya Ramesh、Daniel M. Ziegler、Jeffrey Wu、Clemens Winter、Christopher Hesse、Mark Chen、Eric Sigler、Mateusz Litwin、Scott Gray、Benjamin Chess、Jack Clark、Christopher Berner、Sam McCandlish、Alec Radford、Ilya Sutskever 和 Dario Amodei。2020年。语言模型是少样本学习者。arXiv:2005 14165 [ ] 2005 14165 . cs.CL https://arxiv.org/abs/ 。[11] Weilin Cai、Juyong Jiang、Fan Wang、Jing Tang、Sunghun Kim 和 Jiayi Huang。2025年。大语言模型中混合专家技术综述。*IEEE*知识与数据工程汇刊（2025年），1–20。https://doi.org/10.1109/ tkde.2025.3554028

[12] Mark Chen、Jerry Tworek、Heewoo Jun、Qiming Yuan、Henrique Ponde de Oliveira Pinto、Jared Kaplan、Harri Edwards、Yuri Burda、Nicholas Joseph、Greg Brockman、Alex Ray、Raul Puri、Gretchen Krueger、Michael Petrov、Heidy Khlaaf、Girish Sastry、Pamela Mishkin、Brooke Chan、Scott Gray、Nick Ryder、Mikhail Pavlov、Alethea Power、Lukasz Kaiser、Mohammad Bavarian、Clemens Winter、Philippe Tillet、Felipe Petroski Such、Dave Cummings、Matthias Plappert、Fo- tios Chantzis、Elizabeth Barnes、Ariel Herbert-Voss、William Hebgen Guss、Alex Nichol、Alex Paino、Nikolas Tezak、Jie Tang、Igor Babuschkin、Suchir Balaji、Shantanu Jain、William Saunders、Christopher Hesse、Andrew N. Carr、Jan Leike、Josh Achiam、Vedant Misra、Evan Morikawa、Alec Radford、Matthew Knight、Miles Brundage、Mira Murati、Katie Mayer、Peter Welinder、Bob McGrew、Dario Amodei、Sam McCandlish、Ilya Sutskever 和 Wojciech Zaremba。2021年。《评估基于代码训练的大语言模型》。arXiv:2107.03374 [cs.LG] https://arxiv.org/abs/2107.03374[13] Yukang Chen、Shengju Qian、Haotian Tang、Xin Lai、Zhijian Liu、Song Han 和 Jiaya Jia。2024年。《LongLoRA：长上下文大语言模型的高效微调》。arXiv:2309.12307 [cs.CL] https://arxiv.org/abs/2309.12307[14] Karl Cobbe、Vineet Kosaraju、Mohammad Bavarian、Mark Chen、Heewoo Jun、Lukasz Kaiser、Matthias Plappert、Jerry Tworek、Jacob Hilton、Reiichiro Nakano、Christopher Hesse 和 John Schulman。2021年。《训练验证器以求解数学文字题》。arXiv:2110 14168 [ ] 2110 14168 . cs.LG [15] 2023 200 https://arxiv.org/abs/。英伟达公司。。NVIDIA H Tensor Core GPU。https://resources.nvidia.com/en-us-gpu-resources/hpc-datasheet-sc23。访问日期：2025年11月7日。[16] Richie Cotton 和 Alex Olteanu。2024年。《Meta 的 Llama 3.1 405B 是什么？》https: // www.datacamp.com/blog/llama-3-1-405b-meta-ai。访问日期：2025年9月21日。[17] Daniele De Sensi、Lorenzo Pichetti、Flavio Vella、Tiziano De Matteis、Zebin Ren、Luigi Fusco、Matteo Turisini、Daniele Cesarini、Kurt Lust、Animesh Trivedi、Duncan Roweth、Filippo Spiga、Salvatore Di Girolamo 和 Torsten Hoefler。2024年。《探索GPU间通信：深入了解超级计算机互连》。载于*SC24*：高性能计算、网络、存储与分析国际会议。IEEE，1–15。https://doi.org/10.1109/sc41406. 2024.00039[18] Tim Dettmers、Mike Lewis、Younes Belkada 和 Luke Zettlemoyer。2022年。《LLM.int8()：大规模 Transformer 的8位矩阵乘法》。arXiv: 2208 07339 [ ] 2208 07339 . cs.LG https://arxiv.org/abs/。[19] 2024 Amr Elmeleegy、Nick Comly 和 Sharan Chetlur。。借助 NVIDIA TensorRT-LLM 分块预填充，优化 AI 推理性能与部署。NVIDIA技术博客。https://developer.nvidia.com/blog/streamlining-ai-inference-performance-and-deployment-with-nvidia-tensorrt-llm-chunked- prefill/.[20] Elias Frantar 和 Dan Alistarh。2023年。《SparseGPT：大规模语言模型可通过单次操作实现精确剪枝》。arXiv:2301.00774 [cs.LG] https://arxiv.org/ abs/2301.00774[21] Yichao Fu、Siqi Zhu、Runlong Su、Aurick Qiao、Ion Stoica 和 Hao Zhang。2024年。《通过学习排序实现高效的大语言模型调度》。arXiv:2408.15792 [cs.LG] https: //arxiv.org/abs/2408 15792 。Jianfeng Gao、Michel Galley 和 Lihong Li。2019年。《对话式[22] AI 的神经网络方法》。arXiv:1809 08267 [ ] 1809 08267 . cs.CL [23] https://arxiv.org/abs/。Shangqian Gao、Ting Hua、Reza Shirkavand、Chi-Heng Lin、Zhen Tang、Zhen- gao Li、Longge Yuan、Fangyi Li、Zeyu Zhang、Alireza Ganjdanesh、Lou Qian、Xu Jie 和 Yen-Chang Hsu。2025年。《ToMoE：通过动态结构化剪枝将稠密大语言模型转换为混合专家模型》。arXiv: 2501 15316 [ ] 2501 15316 . cs.LG https://arxiv.org/abs/。Suyu Ge、Yunan Zhang、Liyuan Liu、Minjia Zhang、Jiawei [24] Han 和 Jianfeng Gao。2024年。《模型告诉你该丢弃什么：面向大语言模型的自适应 KV 缓存压缩》。arXiv: 2310 01801 [ ] 2310 01801 . cs.CL [25] https://arxiv.org/abs/。Aaron Grattafiori、Abhimanyu Dubey、Abhinav Jauhri、Abhinav Pandey、Ab- hishek Kadian、Ahmad Al-Dahle、Aiesha Letman、Akhil Mathur、Alan Schelten、Alex Vaughan、Amy Yang、Angela Fan、Anirudh Goyal、Anthony Hartshorn、Aobo Yang、Archi Mitra、Archie Sravankumar、Artem Korenev、Arthur Hinsvark、Arun Rao、Aston Zhang、Aurelien Rodriguez、Austen Gregerson、Ava Spataru、Baptiste Roziere、Bethany Biron、Binh Tang、Bobbie Chern、Charlotte Caucheteux、Chaya Nayak、Chloe Bi、Chris Marra、Chris McConnell、Christian Keller、Christophe Touret、Chunyang Wu、Corinne Wong、Cristian Canton Ferrer、Cyrus Nikolaidis、Damien Allonsius、Daniel Song、Danielle Pintz、Danny Livshits、Danny Wyatt、David Esiobu、Dhruv Choudhary、Dhruv Mahajan、Diego Garcia-Olano、Diego Perino、Dieuwke Hupkes、Egor Lakomkin、Ehab AlBadawy、Elina Lobanova、Emily Dinan、Eric Michael Smith、Filip Radenovic、Francisco Guzmán、Frank Zhang、Gabriel Synnaeve、Gabrielle Lee、Georgia Lewis Anderson、Govind That- tai、Graeme Nail、Gregoire Mialon、Guan Pang、Guillem Cucurell、Hailey Nguyen、Hannah Korevaar、Hu Xu、Hugo Touvron、Iliyan Zarov、Imanol Arrieta Ibarra、Isabel Kloumann、Ishan Misra、Ivan Evtimov、Jack Zhang、Jade Copet、Jaewon Lee、Jan Geffert、Jana Vranes、Jason Park、Jay Mahadeokar、Jeet Shah、Jelmer van der Linde、Jennifer Billock、Jenny Hong、Jenya Lee、Jeremy Fu、Jianfeng Chi、Jianyu Huang、Jiawen Liu、Jie Wang、Jiecao Yu、Joanna Bitton、Joe Spisak、Jongsoo Park、Joseph Rocca、Joshua Johnstun、Joshua Saxe、Junteng Jia、Kalyan Vasu- den Alwala、Karthik Prasad、Kartikeya Upasani、Kate Plawiak、Ke Li、Kenneth Heafield、Kevin Stone、Khalid El-Arini、Krithika Iyer、Kshitiz Malik、Kuenley Chiu、

Kunal Bhalla、Kushal Lakhotia、Lauren Rantala-Yeary、Laurens van der Maaten、Lawrence Chen、Liang Tan、Liz Jenkins、Louis Martin、Lovish Madaan、Lubo Malo、Lukas Blecher、Lukas Landzaat、Luke de Oliveira、Madeline Muzzi、Mahesh Pasupuleti、Mannat Singh、Manohar Paluri、Marcin Kardas、Maria Tsimpoukelli、Mathew Oldham、Mathieu Rita、Maya Pavlova、Melanie Kambadur、Mike Lewis、Min Si、Mitesh Kumar Singh、Mona Hassan、Naman Goyal、Narjes Torabi、Nikolay Bashlykov、Nikolay Bogoychev、Niladri Chatterji、Ning Zhang、Olivier Duchenne、Onur Çelebi、Patrick Alrassy、Pengchuan Zhang、Pengwei Li、Petar Vasic、Peter Weng、Prajjwal Bhargava、Pratik Dubal、Praveen Krishnan、Punit Singh Koura、Puxin Xu、Qing He、Qingxiao Dong、Ragavan Srinivasan、Raj Ganapathy、Ramon Calderer、Ricardo Silveira Cabral、Robert Stojnic、Roberta Raileanu、Rohan Mah- eswari、Rohit Girdhar、Rohit Patel、Romain Sauvestre、Ronnie Polidoro、Roshan Sumbaly、Ross Taylor、Ruan Silva、Rui Hou、Rui Wang、Saghar Hosseini、Sahana Chennabasappa、Sanjay Singh、Sean Bell、Seohyun Sonia Kim、Sergey Edunov、Shaoliang Nie、Sharan Narang、Sharath Raparthy、Sheng Shen、Shengye Wan、Shruti Bhosale、Shun Zhang、Simon Vandenhende、Soumya Batra、Spencer Whit- man、Sten Sootla、Stephane Collot、Suchin Gururangan、Sydney Borodinsky、Tamar Herman、Tara Fowler、Tarek Sheasha、Thomas Georgiou、Thomas Scialom、Tobias Speckbacher、Todor Mihaylov、Tong Xiao、Ujjwal Karn、Vedanuj Goswami、Vibhor Gupta、Vignesh Ramanathan、Viktor Kerkez、Vincent Gonguet、Virginie Do、Vish Vogeti、Vítor Albiero、Vladan Petrovic、Weiwei Chu、Wenhan Xiong、Wenyin Fu、Whitney Meers、Xavier Martinet、Xiaodong Wang、Xiaofang Wang、Xiaoqing Ellen Tan、Xide Xia、Xinfeng Xie、Xuchao Jia、Xuewei Wang、Yaelle Gold- schlag、Yashesh Gaur、Yasmine Babaei、Yi Wen、Yiwen Song、Yuchen Zhang、Yue Li、Yuning Mao、Zacharie Delpierre Coudert、Zheng Yan、Zhengxing Chen、Zoe Papakipos、Aaditya Singh、Aayushi Srivastava、Abha Jain、Adam Kelsey、Adam Shajnfeld、Adithya Gangidi、Adolfo Victoria、Ahuva Goldstand、Ajay Menon、Ajay Sharma、Alex Boesenberg、Alexei Baevski、Allie Feinstein、A manda Kallet、Amit Sangani、Amos Teo、Anam Yunus、Andrei Lupu、Andres Alvarado、Andrew Caples、Andrew Gu、Andrew Ho、Andrew Poulton、Andrew Ryan、Ankit Ram- chandani、Annie Dong、Annie Franco、Anuj Goyal、Aparajita Saraf、Arkabandhu Chowdhury、Ashley Gabriel、Ashwin Bharambe、Assaf Eisenman、Azadeh Yazdan、Beau James、Ben Maurer、Benjamin Leonhardi、Bernie Huang、Beth Loyd、Beto De Paola、Bhargavi Paranjape、Bing Liu、Bo Wu、Boyu Ni、Braden Hancock、Bram Wasti、Brandon Spence、Brani Stojkovic、Brian Gamido、Britt Montalvo、Carl Parker、Carly Burton、Catalina Mejia、Ce Liu、Changhan Wang、Changkyu Kim、Chao Zhou、Chester Hu、Ching-Hsiang Chu、Chris Cai、Chris Tindal、Christoph Feichtenhofer、Cynthia Gao、Damon Civin、Dana Beaty、Daniel Kreymer、Daniel Li、David Adkins、David Xu、Davide Testuggine、Delia David、Devi Parikh、Diana Liskovich、Didem Foss、Dingkang Wang、Duc Le、Dustin Holland、Edward Dowl- ing、Eissa Jamil、Elaine Montgomery、Eleonora Presani、Emily Hahn、Emily Wood、Eric-Tuan Le、Erik Brinkman、Esteban Arcaute、Evan Dunbar、Evan Smothers、Fei Sun、Felix Kreuk、Feng Tian、Filippos Kokkinos、Firat Ozgenel、Francesco Caggioni、Frank Kanayet、Frank Seide、Gabriela Medina Florez、Gabriella Schwarz、Gada Badeer、Georgia Swee、Gil Halpern、Grant Herman、Grigory Sizov、Guangyi、Zhang、Guna Lakshminarayanan、Hakan Inan、Hamid Shojanazeri、Han Zou、Han- nah Wang、Hanwen Zha、Haroun Habeeb、Harrison Rudolph、Helen Suk、Henry Aspegren、Hunter Goldman、Hongyuan Zhan、Ibrahim Damlaj、Igor Molybog、Igor Tufanov、Ilias Leontiadis、Irina-Elena Veliche、Itai Gat、Jake Weissman、James Geboski、James Kohli、Janice Lam、Japhet Asher、Jean-Baptiste Gaya、Jeff Marcus、J eff Tang、Jennifer Chan、Jenny Zhen、Jeremy Reizenstein、Jeremy Teboul、Jessica Zhong、Jian Jin、Jingyi Yang、Joe Cummings、Jon Carvill、Jon Shepard、Jonathan McPhie、Jonathan Torres、Josh Ginsburg、Junjie Wang、Kai Wu、Kam Hou U、Karan Saxena、Kartikay Khandelwal、Katayoun Zand、Kathy Matosich、Kaushik Veeraraghavan、Kelly Michelena、Keqian Li、Kiran Jagadeesh、Kun Huang、Kunal Chawla、Kyle Huang、Lailin Chen、Lakshya Garg、Lavender A、Leandro Silva、Lee Bell、Lei Zhang、Liangpeng Guo、Licheng Yu、Liron Moshkovich、Luca Wehrstedt、Madian Khabsa、Manav Avalani、Manish Bhatt、Martynas Mankus、Matan Has- son、Matthew Lennie、Matthias Reso、Maxim Groshev、Maxim Naumov、Maya Lathi、Meghan Keneally、Miao Liu、Michael L. Seltzer、Michal Valko、Michelle Restrepo、Mihir Patel、Mik Vyatskov、Mikayel Samvelyan、Mike Clark、Mike Macey、Mike Wang、Miquel Jubert Hermoso、Mo Metanat、Mohammad Raste- gari、Munish Bansal、Nandhini Santhanam、Natascha Parks、Natasha White、Navyata Bawa、Nayan Singhal、Nick Egebo、Nicolas Usunier、Nikhil Mehta、Niko- lay Pavlovich Laptev、Ning Dong、Norman Cheng、Oleg Chernoguz、Olivia Hart、Omkar Salpekar、Ozlem Kalinli、Parkin Kent、Parth Parekh、Paul Saab、Pavan Balaji、Pedro Rittner、Philip Bontrager、Pierre Roux、Piotr Dollar、Polina Zvyag- ina、Prashant Ratanchandani、Pritish Yuvraj、Qian Liang、Rachad Alao、Rachel Rodriguez、Rafi Ayub、Raghotham Murthy、Raghu Nayani、Rahul Mitra、Ran- gaprabhu Parthasarathy、Raymond Li、Rebekkah Hogan、Robin Battey、Rocky Wang、Russ Howes、Ruty Rinott、Sachin Mehta、Sachin Siby、Sai Jayesh Bondu、Samyak Datta、Sara Chugh、Sara Hunt、Sargun Dhillon、Sasha Sidorov、Satadru Pan、Saurabh Mahajan、Saurabh Verma、Seiji Yamamoto、Sharadh Ramaswamy、Shaun Lindsay、Shaun Lindsay、Sheng Feng、Shenghao Lin、Shengxin Cindy Zha、Shishir Patil、Shiva Shankar、Shuqiang Zhang、Shuqiang Zhang、Sinong Wang、Sneha Agarwal、Soji Sajuyigbe、Soumith Chintala、Stephanie Max、Stephen Chen、Steve Kehoe、Steve Satterfield、Sudarshan Govindaprasad、Sumit Gupta、Summer Deng、Sungmin Cho、Sunny Virk、Suraj Subramanian、Sy Choudhury、Sydney Goldman、Tal Remez、Tamar Glaser、Tamara Best、Thilo Koehler、Thomas Robin son、Tianhe Li、TianjunZhang、Tim Matthews、Timothy Chou、TzookShaked、Varun Vontimitta、Victoria Ajayi、Victoria Montanez、Vijai Mohan、Vinay Satish Kumar、Vishal Mangla、Vlad Ionescu、Vlad Poenaru、Vlad Tiberiu Mihailescu、Vladimir Ivanov、Wei Li、Wenchen Wang、Wenwen Jiang、Wes Bouaziz、Will Constable、Xiaocheng Tang、Xiaojian Wu、Xiaolan Wang、Xilun Wu、Xinbo Gao、Yaniv Kleinman、Yanjun Chen、Ye Hu、Ye Jia、Ye Qi、Yenda Li、Yilin Zhang、Ying Zhang、Yossi Adi、Youngjin Nam、Yu、Wang、Yu Zhao、Yuchen Hao、Yundi Qian、Yunlu Li、Yuzi He、Zach Rait、Zachary DeVito、Zef Rosnbrick、Zhaoduo Wen、Zhenyu Yang、Zhiwei Zhao 和 Zhiyu Ma。2024年。Llama 3 模型群。arXiv:2407.21783 [cs.AI] https://arxiv.org/abs/2407.21783

[26] William Gropp、Ewing Lusk与Anthony Skjellum。1999年。使用*MPI*：使用消息传递接口进行可移植并行编程 （第2版）。MIT出版社，美国马萨诸塞州剑桥。第93–98页。介绍了包括规约散射和全收集在内的集合通信操作，这些操作构成了环形全归约算法的基础。

[27] Ozgur Guldogan、Jackson Kunde、Kangwook Lee与Ramtin Pedarsani。2024年。用于提升大语言模型推理吞吐量的多箱批处理。arXiv:2412.04504 [cs.CL] https://arxiv.org/abs/2412.04504

[28] Muhammad Usman Hadi、qasem al tashi、Rizwan Qureshi、Abbas Shah、amgad muneer、Muhammad Irfan、Anas Zafar、Muhammad Bilal Shaikh、Naveed Akhtar、Jia Wu与Seyedali Mirjalili。2023年。大语言模型：对其应用、挑战、局限性与未来展望的全面综述。*Authorea*预印本 1, 3 (2023年), 1–26. https://doi.org/10.36227/techrxiv.23589741.v3

[29] Aaron Harlap、Deepak Narayanan、Amar Phanishayee、Vivek Seshadri、Nikhil Devanur、Greg Ganger与Phil Gibbons。2018年。PipeDream：快速高效的流水线并行深度神经网络训练。arXiv:1806.03377 [cs.DC] https://arxiv.org/abs/ 1806.03377

[30] Jordan Hoffmann、Sebastian Borgeaud、Arthur Mensch、Elena Buchatskaya、Trevor Cai、Eliza Rutherford、Diego de Las Casas、Lisa Anne Hendricks、Johannes Welbl、Aidan Clark、Tom Hennigan、Eric Noland、Katie Millican、George van den Driessche、Bogdan Damoc、Aurelia Guy、Simon Osindero、Karen Simonyan、Erich Elsen、Jack W. Rae、Oriol Vinyals与Laurent Sifre。2022年。训练计算最优的大语言模型。arXiv:2203.15556 [cs.CL] https://arxiv.org/abs/ 2203 15556 .

[31] Cheng-Ping Hsieh、Simeng Sun、Samuel Kriman、Shantanu Acharya、Dima Rekesh、Fei Jia、Yang Zhang与Boris Ginsburg。2024年。RULER：你的长上下文语言模型的真实上下文规模是多少？ arXiv:2404.06654 [cs.CL] https://arxiv.org/abs/2404.06654

[32] Zhiyi Hu、Siyuan Shen、Tommaso Bonato、Sylvain Jeaugey、Cedell Alexander、Eric Spada、James Dinan、Jeff Hammond与Torsten Hoefler。2025年。揭秘NCCL：对GPU通信协议与算法的深入分析。arXiv: 2507 04786 [ ] 2507 04786 . cs.DC https://arxiv.org/abs/ .

[33] Haiyang Huang、Newsha Ardalani、Anna Sun、Liu Ke、Hsien-Hsin S. Lee、Shruti Bhosale、Carole-Jean Wu与Benjamin Lee。2024年。迈向高效的混合专家推理。载于神经信息处理系统进展，A. Globerson、L. Mackey、D. Belgrave、A. Fan、U. Paquet、J. Tomczak与C. Zhang（编），第37卷。Curran Associates, Inc.，84033–84059。

[34] Luyang Huang、Shuyang Cao、Nikolaus Parulian、Heng Ji与Lu Wang。2021年。面向长文档摘要的高效注意力。载于*2021*年计算语言学协会北美分会会议：人类语言技术论文集，Kristina Toutanova、Anna Rumshisky、Luke Zettlemoyer、Dilek Hakkani-Tur、Iz Beltagy、Steven Bethard、Ryan Cotterell、Tanmoy Chakraborty与Yichao Zhou（编）。计算语言学协会，在线，1419–1436。https://doi.org/10.18653/v1/2021.naacl-main.112

[35] Yanping Huang、Youlong Cheng、Ankur Bapna、Orhan Firat、Mia Xu Chen、Dehao Chen、HyoukJoong Lee、Jiquan Ngiam、Quoc V. Le、Yonghui Wu与Zhifeng Chen。2019年。GPipe：使用流水线并行高效训练巨型神经网络。arXiv: 1811 06965 [ ] 1811 06965 . cs.CV

[36] https://arxiv.org/abs/ . Sam Ade Jacobs、Masahiro Tanaka、Chengming Zhang、Minjia Zhang、Shuaiwen Leon Song、Samyam Rajbhandari与Yuxiong He。2023年。DeepSpeed Ulysses：支持超长序列Transformer模型训练的系统优化。arXiv: 2309 14509 [ ] 2309 14509 . cs.LG

[37] https://arxiv.org/abs/ . Aditya K. Kamath、Ramya Prabhu、Jayashree Mohan、Si mon Peter、Ramachandran Ramjee与Ashish Panwar。2025年。*POD-Attention*：释放完整的预填充-解码重叠以加速大语言模型推理。计算机协会，纽约州纽约市，美国，897–912。https://doi.org/10.1145/3676641.3715996

[38] Hao Kang、Qingru Zhang、Souvik Kundu、Geonhwa Jeong、Zaoxing Liu、Tushar Krishna与Tuo Zhao。2024年。GEAR：一种用于大语言模型近无损生成式推理的高效KV缓存压缩方案。arXiv:2403.05527 [cs.LG] https://arxiv.org/abs/ 2403 05527 . Jared Kaplan、Sam

[39] McCandlish、Tom Henighan、Tom B. Brown、Benjamin Chess、Rewon Child、Scott Gray、Alec Radford、Jeffrey Wu与Dario Amodei。2020年。神经语言模型的缩放定律。arXiv:2001.08361 [cs.LG] https: //arxiv.org/abs/ 2001 08361 . John Kim、

[40] Wiliam J Dally、Steve Scott与Dennis Abts。2008年。技术驱动的高可扩展蜻蜓拓扑。*ACM SIGARCH*计算机体系结构新闻 36, 3 (2008年), 77–88.

[41] Sehoon Kim、Coleman Hooper、Amir Gholami、Zhen Dong、Xiuyu Li、Sheng Shen、Michael W. Mahoney与Kurt Keutzer。2024年。SqueezeLLM：稠密与稀疏量化。arXiv: 2306 07629 [ ] 2306 07629 . cs.CL [42] 2025 https://arxiv.org/abs/ . Jayacharan Kolla、Pedram Alizadeh与Gilbert Lee。. 理解AMD InstinctTM MI300X上的RCCL带宽与xGMI性能。 https://rocm.blogs.amd.com/software-tools-optimization/mi300x- rccl-xgmi/README.html 访问日期：2025年11月28日。[43] Aran Komatsuzaki、Joan Puigcerver、James Lee-Thorp、Carlos Riquelme Ruiz、Basil Mustafa、Joshua Ainslie、Yi Tay、Mostafa Dehghani与Neil Houlsby。2023年。稀疏升级：从稠密检查点训练混合专家。arXiv: 2212 05055 [ ] 2212 05055 . cs.LG https://arxiv.org/abs/ . Alex [44] 2014 Krizhevsky。. 并行化卷积神经网络的一个奇特技巧。arXiv:1404.5997 [cs.NE] https://arxiv.org/abs/1404.5997[45] Woosuk Kwon、Zhuohan Li、Siyuan Zhuang、Ying Sheng、Lianmin Zheng、Cody Hao Yu、Joseph Gonzalez、Hao Zhang与Ion Stoica。2023年。使用PagedAttention实现大语言模型服务的高效内存管理。载于第*29*届操作系统原理研讨会论文集 （德国科布伦茨）（*SOSP '23*）。计算机协会，纽约州纽约市，美国，611–626。 https://doi.org/10.1145/3600006.3613165[46] Dmitry Lepikhin、HyoukJoong Lee、Yuanzhong Xu、Dehao Chen、Orhan Firat、Yanping Huang、Maxim Krikun、Noam Shazeer与Zhifeng Chen。2020年。GShard：利用条件计算与自动分片扩展巨型模型。arXiv: 2006 16668 [ ] 2006 16668 . cs.CL https://arxiv.org/abs/ . Yaniv [47] 2023 Leviathan、Matan Kalman与Yossi Matias。. 通过推测解码实现Transformer的快速推理。 arXiv:2211.17192 [cs.LG] https://arxiv. org/abs/ 2211 17192 . Ang Li、Shuaiwen [48] Leon Song、Jieyang Chen、Jiajia Li、Xu Liu、Nathan R. Tallent与Kevin J. Barker。2020年。评估现代GPU互连：PCIe、NVLink、NV-SLI、NVSwitch与GPUDirect。*IEEE*并行与分布式系统汇刊 31, 1 (1月 2020 94 110 10 1109 2019 2928289), – . https://doi.org/ . / [49] 2023 TPDS. . Hao Liu、Matei Zaharia与Pieter Abbeel。. 面向近无限上下文的环形注意力与分块 Transformer。arXiv:2310.01889 [cs.CL] https://arxiv. org/abs/ 2310 01889 . Bo Lv、 [50] Quan Zhou、Xuanang Ding、Yan Wang与Zeming Ma。2024年。KVPruner：面向更快且更节省内存的大语言模型的结构化剪枝。arXiv: 2409 11057 [ ] 2409 11057 . cs.CL https://arxiv.org/abs/ . [51] 2025 MLCommons。. MLPerf推理：数据中心基准测试。https:// mlcommons.org/benchmarks/inference-datacenter/。访问日期：2025年10月9日。[52] MLCommons。2025年。使用LLAMA3.1-405b进行文本摘要。https://docs. mlcommons.org/inference/benchmarks/language/get-llama3_1-405b-data/。访问日期：2025年10月9日。[53] Deepak Narayanan、Mohammad Shoeybi、Jared Casper、Patrick LeGresley、Mostofa Patwary、Vijay Anand Korthikanti、Dmitri Vainbrand、Prethvi Kashinkunti、Julie Bernauer、Bryan Catanzaro、Amar Phanishayee与Matei Zaharia。2021年。使用Megatron-LM在GPU集群上高效开展大规模语言模型训练。arXiv:2104.04473 [cs.CL] https://arxiv.org/abs/2104.04473[54] Erik Nijkamp、Bo Pang、Hiroaki Hayashi、Lifu Tu、Huan Wang、Yingbo Zhou、Silvio Savarese与Caiming Xiong。2023年。CodeGen：一个支持多轮程序综合的开放代码大语言模型。arXiv:2203.13474 [cs.LG] https://arxiv.org/abs/ 2203 13474 . 英伟达公司。. TensorRT-LLM后端——模型并行（张量并行、 [55] 2025 流水线并行与专家并行）。https://docs.nvidia.com/deeplearning/triton- inference-server/user-guide/docs/tensorrtllm_backend/README.html#model- parallelism。访问日期：2025年9月15日。[56] OpenAI、Josh Achiam、Steven Adler、Sandhini Agarwal、Lama Ahmad、Ilge Akkaya、Florencia Leoni Aleman、Diogo Almeida、Janko Altenschmidt、Sam Altman、Shyamal Anadkat、Red Avila、Igor Babuschkin、Suchir Balaji、Valerie Balcom、Paul Baltescu、Haiming Bao、Mohammad Bavarian、Jeff Belgum、Irwan Bello、Jake Berdine、Gabriel Bernadett-Shapiro、Christopher Berner、Lenny Bogdonoff、Oleg Boiko、Madelaine Boyd、Anna-Luisa Brakman、Greg Brockman、Tim Brooks、Miles Brundage、Kevin Button、Trevor Cai、Rosie Campbell、Andrew Cann、Brittany Carey、Chelsea Carlson、Rory Carmichael、Brooke Chan、Che Chang、Fotis Chantzis、Derek Chen、Sully Chen、Ruby Chen、Jason Chen、Mark Chen、Ben Chess、Chester Cho、Casey Chu、Hyung Won Chung、Dave Cummings、Jeremiah Currier、Yunxing Dai、Cory Decareaux、Thomas Degry、Noah Deutsch、Damien Deville、Arka Dhar、David Dohan、Steve Dowling、Sheila Dunning、Adrien Ecoffet、Atty Eleti、Tyna Eloundou、David Farhi、Liam Fedus、Niko Felix、Simón Posada Fishman、Juston Forte、Isabella Fulford、Leo Gao、Elie Georges、Christian Gibson、Vik Goel、Tarun Gogineni、Gabriel Goh、Rapha Gontijo-Lopes、Jonathan Gordon、Morgan Grafstein、Scott Gray、Ryan Greene、Joshua Gross、Shixiang Shane Gu、Yufei Guo、Chris Hallacy、Jesse Han、Jeff Harris、Yuchen He、Mike Heaton、Johannes Heidecke、Chris Hesse、Alan Hickey、Wade Hickey、Peter Hoeschele、Brandon Houghton、Kenny Hsu、Shengli Hu、Xin Hu、Joost Huizinga、Shantanu Jain、Shawn Jain、Joanne Jang、Angela Jiang、Roger Jiang、Haozhun Jin、Denny Jin、Shino Jomoto、Billie Jonn、Heewoo Jun、Tomer Kaftan、Łukasz Kaiser、Ali Kamali、Ingmar Kanitscheider、Nitish Shirish Keskar、Tabarak Khan、Logan Kilpatrick、Jong Wook Kim、Christina Kim、Yongjik Kim,

Jan Hendrik Kirchner、Jamie Kiros、Matt Knight、Daniel Kokotajlo、Łukasz Kondraciuk、Andrew Kondrich、Aris Konstantinidis、Kyle Kosic、Gretchen Krueger、Vishal Kuo、Michael Lampe、Ikai Lan、Teddy Lee、Jan Leike、Jade Leung、Daniel Levy、Chak Ming Li、Rachel Lim、Molly Lin、Stephanie Lin、Mateusz Litwin、Theresa Lopez、Ryan Lowe、Patricia Lue、Anna Makanju、Kim Malfacini、Sam Manning、Todor Markov、Yaniv Markovski、Bianca Martin、Katie Mayer、Andrew Mayne、Bob McGrew、Scott Mayer McKinney、Christine McLeavey、Paul McMil- lan、Jake McNeil、David Medina、Aalok Mehta、Jacob Menick、Luke Metz、Andrey Mishchenko、Pamela Mishkin、Vinnie Monaco、Evan Morikawa、Daniel Mossing、Tong Mu、Mira Murati、Oleg Murk、David Mély、Ashvin Nair、Reiichiro Nakano、Rajeev Nayak、Arvind Neelakantan、Richard Ngo、Hyeonwoo Noh、Long Ouyang、Cullen O'Keefe、Jakub Pachocki、Alex Paino、Joe Palermo、Ashley Pantuliano、Giambattista Parascandolo、Joel Parish、Emy Parparita、Alex Passos、Mikhail Pavlov、Andrew Peng、Adam Perelman、Filipe de Avila Belbute Peres、Michael Petrov、Henrique Ponde de Oliveira Pinto、Michael、Pokorny、Michelle Pokrass、Vitchyr H. Pong、Tolly Powell、Alethea Power、Boris Power、Elizabeth Proehl、Raul Puri、Alec Radford、Jack Rae、Aditya Ramesh、Cameron Raymond、Francis Real、Kendra Rimbach、Carl Ross、Bob Rotsted、Henri Roussez、Nick Ryder、Mario Saltarelli、Ted Sanders、Shibani Santurkar、Girish Sastry、Heather Schmidt、David Schnurr、John Schulman、Daniel Selsam、Kyla Sheppard、Toki Sherbakov、Jessica Shieh、Sarah Shoker、Pranav Shyam、Szymon Sidor、Eric Sigler、Maddie Simens、Jordan Sitkin、Katarina Slama、Ian Sohl、Benjamin Sokolowsky、Yang Song、Na- talie Staudacher、Felipe Petroski Such、Natalie Summers、Ilya Sutskever、Jie Tang、Nikolas Tezak、Madeleine B. Thompson、Phil Tillet、Amin Tootoonchian、Eliza- beth Tseng、Preston Tuggle、Nick Turley、Jerry Tworek、Juan Felipe Cerón Uribe、Andrea Vallone、Arun Vijayvergiya、Chelsea Voss、Carroll Wainwright、Justin Jay Wang、Alvin Wang、Ben Wang、Jonathan Ward、Jason Wei、CJ Weinmann、Akila Welihinda、Peter Welinder、Jiayi Weng、Lilian Weng、Matt Wiethoff、Dave Willner、Clemens Winter、Samuel Wolrich、Hannah Wong、Lauren Workman、Sherwin Wu、Jeff Wu、Michael Wu、Kai Xiao、Tao Xu、Sarah Yoo、Kevin Yu、Qiming Yuan、Wojciech Zaremba、Rowan Zellers、Chong Zhang、Marvin Zhang、Shengjia Zhao、Tianhao Zheng、Juntang Zhuang、William Zhuk 和 Barret Zoph。2024年。GPT-4 技术报告。arXiv: 2303 08774 [ ] 2303 08774 。cs.CL [57] 2025 https://arxiv.org/abs/ 。Litu Ou 和 Mirella Lapata。。面向长文档摘要的上下文感知层次合并。 arXiv:2502.00977 [cs.CL] https://arxiv.org/abs/2502. 00977[58] Bowen Pang、Kai Li 和 Feifan Wang。2025年。通过内存感知和服务级别协议约束的动态批处理优化大语言模型推理吞吐量。 arXiv:2503.05248 [cs.DC] https://arxiv.org/abs/2503.05248[59] Reiner Pope、Sholto Douglas、Aakanksha Chowdhery、Jacob Devlin、James Bradbury、Jonathan Heek、Kefan Xiao、Shivani Agrawal 和 Jeff Dean。2023年。高效扩展 Transformer 推理。载于机器学习与系统论文集，D. Song、M. Carbin 和 T. Chen（编），第5卷。Curan，606–624。 https://proceedings.mlsys.org/paper_files/paper/2023/file/ c4be71ab8d24cdfb45e3d06dbfca2780-Paper-mlsys2023.pdf[60] Ramya Prabhu、Ajay Nayak、Jayashree Mohan、Ramachandran Ramjee 和 Ashish Panwar。2025年。 vAttention：无需 PagedAttention 的大语言模型服务动态内存管理。载于第*30*届*ACM*编程语言与操作系统架构支持国际会议论文集，第*1*卷（荷兰鹿特丹）（*ASPLOS '25*）。计算机协会，纽约州纽约市，美国，1133–1150。https: //doi.org/10.1145/3669940.3707256[61] Rui Qian、Xiaoyi Dong、Pan Zhang、Yuhang Zang、Shuangrui Ding、Dahua Lin 和 Jiaqi Wang。2024年。利用大语言模型理解长视频流。arXiv:2405.16009 [cs.CV] https://arxiv.org/abs/2405.16009[62] 通义千问、:、An Yang、Baosong Yang、Beichen Zhang、Binyuan Hui、Bo Zheng、Bowen Yu、Chengyuan Li、Dayiheng Liu、Fei Huang、Haoran Wei、Huan Lin、Jian Yang、Jianhong Tu、Jianwei Zhang、Jianxin Yang、Jiaxi Yang、Jingren Zhou、Junyang Lin、Kai Dang、Keming Lu、Keqin Bao、Kexin Yang、Le Yu、Mei Li、Mingfeng Xue、Pei Zhang、Qin Zhu、Rui Men、Runji Lin、Tianhao Li、Tianyi Tang、Tingyu Xia、Xingzhang Ren、Xuancheng Ren、Yang Fan、Yang Su、Yichang Zhang、Yu Wan、Yuqiong Liu、Zeyu Cui、Zhenru Zhang 和 Zihan Qiu。2025年。Qwen2.5 技术报告。arXiv:2412.15115 [cs.CL] https://arxiv.org/abs/2412.15115[63] Samyam Rajbhandari、Conglong Li、Zhewei Yao、Minjia Zhang、Reza Yazdani Aminabadi、Ammar Ahmad Awan、Jeff Rasley 和 Yuxiong He。2022年。DeepSpeed-MoE：推进混合专家推理与训练，助力下一代人工智能规模化发展。 arXiv:2201.05596 [cs.LG] https://arxiv.org/abs/2201.05596[64] Nir Ratner、Yoav Levine、Yonatan Belinkov、Ori Ram、Inbal Magar、Omri Abend、Ehud Karpas、Amnon Shashua、Kevin Leyton-Brown 和 Yoav Shoham。2023年。大语言模型的并行上下文窗口。arXiv:2212.10947 [cs.CL] https://arxiv.org/abs/2212.10947[65] Gabin Schieffer、Ruimin Shi、Stefano Markidis、Andreas Herten、Jennifer Faj 和 Ivy Peng。2024年。理解 AMD 多 GPU 系统中基于 Infinity Fabric 的数据移动。arXiv: 2410 00801 [ ] 2410 00801 。cs.DC [66] 2025 https://arxiv.org/abs/ 。SGLang。。SGLang — 文档。https://docs.sglang.ai/。访问日期：2025年9月15日。[67] Christopher J. Shallue、Jaehoon Lee、Joseph Antognini、Jascha Sohl-Dickstein、Roy Frostig 和 George E. Dahl。2019年。测量数据并行对神经网络训练的影响

对神经网络训练的影响。arXiv:1811.03600 [cs.LG] https://arxiv.org/abs/1811.03600[68] Noam Shazeer。2020年。GLU变体改进 Transformer。arXiv:2002.05202 [cs.LG] https://arxiv.org/abs/2002.05202[69] Noam Shazeer、Azalia Mirhoseini、Krzysztof Maziarz、Andy Davis、Quoc Le、Geoffrey Hinton 和 Jeff Dean。2017年。规模惊人的神经网络：稀疏门控专家混合层。arXiv:1701.06538 [cs.LG] https: //arxiv.org/abs/1701.06538[70] Ying Sheng、Lianmin Zheng、Binhang Yuan、Zhuohan Li、Max Ryabinin、Daniel Y. Fu、Zhiqiang Xie、Beidi Chen、Clark Barrett、Joseph E. Gonzalez、Percy Liang、Christopher Ré、Ion Stoica 和 Ce Zhang。2023年。FlexGen：使用单个 GPU 对大语言模型进行高吞吐量生成式推理。arXiv:2303.06865 [cs.LG] https://arxiv.org/abs/2303.06865[71] Ying Sheng、Lianmin Zheng、Binhang Yuan、Zhuohan Li、Max Ryabinin、Daniel Y. Fu、Zhiqiang Xie、Beidi Chen、Clark Barrett、Joseph E. Gonzalez、Percy Liang、Christopher Ré、Ion Stoica 和 Ce Zhang。2023年。FlexGen：使用单个 GPU 对大语言模型进行高吞吐量生成式推理。arXiv:2303.06865 [cs.LG] https://arxiv.org/abs/2303.06865[72] Mohammad Shoeybi、Mostofa Patwary、Raul Puri、Patrick LeGresley、Jared Casper 和 Bryan Catanzaro。2020年。Megatron-LM：使用模型并行训练数十亿参数语言模型。arXiv:1909.08053 [cs.CL] https: //arxiv.org/abs/1909.08053[73] Chenchen Shou、Guyue Liu、Hao Nie、Huaiyu Meng、Yu Zhou、Yimin Jiang、Wenqing Lv、Yelong Xu、Yuanwei Lu、Zhang Chen、Yanbo Yu、Yichen Shen、Yibo Zhu 和 Daxin Jiang。2025年。InfiniteHBD：利用光电路交换收发器为大语言模型构建数据中心规模的高带宽域。载于《2025年ACM SIGCOMM会议论文集》（葡萄牙科英布拉圣方济各修道院）（SIGCOMM '25）。计算机协会，纽约州纽约市，美国，1–23。https://doi.org/10.1145/3718958.3750468[74] Mirac Suzgun、Nathan Scales、Nathanael Schärli、Sebastian Gehrmann、Yi Tay、Hyung Won Chung、Aakanksha Chowdhery、Quoc V. Le、Ed H. Chi、Denny Zhou 和 Jason Wei。2022年。具有挑战性的 BIG-Bench任务，以及思维链能否解决这些任务。arXiv:2210.09261 [cs.CL] https://arxiv.org/abs/2210. 09261[75] Mistral AI团队。2024年。规模足够大——Mistral Large 2。https://mistral.ai/news/ mistral-large-2407。[76] Albert Tseng、Tao Yu 和 Youngsuk Park。2025年。使用 MXFP4 训练大语言模型。arXiv: 2502 20586 [ ] 2502 20586 。cs.LG [77] https://arxiv.org/abs/ 。Ehsan Ullah、Anil Parwani、Mirza Mansoor Baig 和 Rajendra Singh。2024年。使用 ChatGPT 等大语言模型（LLM）进行诊断医学的挑战与障碍：聚焦数字病理学的近期范围综述。《诊断病理学》19，1（2024），43。[78] Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N. Gomez、Łukasz Kaiser 和 Illia Polosukhin。2017年。Attention Is All You Need。载于第31届神经信息处理系统国际会议论文集（美国加利福尼亚州长滩）（NIPS'17）。Curran Associates Inc.，美国纽约州雷德胡克，6000–6010。[79] Zongwu Wang、Fangxin Liu、Mingshuai Li 和 Li Jiang。2024年。TokenRing：通过双向通信实现无限上下文大语言模型的高效并行框架。arXiv:2412.20501 [cs.DC] https://arxiv.org/abs/2412.20501[80] Jason Wei、Yi Tay、Rishi Bommasani、Colin Raffel、Barret Zoph、Sebastian Borgeaud、Dani Yogatama、Maarten Bosma、Denny Zhou、Donald Metzler、Ed H. Chi、Tatsunori Hashimoto、Oriol Vinyals、Percy Liang、Jeff Dean 和 William Fedus。2022年。大语言模型的涌现能力。arXiv:2206.07682 [cs.CL] https://arxiv.org/abs/2206.07682[81] Yuhui Xu、Zhanming Jie、Hanze Dong、Lei Wang、Xudong Lu、Aojun Zhou、Amrita Saha、Caiming Xiong 和 Doyen Sahoo。2025年。ThinK：通过查询驱动剪枝打造更精简的键缓存。arXiv:2407.21018 [cs.CL] https://arxiv.org/abs/2407. 21018[82] Amy Yang、Jingyi Yang、Aya Ibrahim、Xinfeng Xie、Bangsheng Tang、Grigory Sizov、Jeremy Reizenstein、Jongsoo Park 和 Jianyu Huang。2025年。面向可扩展百万令牌推理的上下文并行。arXiv:2411.01783 [cs.DC] https://arxiv.org/abs/2411.01783[83] Zhewei Yao、Reza Yazdani Aminabadi、Minjia Zhang、Xiaoxia Wu、Conglong Li 和 Yuxiong He。2022年。ZeroQuant：面向大规模Transformer的高效且经济的训练后量化。arXiv:2206.01861 [cs.CL] https: // arxiv.org/abs/ 2206 01861 。Gyeong-In Yu、Joo Seong Jeong、Geon-Woo Kim、Soojeong Kim 和 Byung-Gon Chun。2022年。Orca：基于Transformer的生成模型分布式服务系统。载于第16届USENIX操作系统设计与实现研讨会（OSDI 22）。USENIX 协会，卡尔斯巴德，加利福尼亚州，521–538。https://www.usenix.org/conference/osdi22/presentation/yu [84] [85] Haopeng Zhang、Philip S. Yu 和 Jiawei Zhang。2025年。文本摘要系统综述：从统计方法到大语言模型。《ACM计算调查》57，11，文章277（2025年6月），41页。https://doi.org/10.1145/3731445[86] Kunming Zhang、Hanlong Liao、Junyu Xue 和 Guoming Tang。2025年。BandPilot：迈向面向性能的人工智能集群 GPU调度。arXiv:2506.15595 [cs.DC] https://arxiv.org/abs/2506.15595[87] Lianmin Zheng、Zhuohan Li、Hao Zhang、Yonghao Zhuang、Zhifeng Chen、Yanping Huang、Yida Wang、Yuanzhong Xu、Danyang Zhuo、Eric P. Xing、Joseph E. Gonzalez 和 Ion Stoica。2022年。Alpa：面向分布式深度学习的算子间与算子内并行化自动实现。arXiv:2201.12023 [cs.LG] https://arxiv.org/abs/2201.12023[88] Yinmin Zhong、Shengyu Liu、Junda Chen、Jianbo Hu、Yibo Zhu、Xuanzhe Liu、Xin Jin 和 Hao Zhang。2024年。DistServe：通过分离预填充与解码，实现有效吞吐量优化的大语言模型服务。载于第 *18* 届 *USENIX* 操作系统设计与实现研讨会（*OSDI 24*）。USENIX 协会，圣克拉拉，加利福尼亚州，193–210。https://www.usenix.org/conference/osdi24/presentation/zhong-yinmin