arXiv:2611.32572v1 [cs.DC] 24 Nov 2026

打破混合专家中的路由与设备间依赖

John Zarka [*1] **Danyal Ahmed** [*1] **Eugene Belilovsky** [2 3] **Edouard Oyallon** [1 4]

摘要

随着混合专家（MoE）模型规模不断扩大，专家并行（EP）对于在设备间分布专家变得愈发重要。然而，由此产生的令牌分发与合并操作可能成为推理过程中的主要瓶颈。标准混合专家架构中的顺序依赖限制了这些通信阶段与计算的重叠。我们提出 BRAID-MoE，这种架构将阶梯式残差连接与早期路由相结合，以放宽这些依赖。阶梯式残差使每个专家合并操作都可以延迟到下一个块执行，而早期路由则根据合并前残差计算下一个路由决策，同时专家仍接收更新后的合并后隐藏状态。由此形成一个扩展的通信窗口，使前一轮合并和下一轮分发能够在专用通信流上背靠背执行，并与注意力计算重叠。在参数规模为 0.4B-A0.1B 和 2.5B-A0.5B 的 Qwen3 结构混合专家模型上，BRAID 在计算量相同的情况下，训练效果与顺序训练十分接近，验证损失分别仅增加 1% 和 0.5%。我们进一步证明，现有 MoE 检查点只需少量额外训练即可转换为 BRAID，并大体恢复原有准确率。在转换后的 Qwen3-30B-A3B 模型上，BRAID 在预填充阶段最高可实现 $1.29\times$ 的加速比，在解码阶段最高可实现 $1.37\times$ 的加速比；在一个或两个配备四块 H100 的节点上，面对多种工作负载和专家并行配置，至少可实现 $1.12\times$ 的加速比。在所有测试设置中，其表现均优于其他竞争性的重叠策略；当通信构成主要延迟瓶颈，且其持续时间接近可用计算窗口时，收益最大。这些结果表明，早期路由能够以适度的模型质量代价，在混合专家模型推理中实现更有效的通信–计算重叠。

1 引言

稀疏混合专家模型在不按比例增加每个词元的算术运算量的情况下提升参数容量：每个词元仅激活规模大得多的专家库中的一小部分（Shazeer等人，2017年；Lepikhin等人，2021；Fedus等人，2022）。这种设计已成为前沿开放权重模型的核心。例如，Qwen3包含混合专家模型变体，总参数量最高达235B，其中激活参数量为22B

每个词元（Yang 等人，2025年）；DeepSeek-V3采用包含大量路由专家的专家库，并进行稀疏激活（DeepSeek-AI 等人，2025年）；而Kimi K3则进一步提高了稀疏性，在一个拥有2.8T参数的模型中激活896个路由专家中的16个（Team等人，2026年）。随着专家库的增长速度超过活跃计算量，专家并行对于混合专家模型的扩展变得至关重要：专家被分片到各个设备上，词元则会被发送到托管其所选专家的设备。

在专家并行下，每个混合专家层都会将词元分发给远程专家，并合并返回的输出。这些通信阶段可能占据执行时间的很大一部分。例如，COMET（Zhang 等人，2025b）报告称，在多种常用混合专家模型中，设备间通信平均占据总执行时间的近一半。此外，通信与计算之间的顺序依赖限制了重叠的机会。为降低这一开销，DeepEP（Zhao 等人，2025年）等专用库提供高吞吐、低延迟的分发与合并内核，而COMET（Zhang 等人，2025b）则优化通信–计算重叠，Lina（Li 等人，2023年）解决通信争用和不均衡的全对全流量问题。一种常见方法是将底层内核优化与双流设计相结合，从而实现通信与计算的并发，平衡SM资源使用，并尽量减少不必要的同步，以降低整体执行时间。

尽管混合专家训练中通信–计算重叠已得到广泛研究，但推理面临着一组不同的挑战。训练具有更大的调度灵活性（Guo 等人，2026年），可以利用相互独立的

[*]共同贡献 [1]ISIR，索邦大学，巴黎，法国 [2]计算机科学与软件工程系，康考迪亚大学，蒙特利尔，加拿大 [3]Mila——魁北克人工智能研究所，蒙特利尔，加拿大 [4]法国国家科学研究中心，巴黎，法国。通信作者：John Zarka <john.zarka@sorbonne-universite.fr>。

预印本。*2026*年*11*月。

反向计算（Jiang 等人，2024年），以及跨微批次的前向–反向流水线（DeepSeek-AI 等人，2025年），以隐藏通信延迟。然而，推理仅限于前向传播，其中专家分发、计算与合并之间的依赖关系使重叠机会更少。因此，要利用并行性，通常需要将词元划分为微批次，并以流水线方式执行（DeepSeek-AI 等人，2025年；vLLM 项目，2026年）。然而，这类方法的效果在很大程度上取决于可用词元的数量。在预填充阶段，较长的输入序列可提供足够多的词元以供分块；而自回归解码在每一步中每个请求只生成一个词元。因此，解码期间要隐藏专家并行通信，通常需要对多个并发请求进行批处理；但在并发度低、内存受限或延迟要求严格时，这可能很难实现。

架构修改提供了另一种提升混合专家执行效率的方法。尽管此类修改需要确保模型质量得到保持，但它们可以为系统级优化带来新的机会。我们的工作建立在两个互补理念之上：阶梯架构和早期路由。阶梯架构（Zhang 等人，2025年a；Dukler 等人，2026年）通过复制残差流，并相对于彼此错开注意力与前馈网络/混合专家计算，对 Transformer 架构进行微小修改，从而在对模型质量影响极小的情况下实现通信与计算重叠。该方法并不专用于混合专家模型，最初是为了在稠密 Transformer 中实现张量并行通信与计算的重叠而探索的。此外，Zhang 等人（2025年a）证明，标准 Transformer 模型只需有限的额外训练即可转换为阶梯架构。另一方面，早期路由（Hwang等人，2024年；Cai 等人，2025年）使专家分配能够在更早的层中获得，从而允许在相应的混合专家层之前开始令牌分发，进而降低通信延迟，但可能会导致模型收敛性下降。

基于这些互补的方法，我们提出 BRAID-MoE（打破混合专家中的路由与设备间依赖），它结合了阶梯式残差连接和早期路由，在缓解各自局限的同时，挖掘出更多通信–计算重叠机会。BRAID 将路由与前一层的合并操作解耦，同时为专家计算保留最新的激活值，使跨层边界的连续合并与分发操作无需插入路由步骤即可背靠背执行。我们通过采用 DeepEP 异步通信内核的双流执行调度来利用这一架构，使两个通信阶段都能与注意力重叠。此外，通过在这些通信操作之后调度专家计算，BRAID 创造了使专家计算与后续注意力侧通信（例如张量并行集体通信）重叠的机会。

贡献。 我们的贡献如下：
（i）我们提出了 BRAID-MoE，这是一种将阶梯式残差连接与早期路由相结合的架构。路由器读取合并前残差，而分发和专家计算则使用新生成的合并后激活值。这使路由从串行的合并—分发路径中移除，为未来将两个通信阶段融合到单个内核中创造了可能。**（ii）**我们实现了事件驱动的双流执行调度，以及定制的 DeepEP 合并尾处理，将残差相加与 RMSNorm 融合在一起。该调度将合并和分发作为前后紧接的通信内核执行，并使两个阶段都与路由和注意力重叠。**（iii）**我们表明，BRAID 可以从头预训练至最高 2.5B-A0.5B 参数规模；与匹配的顺序式混合专家模型相比，验证交叉熵的增幅低于 1%。**（iv）**我们证明，预训练顺序式混合专家检查点可以通过有限的持续训练转换为 BRAID，其中包括对指令微调的 OLMoE-1B-7B 检查点进行后训练恢复。**（v）**在转换后的 Qwen3-30B-A3B 检查点上，使用一个或两个四 H100 节点，相较于顺序执行，我们在预填充中实现了最高 $1.29\times$ 的加速比，在解码中实现了最高 $1.37\times$ 的加速比，并且在所有测试设置下都优于具有通信–计算重叠的竞争性调度方案。

论文结构。第3节介绍顺序、并行、梯形和早期路由混合专家模型的形式。第4 节介绍 BRAID 架构及其多流执行调度。第5 节评估预训练、检查点适配和后训练恢复；第6 节报告预填充和解码延迟以及通信重叠。我们的源代码基于 TorchTitan[1] (Liang 等人，2025年）和 DeepEP[2] (Zhao 等人，2025年），可在以下地址获取：
https://github.com/j-zarka/braid-moe/。

2 相关工作

降低通信量。提高专家并行通信效率的一种方法，是减少设备之间交换的数据量。Latent-MoE（Elango 等人，2026年）在路由专家路径周围引入共享下投影和上投影，使令牌分发、专家计算和输出合并能够在低维潜在空间中进行（Janson等人，2026b；a）。这既降低了通信量，也减少了专家内存流量，从而能够在相近的执行成本下支持更大的专家容量。

[1]https://github.com/pytorch/torchtitan[2] https://github.com/deepseek-ai/DeepEP

近期，KimiK3采用了Stable LatentMoE，将路由表示的维度从7168降至3584，同时激活896个路由专家中的16个（Team等人，2026年）。多头LatentMoE则进一步探索通过头并行改进通信效率的架构调整（Cui等人，2026）。我们还注意到，本地SGD 方法可以与专家并行结合（Belilovsky等人，2026；Cagnasso等人，2026a；b），尽管这些方法并非专为混合专家模型设计。这些方法旨在降低通信成本，而我们的工作则侧重于通过重叠来隐藏通信延迟。

专家并行优化与多流执行。另一类研究在不修改模型架构或改变其表达能力的情况下，改进专家并行执行。DeepEP（Zhao 等人，2025年）为令牌分发与合并提供了专用的高吞吐、低延迟内核，并支持异步通信。Tutel（Hwang等人，2023年）引入自适应并行性与通信—计算流水线，以适应变化的专家负载；而Lina（Li 等人，2023年）则解决了训练期间的通信争用，以及推理期间不均衡的全互联流量。更一般地，多流实现会在独立的 GPU 流上调度通信与计算，以实现并发执行，但重叠内核可能会争用 GPU 资源。这种重叠还受到底层模型依赖关系的限制：通信只能与其输入已就绪的操作重叠。

通信–计算重叠与调度。一些系统通过在现有计算图中分解并重新调度操作来提升重叠程度。COMET（Zhang 等人，2025b）采用细粒度依赖关系分析和工作负载调度，使混合专家通信与专家计算实现重叠。FLUX（Chang 等人，2024年）将通信与相邻的 GPU 计算融合，主要针对张量并行集体通信。在计算图层面，Lancet（Jiang 等人，2024年）利用训练期间的额外调度机会，尤其是将通信与反向传播计算重叠。我们注意到，通过修改反向传播过程也可以实现额外的重叠（Rivaud 等人，2025年；Nabli 等人，2025年；Kopyshevskiy 等人，2026年），不过这种方法仅适用于训练。微批处理为推理期间实现重叠提供了另一种途径：DeepSeek-V3 并发调度来自不同微批次的注意力和混合专家执行（DeepSeek-AI 等人，2025年），而 vLLM 的双批次重叠则在连续的微批次之间交错执行通信和计算（vLLM 项目，2026年）。尽管这些技术十分有效，但它们受到底层混合专家架构顺序依赖关系的限制，而这种依赖限制了暴露独立计算并重叠通信的机会。

通过残差连接实现架构级并行。除了执行级优化之外，架构修改还可以直接放宽变换器子层之间的依赖关系。采用并行注意力和 FFN 分支的架构（PaLM 中便显著采用了这一设计，Chowdhery 等人，2023年）允许两个分支在同一残差状态上运行，从而暴露出额外的并发性。Ladder Residual（Zhang 等人，2025a）进一步拓展了这一思想：让每个模块处理更早的残差状态，从而使其计算能够与前一个模块的通信重叠。尽管适用于一般的残差架构，Ladder Residual 最初是为在稠密 Transformer 中实现张量并行通信与计算的重叠而开发的。作者还证明，现有的预训练 Transformer 可以通过有限的额外训练，部分转换为这种架构。将这一原则扩展到混合专家模型，FarSkip-Collective（Dukler 等人，2026年）利用部分更新的残差状态，使下一注意力层能够在前一个混合专家合并完成之前开始运行，从而让合并与分发和注意力计算重叠。然而，路由仍然依赖于前一个合并操作，因此保留了顺序的“合并—路由—分发”依赖关系，并使重叠效果对注意力计算的划分较为敏感。这些工作展示了架构修改如何暴露出额外的并行性，供系统级优化加以利用。

预判路由决策。一种互补方法是通过根据较早的表示计算专家分配来预判路由决策。Pre-gated MoE（Hwang等人，2024年）提前一个混合专家模块评估路由决策，使得在前一混合专家模块执行期间，可以将选定的专家权重从 CPU 预取到 GPU。尽管该方法的动机是专家卸载，而非重叠专家并行通信，但它展示了提前计算路由决策的潜力。RMoE（Qiu 等人，2025年）通过循环路由器探索了一种相关但不同的方法，利用跨层信息来改进专家选择。与我们的设置更为接近的是，ScMoE（Cai等人，2025年）引入跨层捷径，使专家通信能够与注意力和 FFN 计算重叠。在其主要配置中，稠密 FFN 和混合专家模块交错排列，路由专家使用来自前一模块的表示。这为通信–计算重叠创造了更大的窗口，但路由和路由专家计算都依赖于较早激活值。相比之下，BRAID 将路由与前一合并操作解耦：专家分配根据合并前残差计算，而分发则作用于新鲜的合并后激活值。结合阶梯式残差连接，这使得

合并与分发可以背靠背执行，从而更高效地与注意力重叠，并有可能将两个通信阶段融合到单个内核中。它还使专家计算能够与后续的注意力侧通信重叠。

3 现有方法

现在，我们介绍本文考察的架构变体，并着重说明各变体如何改变注意力、路由与专家计算之间的依赖关系，以展现通信–计算重叠。图 1展示了下文讨论的不同配置。

3.1 顺序式 MoE

我们将 Transformer 块的索引记为 ℓ，并将词元数量记为 T。遵循 Guo 等人 (2026年) 的记法，令 d 表示隐藏维度，E 表示路由专家的数量，K 表示每个词元选中的专家数量。为便于阅读，我们采用预归一化约定，并省略显式的归一化算子，将其隐式地并入相应的注意力和 MoE 操作中。给定输入 $h_\ell \in \mathbb{R}^{T\times d}$ 至块 ℓ，标准的 MoE 变换器（Lepikhin 等人，2021；Fedus 等人，2022）执行：

$$\begin{aligned} z_\ell &= h_\ell + \mathrm{Attn}_\ell(h_\ell), \\ (S_\ell, \pi_\ell) &= \mathrm{Route}_\ell(z_\ell), \\ h_{\ell+1} &= z_\ell + \mathrm{MoE}_\ell(z_\ell; S_\ell, \pi_\ell)\,, \end{aligned} \tag{1}$$

其中，对于 $i \in [T]$,

$$\mathrm{MoE}_\ell(z_\ell; S_\ell, \pi_\ell)_i \triangleq \sum_{e=1}^{E} \pi_{\ell,i,e}\, S_{\ell,i,e}\, \mathrm{Expert}_{\ell,e}(z_{\ell,i}).$$

其中，$z_\ell \in \mathbb{R}^{T\times d}$ 是注意力后表示，Attn_ℓ 表示块 ℓ的注意力子模块，而 $\mathrm{Expert}_{\ell,\,e}$ 表示层 ℓ中对应于专家 e 的前馈网络。矩阵 $S_\ell \in \mathbb{R}^{T\times E}$ 包含路由权重，而$\pi_\ell \in \{0,1\}^{T\times E}$则对前-$K$ 个专家分配进行编码，其中 $\sum_e \pi_{\ell,\,i,\,e} = K$。

专家并行。 在专家并行下，专家分布在各个设备上。设 $\mathcal{J}_r \subseteq [E]$ 表示托管于 rank r上的专家。聚集操作会选出分配给每个专家的词元。在专家并行下，这变为一种分布式的 分发操作 $\mathrm{Dispatch}_\ell(\cdot,\ \pi_\ell)$，将词元传输到托管其所选专家的设备上。因此，在专家计算期间，每个 rank 都持有一组本地专家

$$\mathrm{Expert}^{\mathrm{loc}}_{\ell,r} = \{\mathrm{Expert}_{\ell,e} : e \in \mathcal{J}_r\}.$$

后续的 合并 会将这些专家的输出返回到其源秩，并执行按路由权重加权的聚合，以获得混合专家输出。在合并之前，这些输出一直分片存储在承载专家的各个秩之间。设 μ_ℓ 表示用于合并层 ℓ的专家输出而保留的路由权重和分发元数据；我们用 $\mathrm{Combine}_\ell(\cdot;\mu_\ell)$表示相应的集合通信操作。为清晰起见，在所有图示中，我们分别将 Route、Attn、Expert、Combine 和 Dispatch 缩写为 R、 A、 $\mathcal{E}$、 C和 D。

3.2 并行式 MoE

一种展现额外并行性的方法，是从同一个残差状态计算注意力和混合专家，而不是像公式（1）中那样按顺序计算。这种并行形式遵循并行Transformer设计（Chowdhery 等，2023年），消除了专家计算对当前注意力输出的依赖。于是，并行混合专家块可写为

$$\begin{aligned} (S_\ell, \pi_\ell) &= \mathrm{Route}_\ell(h_\ell), \\ h_{\ell+1} &= h_\ell + \mathrm{Attn}_\ell(h_\ell) + \mathrm{MoE}_\ell(h_\ell; S_\ell, \pi_\ell)\,. \end{aligned}$$

由于两个分支都消耗 h_ℓ，因此路由完成后，注意力可以与专家分发重叠执行。然而，专家计算会安排在注意力之后，因为二者重叠可能导致 GPU资源争用（附录 C.2（SM 分配和资源争用））。下一个块仍依赖于两个分支都完成，包括 MoE合并操作。因此，尽管并行式 MoE 暴露了块内重叠机会，但在连续块之间，通信仍处于关键路径上。

3.3 梯形MoE

Ladder Residual（Zhang 等人，2025年a）通过引入两个交错残差流，进一步放宽了连续 Transformer 子层之间的依赖关系。每个子层不再消耗最近更新的残差，而是处理更早的表示，同时将其输出累积到当前残差流中。在 MoE设置中，与 Dukler 等人（2026年）的形式密切相关的一种表述为

$$\begin{aligned} (S_\ell, \pi_\ell) &= \mathrm{Route}_\ell(y_\ell), \\ h_{\ell+1} &= y_\ell + \mathrm{Attn}_\ell(h_\ell), \\ y_{\ell+1} &= h_{\ell+1} + \mathrm{MoE}_\ell(y_\ell; S_\ell, \pi_\ell). \end{aligned}$$

这里， y_ℓ 表示纳入前一个 MoE 输出后的残差。与并行式 MoE 不同，现在两个分支在不同的残差状态上运行。具体而言，注意力消耗 h_ℓ，它在前一个 MoE 合并完成之前就已可用；而路由和专家计算则消耗更新后的残差 y_ℓ。这使得第 ℓ 层的注意力能够与前一个合并操作 $C_{\ell-1}$ 重叠执行。

然而，路由需要 y_ℓ，而该信息只有在前序合并完成后才可用。因此，分发必须

等待路由，从而保留顺序的合并–路由–分发依赖，限制了进一步的通信重叠。

3.4 早期路由 MoE

此外，Hwang等人（2024年）提出从更早的表示 $z_{\ell-1}$ 中计算路由决策，同时保留新鲜表示 z_ℓ 用于专家计算。一个 p-早期路由混合专家模型从提前 p 层的残差表示中计算路由决策，同时保留公式（1）中的标准混合专家公式。具体而言，路由方程变为

$$(S_\ell, \pi_\ell) = \mathrm{Route}_\ell(z_{\ell-p}).$$

对于 $p = 1$，由于 $z_{\ell-1}$ 在前一个 MoE 合并完成之前即可用，因此可以提前计算专家分配。这将路由从合并到分发的关键路径中移除。然而，分发仍需要更新后的注意力后表示 z_ℓ，从而保留了合并–注意力–分发依赖，阻止通信连续进行。

4 我们的方法：BRAID

基于梯形MoE与早期路由 MoE的互补思想，我们提出了 BRAID-MoE，这是一种新颖的混合专家架构，将阶梯式残差连接与早期路由相结合，以实现更多通信–计算重叠。其核心思想是将路由与前一个 MoE 合并解耦，同时保留新鲜激活值用于专家计算。这样可以将注意力和路由从合并到分发的关键路径中移除，使专家并行通信操作能够连续进行，并与注意力重叠。

4.1 BRAID 架构

如图所示1，BRAID-MoE 保留了阶梯式混合专家模型中的两个交错残差流，但使用较早的残差状态 h_ℓ而非 y_ℓ来计算路由决策。由此形成的架构定义如下：

$$\begin{aligned}(S_\ell, \pi_\ell) &= \mathrm{Route}_\ell(h_\ell),\\ h_{\ell+1} &= y_\ell + \mathrm{Attn}_\ell(h_\ell),\\ y_{\ell+1} &= h_{\ell+1} + \mathrm{MoE}_\ell(y_\ell; S_\ell, \pi_\ell).\end{aligned} \tag{2}$$

在分布式实现中，y_ℓ 通过合并第 $\ell-1$ 层的分片专家输出并将其加到 h_ℓ 上获得。由于在此次 MoE 合并完成之前即可获得 h_ℓ，因此路由独立于前一次 MoE 合并。专家仍会处理 y_ℓ，其中包含前一次 MoE 的输出，但不包含当前的注意力更新。因此，无需专家处理陈旧激活值，即可提前执行路由，专家处理陈旧激活值。由此得到的执行调度详见算法1。

因此，与阶梯式混合专家模型不同，BRAID-MoE 可以将前一层的合并与当前层的分发安排为连续执行，两者之间仅隔着残差更新。这也为未来跨块边界融合这两个通信阶段创造了机会。

4.2 多流执行

连续通信内核。如图1所示，通信流首先将层 X_ℓ $\ell-1$ 的分片专家输出与残 h_ℓ 差 y_ℓ合并，得到，随后立即依据预先计算的路由分配 π_ℓ将其分发给下一层的专家。在我们的实现中，我们扩展了 DeepEP 的 `EPBuffer` 合并内核，将残差相加和 MoE 特定 RMSNorm 融合到其尾处理中，同时保留未归一化的残差 y_ℓ，以用于后续的残差更新，具体如图3所示。这避免了单独的归一化内核，并减少了中间内存流量。重要的是，合并与分发仍然是彼此独立的通信内核。

通信与计算重叠。 我们将 BRAID-MoE 的架构特性转化为双流执行调度，并在图2中将其与顺序、并行、梯形和早期路由混合专家模型的调度并列展示。我们将一个 CUDA 流专用于通信（C、 D），另一个专用于计算（R、 A、 $\mathcal{E}$），仅在数据依赖要求时使用 CUDA 事件进行同步。由于路由和注意力都消耗 h_ℓ，因此它们可以独立于前序合并操作执行。因此，路由可以与合并重叠，而注意力可以与合并和分发同时进行。由此，连续的通信操作大多可以隐藏在路由和注意力计算之后。

剩余的依赖涉及专家计算：在消耗重新分发的激活值之前，专家计算必须等待分发完成。同样，后续的合并操作也无法在相应的专家输出准备就绪之前开始。因此，与路由和注意力不同，在该调度中，专家计算无法隐藏在其所在层的通信操作之后。不过，只要依赖关系和调度安排允许，其执行仍可与其他操作产生的独立通信重叠，例如后续注意力层中的张量并行集体通信。

实现注意事项。 尽管将通信和计算分离到不同的 CUDA 流中可以暴露并发性，但实际的重叠取决于 GPU 资源的可用性。尤其是，DeepEP

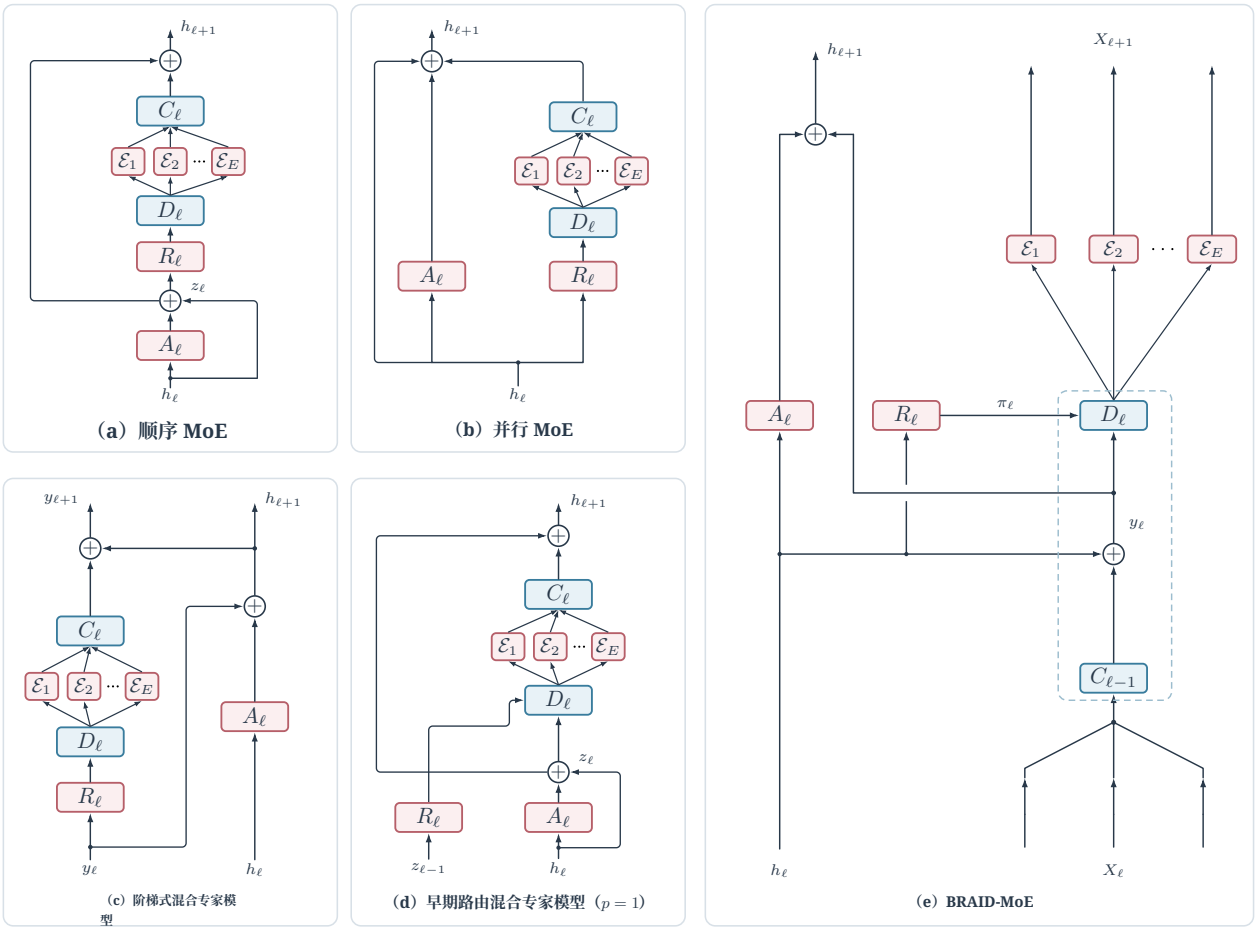

图 *1*。（a）顺序、（b）并行、（c）梯形、（d）早期路由（$p = 1$）和（e）BRAID 混合专家架构的比较。红色方块表示计算，蓝色方块表示专家并行通信。在 BRAID 中，分片专家输出与残差流合并后，会立即分发给下一层的专家，从而实现连续的通信操作。

通信内核会占用 SM 资源，可能与路由和注意力计算竞争。为通信分配更多 SM 可以加速单个集合通信操作，但会减少可用于并发计算的资源。相反，限制通信的 SM 占用率可能会牺牲独立通信吞吐量，但有助于改善重叠。因此，BRAID 引入了一种资源分配权衡：目标不一定是分别最大化每个内核的吞吐量，而是通过平衡通信进度与计算吞吐量，最小化端到端层延迟。

5 训练实验

正如 Zhang 等人（2025a）所述，我们的训练实验旨在实现三个目标：(i) 评估 BRAID 从头预训练时的可扩展性；(ii) 评估现有的预训练混合专家模型能否通过少量持续训练转换为 BRAID 架构，同时保持其原有的模型质量；以及 (iii) 研究监督微调能否在将经过指令微调的混合专家模型转换为 BRAID 后恢复其下游性能。

5.1 从头预训练

实验设置。 我们研究了受 Qwen3 启发的混合专家架构（Yang 等人，2025年），涵盖两种模型规模。较小模型包含 9980 万个激活参数和 3.92 亿个总参数，较大模型包含 5.04 亿个激活参数和 25.1 亿个总参数。两个模型均使用 $E = 64$ 个路由专家、top-K路由，包含 $K = 8$，且不使用共享专家。按照标准做法，注意力和专家矩阵使用 Muon（Jordan 等人，2024年）进行优化，而嵌入、路由器、归一化参数和语言模型头使用 AdamW（Loshchilov 和 Hutter，2019年）。我们采用 5% 的线性预热，随后以余弦衰减将学习率降至峰值学习率的 10%。全局批次包含 512 个长度为 2,048 的序列，在两种规模下，每个优化步骤对应 1,048,576 个词元。两个模型均在预

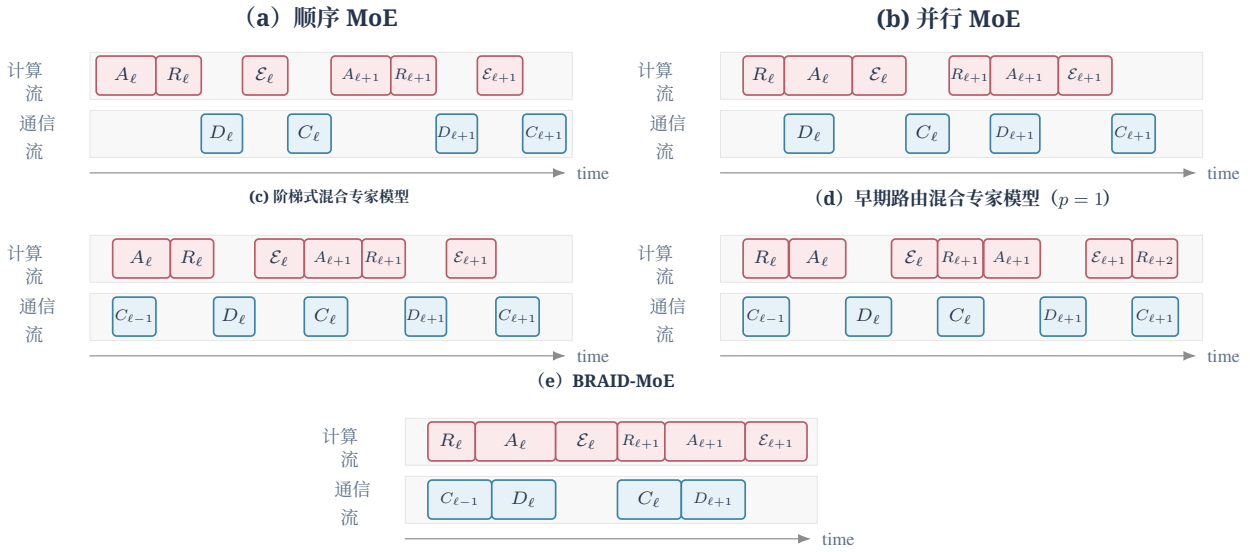

图2。（a）顺序、（b）并行、（c）梯形、（d）早期路由（$p = 1$）和（e）BRAID 混合专家模型的双流执行调度方案示意图。红色块表示路由（R）、注意力（A）和专家计算（$\mathcal{E}$），蓝色块表示分发（D）和合并通信（C）。（a）顺序混合专家模型将注意力与混合专家执行串行化。（b）并行 MoE 将分发与注意力重叠，因为两个分支消耗相同的残差。（c）阶梯式混合专家模型将注意力与前置合并重叠，但路由仍依赖于该合并操作。（d）早期路由从更早的表示中计算路由，使其与前置合并重叠，而分发仍需等待新生成的注意力后表示。（e）BRAID 在合并分片专家输出的同时，根据更早的残差计算路由，从而支持连续层之间连续执行合并与分发操作，并使通信与注意力重叠。计时仅为示意，并非实测结果。

训练于 FineWeb v1.4.0（Penedo 等人，2024年），使用 Llama-2 分词器（Touvron 等人，2023年），词汇表包含 32,000 个词元。评估使用固定的留出数据流，其中包含 100,139,008 个词元，且与训练数据集没有重叠。我们采用联合混合专家规模定律的预Ludziejewski 等人，2025年）来确定训练预算，为 0.4B-A0.1B 模型获得 3.27B 个词元，为 2.5B-A0.5B 模型获得 104.6亿个词元。附录 B.2 详细说明了模型—词元分配依据、其局部 IsoFLOP 验证以及所使用的数据流。

我们首先训练顺序 MoE 基线，随后训练三种架构变体：并行 MoE、阶梯式混合专家模型和 BRAID-MoE。BRAID 为其早期路由器使用单独的 RMSNorm，与专家输入归一化相区分。重要的是，我们在不同架构间保持整个训练方案不变，包括优化器、学习率计划、批量大小、训练时长以及所有其他超参数。我们不针对特定架构进行调优，从而能够隔离架构修改对训练动态和最终模型质量的影响。更多关于架构、初始化和优化的细节见附录 B；硬件规格见附录 A。

结果。 图4 报告了 2.5B-A0.5B 模型训练困惑度随已处理词元数量变化的情况。为便于进行受控比较，所有实验均使用相同的数据加载器随机种子。我们观察到，三种替代架构均表现出稳定的收敛性，其训练轨迹与顺序 MoE 基线高度吻合。并行 MoE、普通阶梯式混合专家模型和 BRAID-MoE 达到了相近的训练困惑度。

表1报告了两种模型尺寸下的最终验证损失。对于模型尺寸 0.4B-A0.1B，三种替代架构的验证损失均与顺序基线相差不超过 1.02%。在 2.5B-A0.5B 规模下，三种替代架构的验证损失均与顺序基线相差不超过 0.6%：并行 MoE、阶梯式混合专家模型和 BRAID-MoE 分别相差 0.53%、0.58% 和 0.49%。这些结果表明，即使训练方案保持不变，替代调度方案带来的额外并行性也可以实现，而不会显著损害模型质量。

最后，我们强调，我们的比较固定的是训练词元数量，而非实际训练时间。该方案将模型质量与执行效率差异区分开来。在固定实际时间预算下进行另一种比较，可能会有利于支持更高通信–计算重叠的架构，使其能够在相同时间预算内处理更多训练词元。评估这一潜在优势需要专门进行端到端训练吞吐量测量。

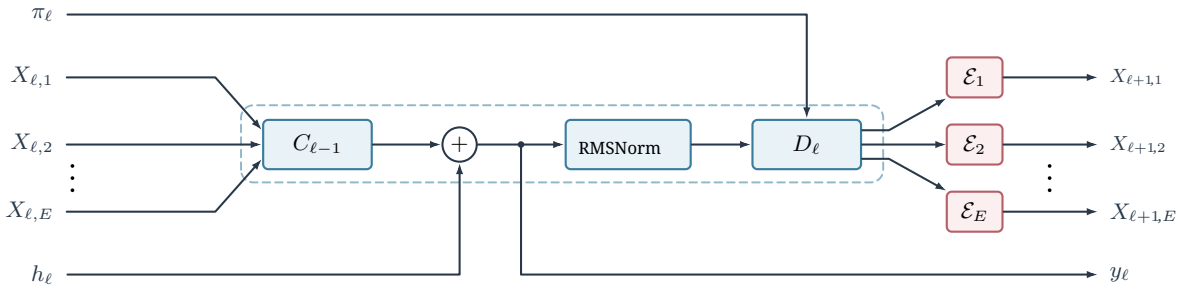

图*3*。BRAID-MoE 中的通信数据流。虚线区域突出显示了通过预计算路由分配实现的连续合并与分发操作。分片专家输出与残差流合并，以生成 y_ℓ。在我们的实现中，残差相加与 MoE 均方根层归一化被融合到合并尾处理中，并且仅在分发路径上应用归一化。未归一化的残差 y_ℓ 会被保留，用于后续的残差更新。

算法 1 BRAID-MoE 在第 ℓ 层的执行。其中，RECORD 表示在当前流上记录一个事件，而 WAIT 则插入对另一条流产生的事件的流内依赖。这些等待不会阻塞 CPU。

要求： 残差 h_ℓ、分片专家输出 X_ℓ、保留的合并元数据 $\mu_{\ell-1}$**确保：** 残差 $h_{\ell+1}$、分片专家输出 $X_{\ell+1}$、保留的元数据 μ_ℓ**计算流** $\mathcal{S}_{\text{comp}}$1: $(S_\ell, \pi_\ell) \leftarrow$ 路由$_\ell(h_\ell)$ 2: 记录(e_ℓ^R) 3: $a_\ell \leftarrow$注意力$_\ell(h_\ell)$ 4: 等待(e_ℓ^Y) 5: $h_{\ell+1} \leftarrow y_\ell + a_\ell$6: 等待$(e_\ell^D)$ 7: $X_{\ell+1} \leftarrow$专家$_\ell(v_\ell)$ 8: 记录(e_ℓ^E)

通信流 $\mathcal{S}_{\text{comm}}$
9: 等待$(e_{\ell-1}^E)$ 10: $m_\ell \leftarrow$ 合并$_{\ell-1}(X_\ell; \mu_{\ell-1})$ 11: $y_\ell \leftarrow h_\ell + m_\ell$12: 记录$(e_\ell^Y)$ 13: 等待(e_ℓ^R) 14: $(v_\ell, \mu_\ell) \leftarrow$ 分发$_\ell(y_\ell; S_\ell, \pi_\ell)$ 15: 记录(e_ℓ^D)

表 *1*。固定词元预算下的预训练比较。损失为留出验证交叉熵。差异是相对于顺序训练计划而言的。相对差异根据未四舍五入的交叉熵计算。

规模	调度	验证集交叉熵	相对于顺序
0.4B-A0.1B 3.27B 个词元	顺序	2.744	–
	并行	**2.770**	**+0.94%**
	梯形	2.772	+1.02%
	BRAID	2.771	+0.98%
2.5B-A0.5B 104.6亿个词元	顺序	2.345	–
	并行	2.357	+0.53%
	梯形	2.358	+0.58%
	BRAID	**2.356**	**+0.49%**

5.2 调整顺序检查点

问题设定。 我们研究预训练顺序 MoE 能否通过一段时间的短期持续训练，高效转换为 BRAID，而不必从头预训练。我们从预训练顺序 MoE 的最终检查点初始化 BRAID，并在与预训练和验证集不相交的 FineWeb v1.4.0 数据流上继续训练。我们将每个模型后半部分的变换器层转换为 BRAID 架构：对于 0.4B-A0.1B，对应第 7 至第 13 层；对于 2.5B-A0.5B，对应第 12 至第 23 层。现有模型权重直接复制，而新增的每个早期路由器 RMSNorm 均使用对应的、经过训练的专家输入 RMSNorm 初始化。

作为对照，我们在相同的续训词元预算下，继续训练原有的串行架构。两种架构使用相同的优化器组和学习率调度形状，但分别独立调优峰值学习率。更多续训超参数和数据流详情见附录B.4。

恢复。 我们将持续训练限制在原预训练预算的约 10%，相当于为 0.4B-A0.1B 额外训练 313 步，为 2.5B-A0.5B 额外训练 998 步。尽管额外计算量有限，BRAID 仍将性能恢复至接近原始顺序检查点的水平；如表2所示，与继续训练前相应的顺序检查点相比，0.4B-A0.1B 和 2.5B-A0.5B 的留出验证交叉熵分别仅高出 0.9% 和 0.3%。这些结果表明，现有顺序式 MoE 可以通过有限的额外训练适配为 BRAID，而无需从头预训练。

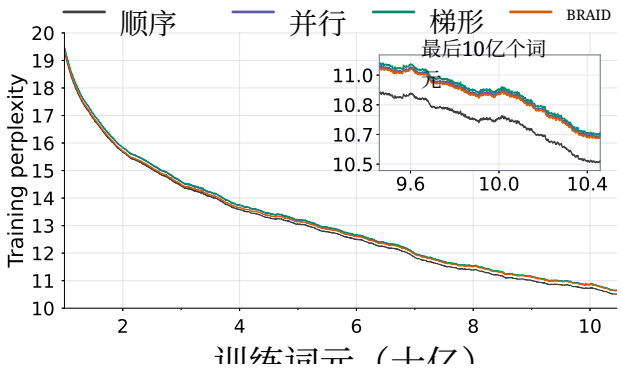

图4. 采用顺序、并行、梯形和 BRAID 调度方案时，2.5B-A0.5B Qwen3 风格混合专家模型的训练困惑度。曲线经过平滑处理，使用向前回溯 0.3B 训练词元的窗口。主图覆盖 10 亿至 104.6亿个词元，插图则放大显示最后10亿个词元。所有调度方案均采用匹配的训练设置。BRAID 对早期路由使用单独的归一化层。

表2. 检查点续训前后的留出验证交叉熵。续训在 0.4B-A0.1B 上进行 313 步，在 2.5B-A0.5B 上进行 998 步。顺序和 BRAID 续训分别使用 $0.1\times$ 和 $0.3\times$ 预训练峰值学习率。

	顺序		BRAID	
规模	之前	之后	之前	之后
0.4B-A0.1B	2.744	2.726	2.932	2.768
2.5B-A0.5B	2.345	2.319	2.416	2.352

5.3 后训练恢复

转换与恢复方案。 我们进一步探究，在将预训练顺序 MoE 转换为 BRAID 后，监督微调（SFT）能否恢复下游任务性能。为此，我们使用 OLMoE-1B-7B-0125-Instruct（Muennighoff 等人，2025年），并按照Zhang 等人 (2025a）的方法，将其上半部分 Transformer 层转换为 BRAID，其余层保持不变。我们评估三种配置：（i）原始顺序模型，（ii）SFT 前转换得到的 BRAID 模型，以及（iii）经过短暂 SFT 恢复阶段后的同一模型。转换过程遵循第 5.2节所述的初始化流程，恢复期间更新所有模型参数。我们在 Infinity-Instruct 7M 和 Gen 子集的 5% 样本上，对转换后的模型进行两个轮次的微调[3]，每个轮次约对应 205.8M 个词元。训练损失仅根据助手令牌计算。我们使用 Open Instruct 框架[4] 和 AdamW，峰值学习率为 5×10^{-6}，先进行 200 次线性预热更新，随后余弦衰减至零，全局批次大小为 32 个对话。这一有意限制的训练预算使我们能够评估：是否可以从现有的指令微调检查点采用 BRAID，而无需大量额外训练。附录中提供了进一步的优化和实现细节 B.5。

恢复结果。 我们使用 OLMES[5] (Gu 等人，2025年）的九项核心任务评估下游性能：ARC-Challenge、ARC-Easy、BoolQ、CommonsenseQA、HellaSwag、OpenBookQA、PIQA、Social IQa 和 Wino-Grande。三个配置均在相同条件下进行评估，包括任务提示、格式、少样本设置和指标实现。表 3 比较了三个配置的下游准确率。将模型的上半部分转换为 BRAID 后，平均准确率从 74.6 降至 60.2；随后进行的 SFT 恢复阶段则将其提升至 70.2。恢复后的模型与原始顺序检查点相差 4.4 个百分点以内。这些结果表明，在有限的额外训练预算下，可以将 BRAID 引入现有的指令微调混合专家，而无需从头进行完整预训练。

6 推理实验

现在，我们评估所提方法的推理效率，并比较四种执行调度方案：顺序、并行、梯形和 BRAID。

6.1 设置

基础设施。 实验在配备四块 H100 的节点上进行。节点内通信采用 NVLink（每块 GPU 的聚合双向带宽最高可达 900GB/s），节点之间则通过 InfiniBand NDR400（每条链路 400 Gb/s）互连。我们比较单节点内的专家并行=4与=8跨两个节点的专家并行；更多硬件细节见附录A。

模型与执行引擎。 我们使用其原生分词器，在 BF16 下通过 vLLM （Kwon 等人，2023 年）和 DeepEP （Zhao 等人，2025年）对 Qwen3- 30B-A3B-Base 进行基准测试。基准提示取自 FineWeb v1.4.0（Penedo 等人，2024年）的一个子集。架构变体未经重新训练，因此，我们在不同调度方案间重放相同的专家分配，以公平比较它们的执行效率。我们还使用相同的静态专家放置方案，该方案由 DeepSeek 的专家并行负载均衡器（EPLB）（DeepSeek-AI 等人，2025年）计算得出，且不进行专家复制，以改善这些基准提示上的 GPU负载均衡。该方案根据留出的FineWeb词元上的专家负载计算一次，并且在

[3]https://huggingface.co/datasets/BAAI/Infinity-Instruct[4]https://github.com/allenai/open-instruct

[5]https://github.com/allenai/olmes

表 3. OLMoE-1B-7B-0125-Instruct（1B 激活参数，7B 总参数）在九项核心 OLMES 任务上的后训练恢复结果。每项任务均采用不含聊天格式的 5-shot 评估，并按照 OLMES 报告多项选择题准确率和完形填空准确率中较高者。Avg. 为九项任务的非加权平均值。所有数值均为百分比。

模型	ARC-C	ARC-E	BoolQ	常识-推理	HellaSwag	OpenBook QA	PIQA	社交 IQa	WinoGrande	平均值
原始 OLMoE	68.8	84.1	81.4	74.0	82.3	65.6	80.4	66.2	68.3	74.6
BRAID-half（SFT 前）	47.0	72.8	44.9	64.6	66.8	51.6	76.1	53.5	64.9	60.2
BRAID-half + SFT	58.4	82.1	81.7	71.8	73.9	59.6	77.4	62.2	64.7	70.2

推理过程中不会重新进行负载均衡。每个变体都使用其可用的内核融合（附录 C.1（融合））。解码使用 CUDA 图重放，以防止启动开销主导总执行时间。

通信重叠与资源争用。 通信–计算重叠的收益取决于可用独立计算的数量以及资源争用的程度。理想情况下，通信可以隐藏在独立计算之后，其最大重叠程度受通信持续时间和可用计算窗口二者中较短者的限制。然而，当操作竞争同一瓶颈资源时，并发执行并不会天然提升性能。通信与计算能够高效重叠，是因为它们主要依赖不同的资源：互连带宽与计算吞吐量或 HBM 带宽。尽管如此，资源争用依然存在。DeepEP 的分发和合并传输仅需少量 SM 即可执行通信内核，而其尾处理阶段还会在分发期间因词元复制和扩展、在合并期间因归约而额外消耗 SM 容量和内存带宽（Zhao 等人，2025年）。这种争用限制了可实现的重叠程度。因此，我们使用留出的提示（附录 C.2）针对每个变体分别调优传输和尾处理阶段的 SM 分配。

方法。 我们测量预填充延迟以及生成 32 个词元期间每个解码步骤的延迟。每个工作负载均在五个独立作业中重复执行，每个作业运行 30 轮，并在每轮中以随机顺序运行全部四种调度方案。我们报告作业中位数的平均值 ± 标准误。更多细节请参见附录 C.3。

6.2 结果：加速比与重叠

并行和梯形主要分别将分发与合并同注意力以及邻近的残差/归一化工作进行重叠。BRAID 还将早期路由进行重叠，并连续调度合并与分发，从而使两个通信阶段都能与注意力侧的计算重叠（图 2）。

我们首先考虑由 16 个提示组成的参考工作负载每个 GPU 处理 1,024 个词元（表 4）。对于专家并行=4，通信始终局限于单个节点内，并通过 NVLink 进行，其耗时足够短，使 BRAID 能够将合并与分发与计算大幅重叠。BRAID 达到1.171× 预填充和 1.182× 解码加速比，相较于顺序执行，预填充阶段合并/分发的重叠比例为 96.3%/92.8%，解码阶段为 89.2%/82.7%。因此，它明显优于并行和梯形。

对于 EP=8，通过 InfiniBand 进行的节点间通信明显长于可用的计算窗口。因此，大部分合并仍可与计算重叠（预填充阶段为 95.3%，解码阶段为 88.8%），但随后的分发仍大多处于暴露状态（重叠比例分别为 31.3% 和 5.8%）。尽管如此，BRAID 相较于顺序调度仍实现了1.178× 预填充和 1.278× 解码加速比。相比并行和梯形，其额外收益较为有限，因为这些调度方案同样可以将一个通信阶段进行重叠。具体而言，BRAID 在预填充阶段仅比并行高出 0.3%。

图5 和6 展示了批量大小 B 或上下文长度 L 变化时，专家并行=4 和专家并行=8的加速比，分别对应这两种情况。预填充和解码呈现出不同趋势，因为它们的计算和通信成本随规模变化的方式不同。附录 7 C 给出了具有代表性的专家并行=4 预填充执行轨迹，展示了不同的调度方案。

预填充。 在预填充阶段，通信量近似随词元数量线性增长 BL，而注意力计算量随 B 线性增长，并随 L呈二次增长。增加上下文长度 L 因此会增加可用于与通信重叠的计算量。对于专家并行=4，BRAID 的加速比从 1.12× （当 $L = 256$）提升至约1.16–1.17× （当 $L \geq 1024$），这是因为注意力计算在执行时间中所占的比例不断增加，并提供了更长的计算窗口来与节点内通信重叠（图5（b））。在固定L 的情况下改变批量大小 B 影响较弱，因为注意力、专家计算和通信量都近似随B线性增长，使它们的相对成本以及 BRAID 相对于顺序执行的加速比基本保持不变（图5（a））。对于专家并行=8，节点间通信仍显著长于可用的计算窗口，从而限制了能够被隐藏的通信比例

表 4. Qwen3-30B-A3B 在每个 GPU 上处理 16 个提示、每个提示含 1,024 个词元的推理结果（EP=4下全局批次为 64，EP=8下为 128）。延迟为五个作业的中位数平均值 ± 标准误（每个作业进行 30 轮测量，每轮按随机顺序测量全部四种调度方案）；解码延迟按步计算。合并与分发重叠是基于三次 Nsight Systems 跟踪得出的中位数，表示每个通信内核的执行时间中与同一 GPU 上计算内核并行的比例。加速比以顺序执行为基准；粗体标出最低延迟和最高加速比。

		预填充				解码			
拓扑	调度方案	延迟 (ms)	加速比	合并重叠	分发重叠	延迟（毫秒/步）	加速比	合并重叠	分发重叠
EP=4	顺序	291.9 ± 0.2	1.000×	0%	0%	11.13 ± 0.01	1.000×	0%	0%
	并行	262.5 ± 0.5	1.112×	0%	93.2%	10.42 ± 0.01	1.068×	0%	86.9%
	梯形	267.8 ± 0.2	1.090×	86.1%	11.6%	10.09 ± 0.01	1.103×	92.8%	11.1%
	BRAID	$\mathbf{249.2} \pm 0.4$	**1.171×**	96.3%	92.8%	$\mathbf{9.41} \pm 0.01$	**1.182×**	89.2%	82.7%
EP=8	顺序	479 ± 3	1.000×	0%	0%	13.45 ± 0.03	1.000×	0%	0%
	并行	408 ± 3	1.175×	0%	52.9%	11.01 ± 0.03	1.222×	0%	90.1%
	梯形	423 ± 3	1.133×	90.6%	2.6%	11.55 ± 0.04	1.165×	65.2%	4.3%
	BRAID	$\mathbf{407} \pm 2$	**1.178×**	95.3%	31.3%	$\mathbf{10.53} \pm 0.03$	**1.278×**	88.8%	5.8%

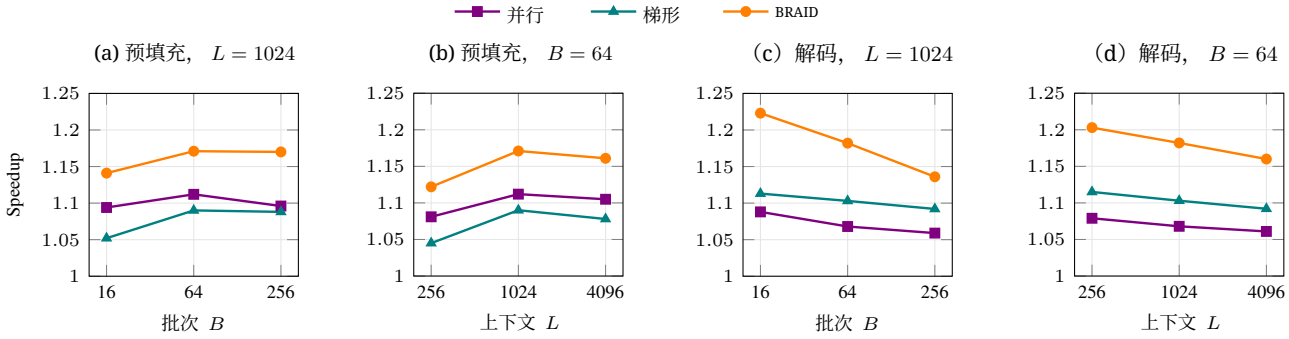

图 5。专家并行相对于顺序执行的加速比=4 （单个四块 H100 节点）。（a）预填充与全局批次大小的关系 B 在 $L = 1024$下；（b）预填充与提示长度的关系 L 在 $B = 64$下；（c）解码与 B 在 $L = 1024$下的关系；以及（d）解码与 L 在 $B = 64$下的关系。每条曲线代表一种执行调度；共享图例适用于全部四个面板。详细的延迟和重叠测量结果见附录 C.4 （表 10）。

。随着 L 增加，更长的注意力窗口使并行执行和 BRAID 能够重叠更多分发操作，而 Ladder 的合并操作在$L = 1024$时已经基本完成重叠，导致其加速比趋于饱和（图6（b））。在$L = 4096$时，BRAID 可实现 1.29×，相比之下，并行执行为 1.27×，Ladder 为 1.12× 。对于专家并行=4，增加B 对加速比的影响较弱（图6（a））。

解码。 解码会为每个请求处理一个新的词元，因此通信量仅取决于 B。在测试的工作负载下，通信在很大程度上受延迟限制，其时长主要由固定开销而非数据传输决定：它随 L 变化不大，并且主要在最大的专家并行=8批次（表 9）下随 B 增长；而解码注意力大致随 BL扩展。对于专家并行=4，计算窗口已经超过了较短的节点内通信时间，因此随着 B 或 L 增长，加速比会下降（图5(c)–(d)）。不过，BRAID 通过重叠两种集体通信，仍然保持领先。对于专家并行=8，较长的节点间通信超过了可用的计算窗口。增加 B 或L 可以让 BRAID 隐藏更多通信，也让 Ladder 隐藏更多合并操作；而 Parallel 的分发已经基本实现重叠，因此其加速比保持在接近 1.21×的水平。BRAID 的加速比从 1.22× 提升至约1.37× （图 6(c)–(d)），在测试的最大 B 和 L下，相较于 Parallel 和 Ladder 达到显著优势。

完整延迟、阶段计时和重叠结果见附录 C （表 9、10和11）。

7 结论

BRAID-MoE 将早期路由与梯形残差相结合，从而放宽顺序执行的合并–路由–分发依赖关系。通过根据合并前的残差计算专家分配，同时为专家计算保留新鲜激活值，BRAID 为专家并行通信与计算的重叠创造了新的机会。我们的实验表明，这种额外的并行性

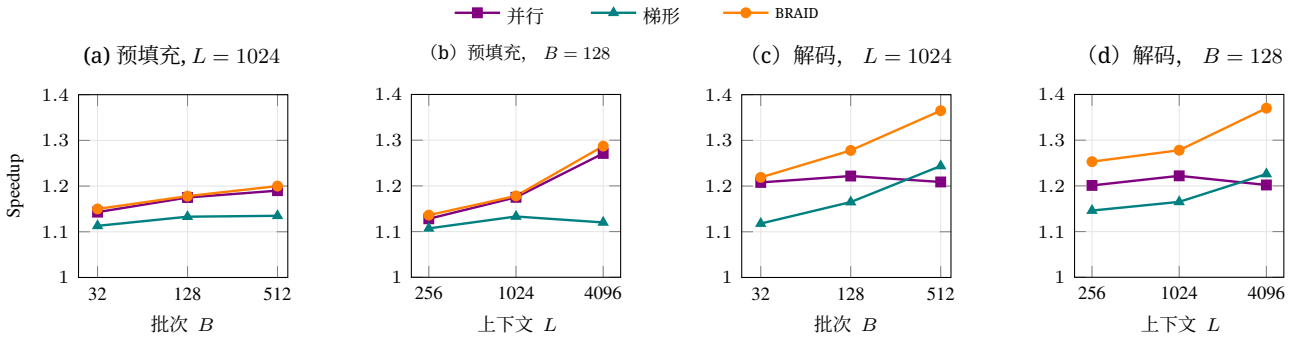

图6。 专家并行相对于顺序执行的加速比=8 （两个四卡H100节点）。(a) 预填充与全局批次大小的关系 B 在 $L = 1024$下；(b) 预填充与提示长度的关系 L 在 $B = 128$下；(c) 解码与 B 在 $L = 1024$下的关系；以及(d) 解码与 L 在 $B = 128$下的关系。每条曲线代表一种执行调度，共享图例适用于全部四个面板。详细的延迟和重叠测量结果见附录 C.4 （表 11）。

只会带来有限的质量损失。在预训练预算相同的情况下，对于 0.4B-A0.1B 和 2.5B-A0.5B，BRAID 的验证损失分别比顺序基线高出不超过 0.98% 和 0.49%。在预训练的顺序模型中，将 Transformer 层的后半部分转换为 BRAID，并仅进行 10% 的额外训练后，对于 0.4B-A0.1B 和 2.5B-A0.5B，验证损失分别与原始顺序检查点相差不超过 0.9% 和 0.3%。对于指令微调的 OLMoE-1B-7B 检查点，仅在 Infinity-Instruct 7M 和 Gen 子集的 5% 数据上进行监督微调，就能将转换后的平均下游准确率从 60.2% 提升至 70.2 %，而原始检查点为 74.6%。在 Qwen3-30B-A3B 上进行的受控推理实验中，BRAID 优于所有竞争调度方案：在我们的参考专家并行=4工作负载上，将预填充和解码分别加速 $1.17\times$ 和$1.18\times$；在两个节点（专家并行=8）上，最高可分别加速 $1.29\times$和 $1.37\times$ 。

这些结果表明，可以利用有限的路由陈旧性来改善通信与计算的重叠，同时基本保持模型质量。由此产生的合并与分发之间的邻接关系，为内核融合以及与张量并行集体通信进一步重叠创造了机会。探索这些优化仍是未来的研究方向。

致谢

JZ、DA 和 EO 获得了 PEPR IA SHARP 项目资助（ANR-23-PEIA-0008）的支持，并受益于通过 GENCI 分配的 AD011015884R1、AD011016641R1 和 AD011017766 配额使用 IDRIS 的 Jean Zay 高性能计算/人工智能资源。EB 感谢 NSERC Discovery 资助。

参考文献

Ahmed, D. 和 Oyallon, E. 知道你需要什么：高效的激活检查点技术。神经信息处理系统第四十届年度会议，2026年。

Ansel, J., Yang, E., He, H., Gimelshein, N., Jain, A., Voznesensky, M., Bao, B., Bell, P., Berard, D., Burovski, E., Chauhan, G., Chourdia, A., Constable, W., Desmai- son, A., DeVito, Z., Ellison, E., Feng, W., Gong, J., Gschwind, M., Hirsh, B., Huang, S., Kalambarkar, K., Kirsch, L., Lazos, M., Lezcano, M., Liang, Y., Liang, J., Lu, Y., Luk, C. K., Maher, B., Pan, Y., Puhrsch, C., Reso, M., Saroufim, M., Siraichi, M. Y., Suk, H., Zhang, S., Suo, M., Tillet, P., Zhao, X., Wang, E., Zhou, K., Zou, R., Wang, X., Mathews, A., Wen, W., Chanan, G., Wu, P., and Chintala, S. PyTorch 2：通过动态 Python 字节码转换和图编译实现更快的机器学习。收录于 *Proceedings of the29th ACM International Conference onArchitectural Sup-port for Programming Languagesand Operating Sys-tems, Volume 2*，ASPLOS '24，第929–947页。ACM，2024年4月。doi：10.1145/3620665.3640366。URLhttps://doi.org/10.1145/3620665.3640366。

Belilovsky, E.、Horoi, S.、Oyallon, E. 和 Th´erien, B. 内部优化器如何影响副本扩展。*arXiv* 预印本 *arXiv:2610.19064*，2026年。

Black, S., Biderman, S., Hallahan, E., Anthony, Q., Gao, L., Golding, L., He, H., Leahy, C., McDonell, K., Phang, J., Pieler, M., Prashanth, U. S., Purohit, S., Reynolds, L., Tow, J., Wang, B. 和 Weinbach, S. GPT-NeoX-20B：一种开源自回归语言模型。载于 Fan, A., Ilic, S., Wolf, T. 和 Gall´e, M.（编），《*BigScience* 第*5*期：创建大语言模型的挑战与展望》研讨会论文集，第95–136页，虚拟会议，+都柏林，2022年5月。计算语言学协会。doi: 10.18653/v1/2022.bigscience-1.9。

URL https://aclanthology.org/2022.bigscience-1.9/。

Cagnasso, P.、Belilovsky, E. 和 Oyallon, E. 统一大语言模型预训练中的局部通信与局部更新，2026a。网址https://arxiv.org/abs/2606.11081。

Cagnasso, P.、Kopyshevskiy, D.、Oyallon, E. 和 Belilovsky, E. Spice：用于大语言模型预训练的稀疏参数 Gossip，2026b。网址 https://arxiv.org/abs/2611.23053。

Cai, W.、Jiang, J.、Qin, L.、Cui, J.、Kim, S. 和 Huang, J. 用于加速混合专家的快捷连接专家并行。在 Singh, A.、Fazel, M.、Hsu, D.、Lacoste-Julien, S.、Berkenkamp, F.、Maharaj, T.、Wagstaff, K. 和 Zhu, J.（编），第*42*届国际机器学习会议论文集，第267卷，机器学习研究论文集，第6211–6228页。PMLR，2025年7月13日至19日。网址 https://proceedings.mlr.press/v267/cai25c.html。

Chang, L.-W., Bao, W., Hou, Q., Jiang, C., Zheng, N., Zhong, Y., Zhang, X., Song, Z., Yao, C., Jiang, Z., Lin, H., Jin, X. 和 Liu, X. FLUX：通过内核融合在 GPU 上实现基于软件的快速通信重叠，2024年。网址 https://arxiv.org/abs/2406.06858。

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pel- lat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S. 和 Fiedel, N. PaLM：利用 Pathways 扩展语言建模。 机器学习研究期刊，24(240):1–113，2023年。网址 http://jmlr.org/papers/v24/22-1144.html。

Cui, C., Jackson, R., Herrera, B. J., T´arano, A. M., 和 Kerner, H. 多头 LatentMoE 与头并行：通信高效且确定性的混合专家并行。在 Zhang, T., Dudik, M., Jaggi, M., Agarwal, A., Li, S., Schuurmans, D., Zhu, J., Berkenkamp, F., Dong, H., 和 Bietti, A.（编），第*43*届国际机器学习会议论文集，第306卷机器学习研究论文集，第21847–21863页。PMLR，2026年7月6日至11日。网址 https://proceedings.mlr.press/v306/cui26c.html。

DeepSeek-AI, Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Guo, D., Yang, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Bao, H., Xu, H., Wang, H., Zhang, H., Ding, H., Xin, H., Gao, H., Li, H., Qu, H., Cai, J. L., Liang, J., Guo, J., Ni, J., Li, J., Wang, J., Chen, J., Chen, J., Yuan, J., Qiu, J., Li, J., Song, J., Dong, K., Hu, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Xu, L., Xia, L., Zhao, L., Wang, L., Zhang, L., Li, M., Wang, M., Zhang, M., Zhang, M., Tang, M., Li, M., Tian, N., Huang, P., Wang, P., Zhang, P., Wang, Q., Zhu, Q., Chen, Q., Du, Q., Chen, R. J., Jin, R. L., Ge, R., Zhang, R., Pan, R., Wang, R., Xu, R., Zhang, R., Chen, R., Li, S. S., Lu, S., Zhou, S., Chen, S., Wu, S., Ye, S., Ye, S., Ma, S., Wang, S., Zhou, S., Yu, S., Zhou, S., Pan, S., Wang, T., Yun, T., Pei, T., Sun, T., Xiao, W. L., Zeng, W., Zhao, W., An, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Li, X. Q., Jin, X., Wang, X., Bi, X., Liu, X., Wang, X., Shen, X., Chen, X., Zhang, X., Chen, X., Nie, X., Sun, X., Wang, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yu, X., Song, X., Shan, X., Zhou, X., Yang, X., Li, X., Su, X., Lin, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhu, Y. X., Zhang, Y., Xu, Y., Xu, Y., Huang, Y., Li, Y., Zhao, Y., Sun, Y., Li, Y., Wang, Y., Yu, Y., Zheng, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Tang, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Wu, Y., Ou, Y., Zhu, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Zha, Y., Xiong, Y., Ma, Y., Yan, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Wu, Z. F., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Huang, Z., Zhang, Z., Xie, Z., Zhang, Z., Hao, Z., Gou, Z., Ma, Z., Yan, Z., Shao, Z., Xu, Z., Wu, Z., Zhang, Z., Li, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Gao, Z., and Pan, Z. Deepseek-v3技术报告，2025年。URLhttps://arxiv.org/abs/2412.19437。

Dukler, Y.、Li, G.、Shah, D.、Liu, J.、Appia, V. 和 Barsoum, E. Farskip-collective：在混合专家模型中解除阻塞通信的限制。载于 Chowdhery, A. 和 Jia, Z.（编），机器学习与系统论文集，第 8 卷，第 1803–1816 页。MLSys，2026年。网址：https://proceedings.mlsys.org/paper_files/paper/2026/file/6feb9b30798abcfae937760d183605e1-Paper-Conference.pdf。

Elango, V., Bhatia, N., Waleffe, R., Shafipour, R., Asida, T., Khattar, A., Assaf, N., Golub, M., Guman, J., Mitra, T., Zhao, R., Borkar, R., Zilberstein, R., Patwary, M., Shoeybi, M. 和 Darvish Rouhani, B. 重新审视混合专家架构中的效率—准确率扩展。

。见 Zhang, T., Dudik, M., Jaggi, M., Agarwal, A., Li, S., Schuurmans, D., Zhu, J., Berkenkamp, F., Dong, H. 和 Bietti, A.（编），第*43*届国际机器学习会议论文集，第306卷，机器学习研究论文集，第27809–27821页。PMLR，2026年7月6日至11日。网址：https://proceedings.mlr.press/v306/elango26a.html。

Fedus, W.、Zoph, B. 和 Shazeer, N. Switch Transformer：以简单高效的稀疏性扩展至万亿参数模型。机器学习研究期刊，23(120):1–39，2023年。网址http://jmlr.org/papers/v23/21-0998.html。

Gu, Y.、Tafjord, O.、Kuehl, B.、Haddad, D.、Dodge, J. 和 Hajishirzi, H. OLMES：语言模型评测标准。载于 Chiruzzo, L.、Ritter, A. 和 Wang, L.（编），计算语言学协会年会论文集：*NAACL 2025*，第5020–5048页，新墨西哥州阿尔伯克基，2025年4月。计算语言学协会。ISBN 979-8-89176-195-7。doi：10.18653/v 1/2025.findings-naacl.282。网址https://aclanthology.org/2025.findings-naacl.282/。Guo, W.、Mishra, M.、Cheng, X.、Stoica, I. 和 Dao, T. Sonicmoe：通过输入输出和磁贴感知优化加速混合专家。在 Vondrick, C.、Hariharan, B.、Raffel, C.、Pinto, L.、Yang, D. 和 Faust, A.（编），国际学习表征会议，第2026卷，第67814–67839页，2026年。网址https://proceedings.iclr.cc/paper_files/paper/2026/file/6ed5bf446f59e2c6646d23058c86424b-Paper-Conference.pdf。Hwang, C.、Cui, W.、Xiong, Y.、Yang, Z.、Liu, Z.、Hu, H.、Wang, Z.、Salas, R.、Jose, J.、Ram, P.、Chau, H.、Cheng, P.、Yang, F.、Yang, M. 和 Xiong, Y. Tutel：大规模自适应混合专家。在 Song, D.、Carbin, M. 和 Chen, T.（编），机器学习与系统论文集，第5卷，第269–287页。Curan，2023年。网址https://proceedings.mlsys.org/paper_files/paper/2023/file/5616d34cf8ff73942cfd5aa922842556-Paper-mlsys2023.pdf。Hwang, R.、Wei, J.、Cao, S.、Hwang, C.、Tang, X.、Cao, T. 和 Yang, M. Pre-gated MoE：用于快速且可扩展混合专家推理的算法—系统协同设计。载于*2024*年第*51*届*ACM/IEEE*计算机体系结构国际年会（*ISCA*），第1018–1031页，2024年。doi：10.1109/ISCA59077.2024.00078。Janson, P.、Oyallon, E. 和 Belilovsky, E. 面向通信高效流水线并行的学习型子空间压缩。*arXiv*预印本 *arXiv:2606.05484*，2026年a。Janson, P.、Oyallon, E. 和 Belilovsky, E. 稳定原生低秩大语言模型预训练。载于第*43*届国际机器学习大会，《机器学习研究论文集》第306卷，第50830–50852页。PMLR，2026年7月6日至11日b。网址：https://proceedings.mlr.press/v306/janson26a.html。Jiang, C.、Tian, Y.、Jia, Z.、Zheng, S.、Wu, C.和Wang, Y. Lancet：通过全图计算通信重叠加速混合专家训练。载于Gibbons, P.、Pekhimenko, G.和Sa, C. D.（编），机器学习与系统论文集，第6卷，第74–86页，2024年。网址：https://proceedings.mlsys.org/paper_files/paper/2024/file/339caf45a6fa281cae8adc6465343464-Paper-Conference.pdf。Jordan, K.、Jin, Y.、Boza, V.、You, J.、Cesista, F.、Newhouse, L.和Bernstein, J. Muon：神经网络隐藏层的优化器，2024年。网址：https://kellerjordan.github.io/posts/muon/。Kopyshevskiy, D.、Th´erien, B.、Pinto, N.和Oyallon, E. LACCO：通过计算–通信重叠实现局部累积。工作论文或预印本，2026年11月。网址：https://hal.science/hal-05862888。Kwon, W.、Li, Z.、Zhuang, S.、Sheng, Y.、Zheng, L.、Yu, C. H.、Gonzalez, J.、Zhang, H.和Stoica, I. 通过分页注意力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，SOSP '23，第611–626页，美国纽约州纽约市，2023年。美国计算机协会。ISBN 9798400702297。doi：10.1145/3600006.3613165。网址：https://doi.org/10.1145/3600006.3613165。Lepikhin, D.、Lee, H.、Xu, Y.、Chen, D.、Firat, O.、Huang, Y.、Krikun, M.、Shazeer, N.和Chen, Z. {GS}hard：通过条件计算和自动分片扩展巨型模型。载于国际学习表征会议，2021年。网址：https://openreview.net/forum?id=qrwe7XHTmYb。Li, J.、Jiang, Y.、Zhu, Y.、Wang, C.和Xu, H. 使用lina加速分布式混合专家训练与推理。载于*2023*年*USENIX*年度技术会议（*USENIX ATC 23*），第945–959页，美国马萨诸塞州波士顿，2023年7月。USENIX协会。ISBN 978-1-939133-35-9。网址：https://www.usenix.org/conference/atc23/presentation/li-jiamin。Liang, W.、Liu, T.、Wright, L.、Constable, W.、Gu, A.、Huang, C.-C.、Zhang, I.、Feng, W.、Huang, H.、Wang, J.、Purandare, S. J.、Nadathur, G.和Idreos, S. Torchtitan：面向生产级大语言模型预训练的一站式PyTorch原生解决方案。载于Yue, Y.、Garg, A.、Peng, N.、Sha, F.、

和Yu, R.（编），国际学习表征会议，第2025卷，第92161–92181页，2025年。网址：https://proceedings.iclr.cc/paper_files/paper/2025/file/e6231c5f46598cfd09ff1970524e0436-Paper-Conference.pdf。

Liu, J.、Su, J.、Yao, X.、Jiang, Z.、Lai, G.、Du, Y.、Qin, Y.、Xu, W.、Lu, E.、Yan, J.、Chen, Y.、Zheng, H.、Liu, Y.、Liu, S.、Yin, B.、He, W.、Zhu, H.、Wang, Y.、Wang, J.、Dong, M.、Zhang, Z.、Kang, Y.、Zhang, H.、Xu, X.、Zhang, Y.、Wu, Y.、Zhou, X. 和 Yang, Z. Muon 对大语言模型训练具有可扩展性，2025年。URLhttps://arxiv.org/abs/2502.16982。

Loshchilov, I. 和 Hutter, F. SGDR：随机梯度下降与热重启。在国际学习表征会议，2017年。URL https://openreview.net/forum?id=Skq89Scxx。

Loshchilov, I. 和 Hutter, F. 解耦权重衰减正则化。在国际学习表征会议，2019年。网址：https://openreview.net/forum?id=Bkg6RiCqY7。

Ludziejewski, J.、Pi´oro, M.、Krajewski, J.、Stefaniak, M.、Krutul, M.、Mała´snicki, J.、Cygan, M.、Sankowski, P.、Adamczewski, K.、Miło´s, P. 和 Jaszczur, S. 联合混合专家缩放定律：混合专家可以实现高效内存使用。载于 Singh, A.、Fazel, M.、Hsu, D.、Lacoste-Julien, S.、Berkenkamp, F.、Maharaj, T.、Wagstaff, K. 和 Zhu, J.（编），第*42*届国际机器学习大会论文集，第267卷，收录于机器学习研究论文集，第41056–41073页。PMLR，2025年7月13日至19日。网址：https://proceedings.mlr.press/v267/ludziejewski25a.html。

Muennighoff, N.、Soldaini, L.、Groeneveld, D.、Lo, K.、Mor- rison, J.、Min, S.、Shi, W.、Walsh, P.、Tafjord, O.、Lambert, N.、Gu, Y.、Arora, S.、Bhagia, A.、Schwenk, D.、Wadden, D.、Wettig, A.、Hui, B.、Dettmers, T.、Kiela, D.、Farhadi, A.、Smith, N.、Koh, P. W.、Singh, A. 和 Hajishirzi, H.《Olmoe：开放式混合专家语言模型》。载于 Yue, Y.、Garg, A.、Peng, N.、Sha, F. 和 Yu, R.（编），国际学习表征会议，第2025卷，第62061–62121页，2025年。网址：https://proceedings.iclr.cc/paper_files/paper/2025/file/9b224ace8963c9385ad5e2b5c9039b97-Paper-Conference.pdf。

Nabli, A., Fournier, L., ERBACHER, P., Serrano, L., Belilovsky, E., 和 Oyallon, E. Acco：在通信的同时进行累积，用于通信重叠的分片大语言模型训练。载于 Belgrave, D., Zhang, C., Lin, H., Pas- canu, R., Koniusz, P., Ghassemi, M., 和 Chen, N.（编），神经信息处理系统进展，第 38 卷，主会议，第 69278–69305 页。Curran Associates, Inc., 2025年。doi：10.52202/085713-2329。网址：https://proceedings.neurips.cc/paper_files/paper/2025/file/6454dcd80b5373daaa97e53ce32c78a1-Paper-Conference.pdf。

Penedo, G., Kydl´ı˘cek, H., allal, L. B., Lozhkov, A., Mitchell, M., Raffel, C., Von Werra, L. 和 Wolf, T. The fineweb datasets：以规模化方式提炼网络中的高质量文本数据。在 Globerson, A.、Mackey, L.、Belgrave, D.、Fan, A.、Paquet, U.、Tomczak, J. 和 Zhang, C.（编），神经信息处理系统进展，第37卷，第30811–30849页。Curran Associates, Inc.，2024年。doi：10.52202/079017- 0970。URLhttps://proceedings.neurips.cc/paper_files/paper/2024/file/370df50ccfdf8bde18f8f9c2d9151bda-Paper-Datasets_and_基准_Track.pdf。

Qiu, Z.、Huang, Z.、Cheng, S.、Zhou, Y.、wang, z.、Titov, I. 和 Fu, J. 逐层循环路由器用于混合专家。在 Yue, Y.、Garg, A.、Peng, N.、Sha, F. 和 Yu, R.（编）国际学习表征会议，第2025卷，第14179–14205页，2025年。网址：https://proceedings.iclr.cc/paper_files/paper/2025/file/250a6c6a7a7216f5dec1364e9a93abf4-Paper-Conference.pdf。

Qiu, Z.、Wang, Z.、Li, X.、Li, Y.、Xu, Y.、Wang, Y.、Zhang, H.、Men, R.、Mao, B.、Zhang, C.、Zhou, F.、Luo, H.、Huang, H.、Lian, H.、Huang, H.、Chen, H.、Zhang, J.、Xu, J.、Wang, J.、Chen, L.、Wang, L.、Jiang, L.、Yuan, M.、Sun, M.、Jin, P.、Zhang, S.、Wang, S.、Ren, X.、Wang, Y.、Zhang, Y.、Dong, Y.、Cao, Y.、Ma, Y.、Mao, Y.、Zheng, B. 和 Liu, D. Qwen3.8-Next架构的设计：评估、效率与训练稳定性，2026年。网址：https://arxiv.org/abs/2608.30320。

Rivaud, S., Fournier, L., Pumir, T., Belilovsky, E., Eick- enberg, M. 和 Oyallon, E. Petra：采用可逆架构进行端到端并行训练。在 Yue, Y., Garg, A., Peng, N., Sha, F. 和 Yu, R.（编），国际学习表征会议，第2025 卷，第 80162–80178 页，2025年。网址https://proceedings.iclr.cc/paper_files/paper/2025/file/c710d6b4507e70c6332bee871b8d1ca5-Paper-Conference.pdf。

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G. 和 Dean, J. 极其庞大的神经网络：稀疏门控混合专家层。在 国际学习表征会议，2017年。网址https://openreview.net/forum?id=B1ckMDqlg。

Su, J.、Ahmed, M.、Lu, Y.、Pan, S.、Bo, W. 和 Liu, Y. Roformer：增强型变换器与旋转位置

嵌入。 *Neurocomputing*，568:127063，2024年。ISSN 0925-2312。doi: https://doi.org/10.1016/j.neucom.202 3.127063。URLhttps://www.sciencedirect.com/science/article/pii/S0925231223011864。

Team, K., Bai, T., Bai, Y., Bao, Y., C., M., Cai, J., Cai, X., Cao, P., Cao, Y., Chai, Z., Charles, Y., Che, H. S., Chen, G., Chen, G., Chen, G., Chen, H., Chen, J., Chen, J., Chen, J., Chen, K., Chen, P., Chen, R., Chen, W., Chen, X., Chen, Y., Chen, Y., Chen, Y., Chen, Y., Chen, Y., Chen, Y., Chen, Y., Chen, Z., Cheng, D., Cheng, Y., Cui, J., Cui, J., Dai, A., Deng, J., Ding, H., Ding, R., Ding, S., Dong, M., Dong, M., Dong, Y., Dong, Y., Du, A., Du, C., Du, D., Du, J., Du, Y., Fan, Y., Feng, J., Feng, Q., Feng, Y., Fu, K., Fu, Q., Gao, F., Gao, H., Gao, J., Gao, T., Gao, W., Geng, S., Gong, J., Gong, L., Gong, S., Gong, X., Gu, Q., Gu, Y., Guan, S., Guo, H., Guo, S., Guo, X., Guo, Z., Hao, B., Hao, W., Hao, X., He, D., He, H., He, L., He, Q., He, W., He, X., He, X., He, Y., He, Y., Hong, C., Hong, T., Hu, H., Hu, J., Hu, R., Hu, W., Hu, Y., Hu, Z., Hua, L., Huang, J., Huang, K., Huang, R., Huang, S., Huang, W., Huang, Y., Huang, Z., Huang, Z., Hui, Y., Jia, C., Jiang, Y., Jiang, Z., Jiang, Z., Jin, W., Jin, X., Jing, Y., Kong, H., Lai, G., Li, A., Li, C., Li, C., Li, C., Li, F., Li, G., Li, H., Li, J., Li, J., Li, L., Li, L., Li, L., Li, W., Li, W., Li, X., Li, Y., Li, Y., Li, Y., Li, Y., Li, Z., Li, Z., Li, Z., Li, Z., Li, Z., Lin, J., Lin, X., Lin, Y., Lin, Z., Lin, Z., Liu, B., Liu, B., Liu, C., Liu, L., Liu, S., Liu, S., Liu, S., Liu, T., Liu, W., Liu, Y., Liu, Y., Liu, Y., Liu, Y., Liu, Z., Liu, Z., Lu, E., Lu, H., Lu, L., Lu, T., Lu, Z., Luo, A., Luo, G., Luo, J., Luo, Y., Lyu, B., Lyu, W., Mao, S., Mei, Y., Men, X., Ni, M., Niu, Y., Pan, S., Peng, S., Qi, Z., Qin, R., Qin, Z., Qin, Z., Qiu, H., Qiu, J., Qiu, J., Qu, B., Qu, Y., Shang, Z., Shao, Y., Shen, H., Shi, J., Shi, J., Shi, L., Shi, S., Siu, W., Song, P., Song, X., Su, J., Su, Y., Su, Z., Sui, L., Sun, J., Sun, J., Sun, S., Sun, S., Sun, T., Sun, Y., Tai, Y., Tang, C., Tang, H., Tang, S., Tang, Z., Tian, C., Tian, R., Tian, Y., Tu, W., Wang, C., Wang, C., Wang, C., Wang, D., Wang, F., Wang, H., Wang, H., Wang, H., Wang, H., Wang, H., Wang, H., Wang, J., Wang, J., Wang, J., Wang, J., Wang, L., Wang, S., Wang, S., Wang, S., Wang, S., Wang, S., Wang, T., Wang, W., Wang, X., Wang, X., Wang, X., Wang, X., Wang, Y., Wang, Y., Wang, Y., Wang,Y., Wang,Y., Wang, Y., Wang, Y., Wang, Y., Wang, Z., Wang, Z., Wang, Z., Wang, Z., Wang, Z., Wang, Z., Wei, C., Wei, M., Wei, S., Wen, Z., Wu, F., Wu, H., Wu, R., Wu, W., Wu, X., Wu, Y., Wu, Y., Wu, Y., Wu, Z., Xian, X., Xiang, C., Xiang, Y., Xiao, B., Xiao, C., Xiao, X., Xie, J., Xie, X., Xie, Y., Xie, Z., Xing, B., Xiong, Y., Xu, B., Xu, B., Xu, J., Xu, J., Xu, J., Xu, J., Xu, L. H., Xu, Q., Xu, S., Xu, S., Xu, T., Xu, T., Xu, W., Xu, X., Xu, Y., Xu, Y., Xu, Y., Xu, Z., Xue, H., Yan, J., Yan, Y., Yang, F., Yang, G., Yang,】【。

H.、Yang, J.、Yang, R.、Yang, W.、Yang, X.、Yang, X.、Yang, Y.、Yang, Y.、Yang, Y.、Yang, Y.、Yang, Z.、Yang, Z.、Yang, Z.、Yang, Z.、Yao, H.、Ye, D.、Ye, H.、Ye, W.、Ye, Z.、Yin, B.、Yin, H.、Yin, X.、Yu, C.、Yu, H.、Yu, L.、Yu, S.、Yu, S.、Yu, T.、Yuan, E.、Yuan, M.、Yue, T.、Yue, W.、Yue, Y.、Zha, D.、Zhan, H.、Zhang, B. H.、Zhang, D.、Zhang, F.、Zhang, H.、Zhang, H.、Zhang, H.、Zhang, J.、Zhang, J.、Zhang, J.、Zhang, K.、Zhang, M.、Zhang, P.、Zhang, Q.、Zhang, R.、Zhang, R.、Zhang, S.、Zhang, S.、Zhang, X.、Zhang, X.、Zhang, Y.、Zhang, Y.、Zhang, Y.、Zhang, Y.、Zhang, Y.、Zhang, Y.、Zhang, Y.、Zhang, Y.、Zhang, Y.、Zhang, Z.、Zhang, Z.、Zhao, B.、Zhao, C.、Zhao, F.、Zhao, J.、Zhao, J.、Zhao, S.、Zhao, W.、Zhao, X.、Zhao, X.、Zhao, Y.、Zhao, Z.、Zheng, H.、Zheng, H.、Zheng, R.、Zheng, S.、Zheng, T.、Zhong, H.、Zhong, L.、Zhong, L.、Zhou, M.、Zhou, Q.、Zhou, R.、Zhou, R.、Zhou, X.、Zhou, Y.、Zhou, Z.、Zhu, J.、Zhu, L.、Zhu, X.、Zhu, Y.、Zhu, Y.、Zhu, Z.、Zhuang, C.、Zhuang, W.、以及 Zu, X. Kimi K3：开放前沿智能，2026年。URLhttps://arxiv.org/abs/2607.24653。Touvron, H.、Martin, L.、Stone, K.、Albert, P.、Almahairi, A.、Babaei, Y.、Bashlykov, N.、Batra, S.、Bhargava, P.、Bhosale, S.、Bikel, D.、Blecher, L.、Ferrer, C. C.、Chen, M.、Cucurull, G.、Esiobu, D.、Fernandes, J.、Fu, J.、Fu, W.、Fuller, B.、Gao, C.、Goswami, V.、Goyal, N.、Hartshorn, A.、Hosseini, S.、Hou, R.、Inan, H.、Kardas, M.、Kerkez, V.、Khabsa, M.、Kloumann, I.、Korenev, A.、Koura, P. S.、Lachaux, M.-A.、Lavril, T.、Lee, J.、Liskovich, D.、Lu, Y.、Mao, Y.、Martinet, X.、Mihaylov, T.、Mishra, P.、Molybog, I.、Nie, Y.、Poulton, A.、Reizenstein, J.、Rungta, R.、Saladi, K.、Schelten, A.、Silva, R.、Smith, E. M.、Subramanian, R.、Tan, X. E.、Tang, B.、Taylor, R.、Williams, A.、Kuan, J. X.、Xu, P.、Yan, Z.、Zarov, I.、Zhang, Y.、Fan, A.、Kambadur, M.、Narang, S.、Rodriguez, A.、Stojnic, R.、Edunov, S.、以及 Scialom, T. Llama 2：开放基础模型与微调聊天模型，2023年。URL https://arxiv.org/abs/2307.09288。vLLM 项目。双批次重叠。https://docs.vllm.ai/en/latest/design/dbo/，2026年。vLLM 文档。访问日期：2026年10月30日。Yang, A.、Li, A.、Yang, B.、Zhang, B.、Hui, B.、Zheng, B.、Yu, B.、Gao, C.、Huang, C.、Lv, C.、Zheng, C.、Liu, D.、Zhou, F.、Huang, F.、Hu, F.、Ge, H.、Wei, H.、Lin, H.、Tang, J.、Yang, J.、Tu, J.、Zhang, J.、Yang, J.、Yang, J.、Zhou, J.、Zhou, J.、Lin, J.、Dang, K.、Bao, K.、Yang, K.、Yu, L.、Deng, L.、Li, M.、Xue, M.、Li, M.、Zhang, P.、Wang, P.、Zhu, Q.、Men, R.、Gao, R.、Liu, S.、Luo, S.、Li, T.、Tang, T.、Yin, W.、Ren, X.、Wang, X.、Zhang, X.、Ren, X.、Fan, Y.、Su, Y.、Zhang, Y.、Zhang, Y.、Wan, Y.、Liu, Y.、Wang, Z.、Cui, Z.、Zhang, Z.、Zhou, Z.、以及 Qiu, Z. Qwen3 技术报告，2025年。URL https://arxiv.org/abs/2505.09388。

Zhang, M.、Mishra, M.、Zhou, Z.、Brandon, W.、Wang, J.、Kim, Y.、Ragan-Kelley, J.、Song, S. L.、Athiwaratkun, B.、以及 Dao, T. 梯形残差：用于通过通信重叠加速大模型推理的并行性感知架构。载于 Singh, A.、Fazel, M.、Hsu, D.、Lacoste-Julien, S.、Berkenkamp, F.、Maharaj, T.、Wagstaff, K.、以及 Zhu, J.（编），第*42*届国际机器学习大会论文集，第267卷，机器学习研究论文集，第75779–75792页。PMLR，2025年7月13日至19日。URL https://proceedings.mlr.press/v267/zhang25bg.html。

Zhang, S., Zheng, N., Lin, H., Jiang, Z., Bao, W., Jiang, C., Hou, Q., Cui, W., Zheng, S., Chang, L.-W., Chen, Q., and Liu, X. Comet：面向混合专家的细粒度计算与通信重叠。载于 Zaharia, M., Joshi, G., and Lin, Y.（编），机器学习与系统论文集，第 7 卷。MLSys，2025b。网址：https://proceedings.mlsys.org/paper_files/paper/2025/file/e27ea0cd50b798ff8942caf9203f0992-Paper-Conference.pdf。

Zhao, C., Zhou, S., Zhang, L., Deng, C., Xu, Z., Liu, Y., Yu, K., Li, J., and Zhao, L. DeepEP：一种高效的专家并行通信库。https://github.com/deepseek-ai/DeepEP，2025年。

人工智能使用披露

使用 ChatGPT 协助进行文献综述、稿件撰写和编辑。使用 Codex 和 Claude Code 生成并审查代码。

表*5*。用于受控实验的 Qwen3 风格模型配置。

规模	层	d	Q/KV 头部	专家宽度	专家
0.4B-A0.1B	14	576	9/1	216	64
2.5B-A0.5B	24	1152	18/2	432	64

A 实验硬件

所有实验均使用同一集群中的四 GPU H100 SXM5 节点（每块 GPU 配备 80 GB HBM3 和 132 个 SM）。每块 GPU 有 18 条 NVLink 4 链路，其中连接到节点内其他每块 GPU 的链路各有六条；每对 GPU 的双向带宽最高可达 300 GB/s（每个方向 150 GB/s），每块 GPU 的总双向带宽为 900 GB/s。节点间通信通过每个节点上的四块 ConnectX-7 NDR400 InfiniBand 适配器实现，每块适配器的额定带宽为每个方向 50 GB/s（每个节点每个方向 200 GB/s）。预训练和检查点延续使用一个四 GPU 节点；后训练恢复使用两块 GPU；推理则针对 EP=4 使用一个节点，或针对 EP=8 使用两个节点。

B 架构与训练细节

B.1 架构与初始化

我们的 0.4B-A0.1B 和 2.5B-A0.5B Qwen3 风格模型保留了 Qwen3-MoE（Yang 等人，2025年）的部分比例。我们保持注意力头维度 $d_h = 128$、总查询宽度 $h_q d_h = 2d$和专家中间层宽度 $m = 3d/8$不变。通过选择深度和模型宽度，使其达到目标激活参数规模。两个模型均采用 64 个路由专家、Top-8 路由，且不使用共享专家。表 5总结了它们的几何参数。

骨干网络采用前置 RMSNorm，使用 $\epsilon = 10^{-6}$、Q/K RMS 归一化、SwiGLU 专家和旋转位置嵌入（RoPE）（Su 等人，2024年），基数为 10^6。输入嵌入和语言模型头不共享权重，词汇表包含 32,000 个词元。路由器对所有专家执行 Softmax，再选出 Top-8，并对所选概率重新归一化，使其总和为 1。在 BRAID 中，早期路由器使用独立的 RMSNorm，与用于新鲜专家输入的归一化不同。

我们遵循 TorchTitan 的 Qwen3 式初始化（Liang 等人，2025年）。嵌入权重、语言模型头、注意力输入投影、路由器权重以及专家输入/上投影均从 $\mathcal{N}(0, 0.02_2)$ 初始化。注意力输出投影和专家下投影使用标准差 $0.02/\sqrt{2N_{\text{layers}}}$，其中 N_{layers}为 Transformer 层的数量。所有归一化权重，包括单独的早期路由器归一化，均被初始-

化为 1。

B.2 模型—令牌分配与数据流 ms

对于第 5.1 节中的预训练实验，我们采用联合 MoE 缩放定律方案（Ludziejewski 等人，2025年）来确定训练预算。尽管原始拟合假设了不同的路由配置，但在较小规模上进行的局部等 FLOP 扫描表明，在测试的 75M、100M 和 125M 配置中，100M 模型表现最佳。因此，我们在 2.5B-A0.5B 上采用相同的方案，将其视为一种可复现的预算规则，而非全局计算最优性的证据。由此得到的训练时长分别为 3,123 步和 9,974 个优化步骤。

数据流。 我们从 FineWeb v1.4.0 的 CC-MAIN-2022-05 通用爬取快照中提取预训练和验证词元（Penedo 等人，2024年），该数据集会对每个快照中的文档进行去重。训练集和留出的验证集使用互不相交的 Parquet 文件，以防止两组数据之间出现重叠。

B.3 预训练优化

我们的预训练设置总体遵循Ahmed 和 Oyallon(2026年)，并针对混合专家训练进行了调整。注意力和专家矩阵使用 Muon（Jordan 等人，2024年），动量为 0.95，使用 Nesterov 动量和五次 Newton–Schulz 迭代。我们以更新均方根缩放（Liu 等人，2025年）替代 Muon 原本的更新缩放。嵌入、语言模型头、路由器和归一化参数使用 AdamW（Loshchilov 和 Hutter，2019年），其 $(\beta_1,\ \beta_2) = (0.9,\ 0.95)$。遵循Qiu 等人 (2026年）的研究结论：将 Muon 应用于混合专家路由器会加剧训练早期的波动，却不会带来显著的性能提升，因此我们对路由器参数保留 AdamW。归一化参数不进行权重衰减，梯度裁剪至全局范数 1.0。负载均衡使用系数为 10^{-3} 的专家偏置更新。不使用可微的负载均衡损失，也不丢弃词元。

学习率先进行 5% 的线性预热，随后采用余弦衰减（Loshchilov 和 Hutter，2017年），降至峰值学习率的 10%。我们使用顺序基准为每个规模选择峰值学习率，在 $\{1, 5, 7\} \times 10^{j}$ 网格上搜索。随后，在同一规模的不同架构中，选定的学习率和权重衰减保持不变。在 0.4B-A0.1B 规模下，Muon 和 AdamW 的峰值学习率均为 7×10^{-3}，在总计 3,123 步的调度中包含 156 个预热步数。在 2.5B-A0.5B 规模下，两者的峰值学习率均为 5×10^{-3}，在总计 9,974 步中包含 498 个预热步数。两个规模的权重衰减均为 0.1。所有训练运行均使用 BF16。

B.4 检查点延续

对于第 5.2 节中的检查点适应实验，在 0.4B-A0.1B 规模下，BRAID 使用的峰值学习率为 $2.1 \times 10^{-3}(0.3\times$ 预训练峰值)，顺序控制使用 $7 \times10^{-4}\,(0.1\times)$，两者均采用 15 个预热步数。在 2.5B-A0.5B 规模下，BRAID 使用的峰值学习率为 $1.5 \times 10^{-3}\,(0.3\times$ 预训练峰值)，顺序控制使用 $5\times10^{-4}\,($ $0.1\times)$，两者均采用 49 个预热步数。两种续训方案的权重衰减均为 0.1，并采用余弦衰减，将学习率降至峰值的 10%。续训数据取自 FineWeb v1.4.0 的 CC-MAIN-2022-05 快照中的独立 Parquet 文件，与预训练数据和留出验证集数据互不重叠。

B.5 OLMoE 监督微调

对于第 5.3 节中的训练后恢复实验，采样语料库包含 444,913 段对话，同时保留原始子集比例和多轮对话。每个周期约 205.8M 个词元中，有 138.3M 个助手令牌计入训练损失。我们使用 4,096 的最大序列长度、零权重衰减，并将全局范数的梯度裁剪阈值设为 1.0。训练使用 BF16 和 DeepSpeed ZeRO-3。

C 推理实现与性能分析细节

本附录介绍了第 6 节背后的执行、工作负载和性能分析选择，随后对观测到的加速比进行详细拆解。

C.1 实现与实验控制

执行引擎。 这四种调度方案实现为自定义的 vLLM（Kwon 等人，2023年）模型类，其前向传播运行我们的双流调度代码。vLLM 异步调度请求、管理分页键值（KV）缓存，使用其 FlashAttention 后端运行注意力，并进行贪心采样。注意力采用数据并行，专家采用专家并行：每个 GPU 运行一个执行引擎，禁用张量并行，并持有 1/专家并行 的请求以及每层 128/专家并行 个专家。分发和合并使用 DeepEP 的高吞吐内核，并配备一个大小足以容纳整个本地预填充的缓冲区。禁用分块预填充和前缀缓存，因此每个 GPU 都在单步内预填充其本地批次；显式设置 KV 缓存大小，以确保不会抢占任何请求，并忽略序列结束词元，使每个请求都恰好生成 32 个词元。

编译与 CUDA 图。 我们禁用了 vLLM 的图编译器，因为它会在注意力处拆分前向传播，且无法跟踪跨层流切换和 DeepEP 调用。相反，逐元素运算和归一化区域（路由、融合残差加法和 RMSNorm、查询/键

使用 RoPE 的归一化）会分别通过 `torch.compile`(Ansel 等人，2024年）进行编译（完整图、静态形状），并将权重作为输入，在各层之间复用同一个已编译函数。专家模块在编译之外运行，在 Triton SwiGLU 内核的前后各执行一次分组 GEMM。预填充不使用 CUDA 图，而是使用 CPU 同步来精确确定 DeepEP 接收缓冲区的大小。解码则通过为完整本地批次分配缓冲区，并将包括两个流和 DeepEP 内核在内的整个前向传播捕获到每步一个 CUDA 图中，避免这种同步。每种调度和 SM 设置都会重新捕获图，且记录过程不计入计时。

权重与路由。 BF16 检查点在每个 GPU 上加载一次，其中包括注意力、归一化、路由器、嵌入和本地专家权重，并将门控投影与上投影拼接。路由遵循 Qwen3：对 128 个专家执行 FP32 softmax，选择 top-8 个专家并重新归一化。由于 DeepEP 的合并操作会将专家输出相加而不进行加权，因此每个接收行都会在 SwiGLU 内核中、下投影之前乘以其 FP32 路由权重。为隔离执行差异，所有调度方案都会重放同一提示词上一次未计时顺序运行所记录的专家索引和权重。顺序重放会复现其原始生成的词元。

融合。 每种调度方案都会利用其依赖关系所允许的融合。顺序方案的合并尾处理会执行专家归约、残差相加和下一次注意力归一化，而其注意力残差相加、MoE 输入归一化和路由则组成一个编译区域。并行方案的合并尾处理会将两个分支都加到残差上，并在一次双输出处理中计算下一层的两种归一化。这些归一化共享残差输入，与并行 Transformer 块中的做法相同（Black 等人，2022）。梯形方案和 BRAID 共享一个合并尾处理，该尾处理执行专家归约和残差相加，生成更新后的残差及其归一化的专家输入。BRAID 还融合了注意力归一化和早期路由器归一化，仅计算一次 RMS 统计量，并应用独立的学习缩放因子。在梯形方案和 BRAID 中，注意力残差相加先于等待分发，这与算法 1 相同。

工作负载。 提示是从 FineWeb v1.4.0 子集中抽取的固定长度词元窗口，使用原生 Qwen3-30B-A3B-Base 分词器。我们改变批量大小 B 在固定 $L = 1024$ 和提示长度 L 为每个 GPU 固定 16 个提示（表 6）。对于专家并行=8，全局批次大小相较于专家并行=4 翻倍，以保持每个 GPU 的提示数相同。

专家放置。 对于所有调度方案，我们都使用 DeepSeek 的 EPLB(DeepSeek-AI 等人，2025年) 计算相同的静态专家放置，且不进行专家副本

表 6。推理工作负载：专家并行=4 和专家并行=8的提示长度 L 和批量 B。

L	B (专家并行=4)	B (专家并行=8)	每个GPU的提示数
1,024	16	32	4
256	64	128	16
1,024	64	128	16
4,096	64	128	16
1,024	256	512	64

表7。 基准提示中最繁忙GPU的负载相对于平均值的比例，按层平均（预填充）。

L	1,024	256 1	,024 4	,096 1	,024
每个GPU的提示数	4	16	16	16	64
专家并行=4，连续	1.208	1.214	1.202	1.197	1.200
EP=4, EPLB	1.0671.042	1.028	1.016	1.013	
EP=8, EPLB	1.086	1.059	1.032	1.025	1.024

。专家负载通过 420 万个留出的 FineWeb 词元进行测量；专家按负载从高到低分配给 GPU，每次选择有空余槽位且负载最低的 GPU。推理期间，放置方案保持固定，并在加载时对专家权重和路由器行进行置换。表 7 报告了由此产生的负载均衡情况。

C.2 通信内核调优

SM 分配和资源争用。 我们用一个简单模型说明资源争用对通信–计算重叠的影响。假设两个内核单独运行时分别耗时 t_1 和 t_2，且其运行时间与分配到的 SM 份额成反比。当以 λ 和 $1-\lambda$ 的比例并发运行时，有

$$\min_{0<\lambda<1} \max\left(\frac{t_1}{\lambda}, \frac{t_2}{1-\lambda}\right) = t_1 + t_2.$$

在这些假设下，即使进行最优资源分配，相较于顺序执行也无法获得加速比。然而，通信和计算主要依赖不同的资源：互连带宽与计算吞吐量或 HBM 带宽。因此，尽管存在一定的 SM 争用，二者仍可实现有益的重叠。为通信分配更多 SM 可能会降低其延迟，但同时减慢并发计算。因而，我们调优 DeepEP 的传输和尾处理阶段的 SM 分配，以最小化端到端延迟，而不只是通信延迟。

SM 调优。 对于每个调度，我们会调优五个参数：DeepEP 的分发和合并传输及其尾处理阶段的 SM 分配，以及计算 SM 余量 m，该余量限制了注意力内核可用的 SM。搜索范围涵盖 DeepEP 的默认设置、从 2 到 96 个 SM 的传输分配、16、32、64 个或全部 SM 的尾处理阶段，以及从 0 到 64 个 SM 的余量。候选配置将在以下数据上进行评估：

表 8。针对不同调度、拓扑和阶段所选用的 DeepEP 设置：分发/合并传输 SM、分发/合并尾声 SM，以及计算 SM 余量 m。“auto”表示 DeepEP 的自动传输预算（EP=4 时为 20 个 SM，EP=8 时为 12 个）；“all”表示全部 132 个 SM。

		EP=4		EP=8	
调度		传输	尾处理 / m	传输	Epil. / m
顺序	预填充	64/64	全部/全部 / 0	24/64	全部/全部 / 0
	解码	8/32	64/64 / 0	2/16	64/64 / 0
并行	预填充	12/96	32/全部 / 0	24/64	全部/全部 / 48
	解码	8/20	64/64 / 0	2/16	全部/全部 / 0
梯形	预填充	64/24	全部/全部 / 0	24/24	全部/全部 / 32
	解码	自动/4	64/64 / 0	4/32	全部/全部 / 64
BRAID	预填充	12/64	全部/全部 / 16	24/64	全部/全部 / 64
	解码	4/32	64/64 / 0	4/32	all/all / 64

与基准数据不重叠的留出提示，涵盖每种拓扑的全部五个工作负载，并使用两次预热和七个随机轮次。我们针对每个调度、拓扑和阶段（预填充或解码）选择一个配置，使其相对于每个工作负载上最快配置的最差相对减速最小。表 8 报告了所选设置。

C.3 计时与通信重叠

计时。每个作业针对每种调度方案运行两次预热迭代，随后进行 30 轮测试；每轮中，每种调度方案按随机顺序运行一次迭代。为将调度影响与专家分配和负载均衡差异隔离开来，所有调度方案都重放由未计时的顺序基准运行记录的相同专家索引和路由权重。一次迭代会同步所有 GPU，在一步中为每个本地提示预填充一个请求，并生成 32 个词元。预填充时间为第一步的耗时，解码时间则为剩余 31 步耗时的平均值；二者均在最慢的 GPU 上测量。每个作业报告其各轮结果的中位数；对于每个工作负载，我们报告五个作业中位数的平均值及标准误，加速比根据未四舍五入的平均值计算。重叠测量值是三条 Nsight Systems 轨迹的中位数，每条轨迹均来自一个独立的性能分析作业，且该作业不计入延迟测量。表 9给出了顺序阶段计时。

理想化重叠上限。令 T_c 和 T_m 分别表示顺序基线的计算时间和通信时间，因此 $T = T_c + T_m$。我们假设所有通信都可以重叠，但只有部分计算 $T_c^{\text{ov}} \le T_c$ 提供独立的重叠窗口。在完美重叠且不存在资源争用的情况下，延迟降低的最大值为 $\min(T_m,\ T_c^{\text{ov}})$，从而得到理想加速比上界

$$S \le \frac{T}{T - \min(T_m, T_c^{\text{ov}})}. \tag{3}$$

对于固定的 T，当 T_m 和 T_c^{ov} 达到平衡，且它们的最小值占 T 的比例更大时，该上界更大。

对于表9，定义 $\rho = T_m/T_{\text{ovc}}$ 以及 $S_{\text{ideal}} = T/[T - \min(T_m,\ T_c^{\text{ov}})]$。比值 ρ 用于判断通信时间还是独立计算窗口更长； S_{ideal} 是在固定顺序成本分解下的理想化重叠上限，而不是实测的 BRAID 加速比。

例如，在 EP=4下，预填充阶段使用 $B = 64$ 和 $L = 1024$，$T_m = 88.2$ 毫秒和 $T_c^{\text{ov}} = 82.1$ 毫秒分别得到 $\rho = 1.07$ 和$S_{\text{ideal}} = 1.39$。在 EP=8 下，每个 GPU 的提示数相同（$B = 128$），通信时间增加到 273.6 毫秒，而计算窗口为 82.8 毫秒，得到 $\rho = 3.30$，并将S_{ideal} 降至 1.21。实测加速比仍低于这些上限，部分原因是计算时间增加：在预填充阶段，BRAID 的计算内核总持续时间在 EP=4 下比 Sequential 长 17–25%，在 EP=8下长 31–45%；此时 BRAID 使用 64-SM 计算余量。这些差异与资源争用和可用计算 SM 减少相一致，不过内核融合的差异也可能有所影响。因此，大规模的通信重叠并不一定会转化为端到端延迟的同比改善。

C.4 工作负载扫描

表 10和 11给出了专家并行的完整延迟、加速比和重叠测量数据=4和专家并行=8。对应的加速比图见图 5和6。

表*9.* 秩 0 上的顺序阶段计时，取三条 Nsight Systems 跟踪记录的中位数，单位为毫秒。 T表示 GPU 总耗时， T_m 表示 DeepEP 分发与合并内核及其尾处理阶段的时长总和，而 T_E 表示专家计算时间。我们根据注意力和其他非专家计算内核的时长总和，估算独立计算窗口 T_c^{ov}。各阶段的最后两列显示 $\rho = T_m/T_{\text{重叠c}}$ 以及根据公式（3）得出的理想化上限。跟踪记录的总耗时与基准延迟相差约 5% 以内；基准延迟是在五个独立的随机计时任务中，于最慢的 GPU 上测得的。

	预填充						解码步骤					
工作负载	T	T_m	$T_{\text{重叠c}}$	T_E	ρ	S_{ideal}	T	T_m	T_{ovc}	T_E	ρ	S_{ideal}
专家并行=4 16×1024	91.0	28.7	20.8	29.6	1.38×	1.30×	9.58	2.16	2.98	3.78	0.72×	1.29×
专家并行=4 64×256	87.4	26.9	19.2	29.2	1.41×	1.28×	10.74	2.12	3.11	4.86	0.68×	1.25×
专家并行=4 64×1024	293.3	88.2	82.1	111.7	1.07×	1.39×	11.22	2.14	3.52	4.94	0.61×	1.24×
专家并行=4 64×4096	1274.7	333.9	465.7	464.4	0.72×	1.35×	12.87	2.20	5.10	4.96	0.43×	1.21×
专家并行=4 256×1024	1149.3	347.5	334.9	455.3	1.04×	1.41×	14.05	2.34	5.47	5.63	0.43×	1.20×
专家并行=8 32×1024	145.1	82.8	21.2	28.8	3.91×	1.17×	11.74	5.30	3.08	2.68	1.72×	1.36×
专家并行=8 128×256	145.0	83.1	19.5	28.4	4.25×	1.16×	12.98	5.99	3.20	3.08	1.87×	1.33×
专家并行=8 128×1024	481.6	273.6	82.8	114.6	3.30×	1.21×	13.54	6.00	3.63	3.14	1.65×	1.37×
专家并行=8 128×4096	1927.6	978.4	463.0	468.8	2.11×	1.32×	15.16	6.00	5.22	3.18	1.15×	1.52×
专家并行=8 512×1024	1844.5	1017.0	338.8	468.2	3.00×	1.23×	19.52	8.86	5.61	4.38	1.58×	1.40×

表 *10.* 采用 EP=4 的所有工作负载：五个作业各运行 30 轮所得中位数的平均值 ± 及其标准误、相对于顺序执行的加速比，以及来自 Nsight Systems 跟踪的合并/分发重叠（在同一 GPU 上与计算并发的通信内核时间占比，跨 GPU 统计）。

		预填充				解码			
工作负载	调度	延迟 (ms)	加速比	合并重叠	分发重叠	延迟（毫秒/步）	加速比	合并重叠	分发重叠
16×1024	顺序	87.5 ± 0.2	1.000×	0%	0%	9.36 ± 0.01	1.000×	0%	0%
	并行	79.9 ± 0.3	1.094×	0%	74.7%	8.60 ± 0.01	1.088×	0%	86.5%
	梯形	83.2 ± 0.1	1.052×	81.2%	8.2%	8.41 ± 0.01	1.113×	89.4%	10.0%
	BRAID	$\mathbf{76.7} \pm 0.3$	**1.141**×	86.2%	48.1%	$\mathbf{7.65} \pm 0.01$	**1.223**×	87.5%	77.7%
64×256	顺序	84.5 ± 0.1	1.000×	0%	0%	10.64 ± 0.01	1.000×	0%	0%
	并行	78.1 ± 0.1	1.081×	0%	66.7%	9.86 ± 0.01	1.079×	0%	85.8%
	梯形	80.8 ± 0.1	1.045×	81.9%	8.8%	9.54 ± 0.01	1.115×	91.2%	11.5%
	BRAID	$\mathbf{75.3} \pm 0.2$	**1.122**×	89.8%	40.7%	$\mathbf{8.84} \pm 0.01$	**1.203**×	88.4%	81.6%
64×1024	顺序	291.9 ± 0.2	1.000×	0%	0%	11.13 ± 0.01	1.000×	0%	0%
	并行	262.5 ± 0.5	1.112×	0%	93.2%	10.42 ± 0.01	1.068×	0%	86.9%
	梯形	267.8 ± 0.2	1.090×	86.1%	11.6%	10.09 ± 0.01	1.103×	92.8%	11.1%
	BRAID	$\mathbf{249.2} \pm 0.4$	**1.171**×	96.3%	92.8%	$\mathbf{9.41} \pm 0.01$	**1.182**×	89.2%	82.7%
64×4096	顺序	1279 ± 2	1.000×	0%	0%	12.92 ± 0.01	1.000×	0%	0%
	并行	1158 ± 1	1.105×	0%	99.7%	12.18 ± 0.01	1.061×	0%	85.5%
	梯形	1186 ± 1	1.078×	86.7%	13.2%	11.83 ± 0.01	1.092×	95.4%	11.1%
	BRAID	$\mathbf{1101} \pm 2$	**1.161**×	98.0%	99.6%	$\mathbf{11.14} \pm 0.01$	**1.160**×	90.4%	88.4%
256×1024	顺序	1154 ± 2	1.000×	0%	0%	14.01 ± 0.01	1.000×	0%	0%
	并行	1052 ± 3	1.096×	0%	93.0%	13.23 ± 0.01	1.059×	0%	89.1%
	梯形	1061 ± 2	1.088×	85.6%	13.1%	12.83 ± 0.01	1.092×	95.9%	10.8%
	BRAID	$\mathbf{986} \pm 2$	**1.170**×	98.0%	94.5%	$\mathbf{12.33} \pm 0.01$	**1.136**×	89.6%	89.6%

表*11*。两个节点上所有采用专家并行=8 的工作负载：30 轮的五个作业中位数的平均值 ± 及其标准误、相较于顺序执行的加速比，以及根据 Nsight Systems 跟踪数据得出的合并/分发重叠（所有 GPU 上与计算同时进行的通信内核时间占比）。

		预填充				解码			
工作负载	调度	延迟 (ms)	加速比	合并重叠	分发重叠	延迟（毫秒/步）	加速比	合并重叠	分发重叠
32×1024	顺序	141.1 ± 0.1	1.000×	0%	0%	11.69 ± 0.02	1.000×	0%	0%
	并行	123.4 ± 0.1	1.143×	0%	48.5%	9.67 ± 0.02	1.208×	0%	83.5%
	梯形	126.7 ± 0.1	1.113×	75.2%	2.5%	10.45 ± 0.02	1.118×	59.9%	4.7%
	BRAID	$\mathbf{122.7} \pm 0.1$	**1.150**×	88.6%	19.1%	$\mathbf{9.59} \pm 0.02$	**1.219**×	82.6%	4.4%
128×256	顺序	137.9 ± 0.4	1.000×	0%	0%	12.83 ± 0.02	1.000×	0%	0%
	并行	122.2 ± 0.4	1.128×	0%	43.1%	10.68 ± 0.02	1.201×	0%	76.6%
	梯形	124.6 ± 0.4	1.107×	70.3%	2.6%	11.20 ± 0.03	1.146×	55.7%	4.4%
	BRAID	$\mathbf{121.3} \pm 0.4$	**1.136**×	87.9%	15.9%	$\mathbf{10.24} \pm 0.04$	**1.253**×	79.3%	4.5%
128×1024	顺序	479 ± 3	1.000×	0%	0%	13.45 ± 0.03	1.000×	0%	0%
	并行	408 ± 3	1.175×	0%	52.9%	11.01 ± 0.03	1.222×	0%	90.1%
	梯形	423 ± 3	1.133×	90.6%	2.6%	11.55 ± 0.04	1.165×	65.2%	4.3%
	BRAID	$\mathbf{407} \pm 2$	**1.178**×	95.3%	31.3%	$\mathbf{10.53} \pm 0.03$	**1.278**×	88.8%	5.8%
128×4096	顺序	1940 ± 10	1.000×	0%	0%	15.05 ± 0.06	1.000×	0%	0%
	并行	1530 ± 8	1.271×	0%	81.9%	12.52 ± 0.03	1.202×	0%	95.1%
	梯形	1740 ± 10	1.120×	98.1%	2.8%	12.27 ± 0.04	1.226×	93.5%	4.0%
	BRAID	$\mathbf{1510} \pm 10$	**1.287**×	97.1%	69.1%	$\mathbf{10.98} \pm 0.08$	**1.370**×	91.9%	46.8%
512×1024	顺序	1833 ± 9	1.000×	0%	0%	19.58 ± 0.09	1.000×	0%	0%
	并行	1541 ± 6	1.190×	0%	54.0%	16.20 ± 0.08	1.209×	0%	86.8%
	梯形	1615 ± 7	1.135×	88.4%	2.7%	15.75 ± 0.09	1.244×	84.6%	3.2%
	BRAID	$\mathbf{1528} \pm 7$	**1.200**×	97.2%	36.1%	$\mathbf{14.35} \pm 0.09$	**1.365**×	93.5%	22.7%

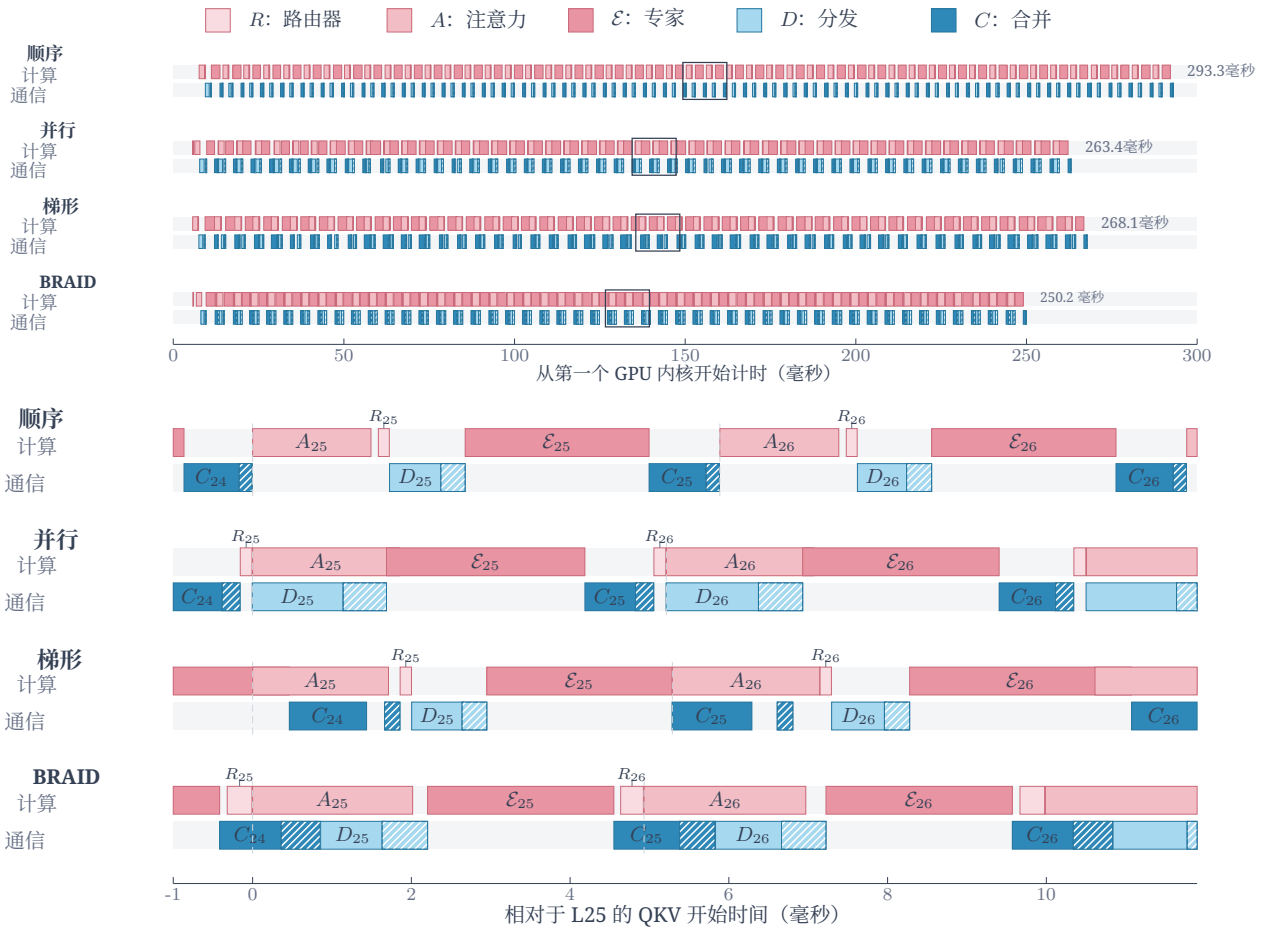

图7。 Qwen3-30B-A3B-Base 在 BF16 精度下，使用 vLLM 和 DeepEP 在四个 H100 GPU 上运行时的 GPU 执行时间线（专家并行 $= 4$），全局批次为 64 个长度为 1,024 的提示。上方时间线展示完整的 48 层预填充过程，下方时间线则放大展示第 25 层和第 26 层，以及此前第 24 层的合并传输。计算模块的跨度从每个逻辑阶段的第一个 GPU 内核延伸至最后一个 GPU 内核；每次 DeepEP 分发或合并传输及其 GPU 收尾阶段均单独显示。收尾阶段保留其对应通信阶段的颜色，并以白色斜线标记。未标注的小时间间隔包含残差或归一化工作，或执行间隙。这些时间线来自秩 0 上单次经过 Nsight 插桩的迭代；表 4 报告了随机化延迟测量结果。