深入解析 DeepSeek-V3：AI 架构的扩展挑战与硬件反思

Chenggang Zhao
DeepSeek-AI
Beijing, China
chenggangz@deepseek.com

Chengqi Deng
DeepSeek-AI
Beijing, China
cq.deng@deepseek.com

Chong Ruan
DeepSeek-AI
Beijing, China
chong.ruan@deepseek.com

Damai Dai
DeepSeek-AI
Beijing, China
damai.dai@deepseek.com

Huazuo Gao
DeepSeek-AI
Beijing, China
gaohuazuo@deepseek.com

Jiashi Li
DeepSeek-AI
Beijing, China
js.li@deepseek.com

Liyue Zhang*
DeepSeek-AI
Beijing, China
ly.zhang@deepseek.com

Panpan Huang
DeepSeek-AI
Beijing, China
pp.huang@deepseek.com

Shangyan Zhou
DeepSeek-AI
Beijing, China
sy.zhou@deepseek.com

Shirong Ma
DeepSeek-AI
Beijing, China
mashirong.2000@deepseek.com

Wenfeng Liang
DeepSeek-AI
Beijing, China
wenfeng.liang@deepseek.com

Ying He
DeepSeek-AI
Beijing, China
ying.he@deepseek.com

Yuqing Wang*
DeepSeek-AI
Beijing, China
wangyq@deepseek.com

Yuxuan Liu
DeepSeek-AI
Beijing, China
liuyuxuan@deepseek.com

Y.X. Wei
DeepSeek-AI
Beijing, China
weiyx@deepseek.com

Abstract

大型语言模型（LLM）的快速扩展揭示了当前硬件架构的关键局限性，包括内存容量、计算效率和互连带宽方面的限制。在 2,048 块 NVIDIA H800 GPU 上训练

*Yuqing Wang and Liyue Zhang are the corresponding authors of this paper. Authors are listed in alphabetical order of their first names.

ISCA '25, June 21–25, 2025, Tokyo, Japan

ACM ISBN 979-8-4007-1261-6/2025/06
https://doi.org/10.1145/3695053.3731412

的 DeepSeek-V3 展示了如何通过硬件感知的模型协同设计有效应对这些挑战，从而实现大规模、高性价比的训练与推理。本文深入分析了 DeepSeek-V3/R1 的模型架构及其 AI 基础设施，重点介绍了多项关键创新：用于提升内存效率的多头潜在注意力（MLA）、用于优化计算与通信权衡的混合专家（MoE）架构、用于充分释放硬件潜力的 FP8 混合精度训练，以及用于最小化集群级网络开销的多平面网络拓扑。基于 DeepSeek-V3 开发过程中遇到的硬件瓶颈，我们与学术界和工业界的同行就未来硬件的潜在发展方向进行了更广泛的探讨，包括精确的低精度计算单元、纵向扩展（scale-up）与横向扩展（scale-out）的融合，以及低延迟通信架构的创新。这些见解凸显了硬件与模型协同设计在应对日益增长的 AI 工作负载需求中的关键作用，为下一代 AI 系统的创新提供了切实可行的蓝图。

CCS Concepts

• **Computer systems organization → Architectures**.

Keywords

大型语言模型，混合专家，深度学习，FP8 混合精度训练，多平面网络，协同设计

ACM Reference Format:

Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Huazuo Gao, Jiashi Li, Liyue Zhang, Panpan Huang, Shangyan Zhou, Shirong Ma, Wenfeng Liang, Ying He, Yuqing Wang, Yuxuan Liu, Y.X. Wei . 2025. 深入解析 DeepSeek-V3：AI 架构的扩展挑战与硬件反思. In *Proceedings of the 52nd Annual International Symposium on Computer Architecture (ISCA '25), June 21–25, 2025, Tokyo, Japan.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3695053.3731412

1 引言

1.1 背景

近年来，在模型设计、算力和数据可用性的迭代进步推动下，大型语言模型（LLM）经历了快速演进。2024 年，GPT4o [?]、LLaMa-3 [?]、Claude 3.5 Sonnet [?]、Grok-2 [?]、Qwen2.5 [?]、Gemini-2 [?] 以及我们的 DeepSeek-V3 [?] 等突破性模型展现了显著进展，进一步缩小了通往通用人工智能（AGI）的差距。正如缩放定律（Scaling Laws）[?] 所示，增加模型规模、训练数据和计算资源能够显著提升模型性能，凸显了扩展在推动 AI 能力发展中的核心作用。总体而言，这些进展开启了一个新时代，在这个时代中，扩展模型规模和算力被视为解锁更高水平智能的关键。

近期的进展表明，诸如 OpenAI 的 o1/o3 系列模型 [? ?]、DeepSeek-R1 [?]、Claude-3.7 Sonnet [?]、Gemini 2.5 Pro [?]、Seed1.5-Thinking [?] 和 Qwen3 [?] 等推理模型，不仅展现了大规模架构带来的优势，也凸显了提升推理效率的必要性，尤其是在处理更长上下文和实现更深推理深度方面。这些进展进一步强调了更快、更高效推理的必要性，从而对计算资源提出了日益增长的要求。

为应对这些挑战，阿里巴巴、字节跳动、Google、xAI 和 Meta 等行业领军企业已部署了庞大的训练集群 [? ? ? ? ? ?]，配备数万甚至数十万块 GPU 或 TPU。尽管此类庞大基础设施推动了最先进模型的开发，但其高昂的成本为小型研究团队和机构设置了显著障碍。尽管面临这些障碍，DeepSeek [? ? ? ? ?] 和 Mistral [? ?] 等开源初创公司仍在努力开发最先进模型。其中，DeepSeek 尤其证明了有效的软硬件协同设计能够实现大模型的高性价比训练，为小型团队创造了公平的竞争环境。秉承这一传统，DeepSeek-V3 [?] 在高性价比训练方面树立了新的里程碑。仅借助 2,048 块 NVIDIA H800 GPU，DeepSeek-V3 便实现了最先进的性能。这一成就与我们致力于通过实用且可扩展的解决方案推动 AI 发展的承诺相一致，此前在 Fire-Flyer AI-HPC [?] 的高性价比架构中已得到验证。从 DeepSeek-V3 中得出的实践经验与见解展示了如何充分挖掘现有硬件资源的潜力，为更广泛的 AI 和 HPC 社区提供了宝贵的经验。

1.2 研究目标

本文并非旨在重复阐述 DeepSeek-V3 的详细架构与算法细节，这些内容在其技术报告 [?] 中已有详尽记载。相反，本文采用硬件架构与模型设计的双重视角，探

讨二者在实现高性价比大规模训练与推理过程中的复杂相互作用。通过剖析这种协同效应，我们旨在为高效扩展大语言模型（LLM）提供切实可行的见解，同时不牺牲性能或可及性。

具体而言，本文聚焦于以下方面：

- **硬件驱动的模型设计：** 分析硬件特性（如 FP8 低精度计算以及 Scale-up/Scale-out 网络属性）如何指导 DeepSeek-V3 的架构选择。
- **硬件与模型之间的相互依赖：** 探讨硬件能力如何塑造模型创新，以及 LLM 不断演进的需求如何推动下一代硬件的发展。
- **硬件开发的未来方向：** 从 DeepSeek-V3 中提炼切实可行的见解，以指导未来硬件与模型架构的协同设计，为可扩展、高性价比的 AI 系统铺平道路。

1.3 本文结构

本文其余部分组织如下。第 2 节探讨了支撑 DeepSeek-V3 模型架构的设计原则，重点介绍了多头潜在注意力（Multi-head Latent Attention）、混合专家（Mixture-of-Experts）优化以及多 Token 预测模块等关键创新。第 3 节阐述了我们的模型架构如何实现低精度计算与通信。第 4 节涵盖 Scale-up 互连优化，讨论 Scale-up/Scale-out 的融合，并探讨硬件特性如何影响并行策略与专家选择策略。第 5 节聚焦于 Scale-out 网络优化，包括多平面网络协同设计与低延迟互连技术。除第 3~5 节中提及的当前局限性与未来建议外，第 6 节进一步深入剖析了来自 DeepSeek-V3 的关键见解，并明确了未来硬件与模型协同设计的发展方向。

2 DeepSeek 模型的设计原则

DeepSeek-V3 的开发体现了扩展 LLM 的硬件感知方法，其中每一项设计决策都经过精心考量，以契合硬件约束，从而优化性能与成本效益。

如图 1 所示，DeepSeek-V3 采用了在 DeepSeek-V2 [?] 中已被证明有效的 **DeepSeekMoE** [?] 与 **多头潜在注意力（Multi-head Latent Attention, MLA）** [?] 架构。DeepSeekMoE 释放了 MoE 架构的潜力，而 MLA 则通过压缩键值（Key-Value, KV）缓存大幅降低了内存消耗。此外，**DeepSeek-V3** 引入了 **FP8 混合精度训练**，在保障模型质量的前提下显著降低了计算成本，使大规模训练更具可行性。为提升推理速度，DeepSeek-V3 基于其 **多 Token 预测模块**集成了投机解码（speculative decoding），显著提高了生成速度。在模型架构之外，我们还通过部署 **多平面（Multi-Plane）** 两层胖树（Fat-Tree）网络来替代传统的三层胖树拓扑，探索了高性价比的 AI 基础设施，从而降低了集群网络成本。

这些创新旨在应对扩展 LLM 过程中的三大核心挑战——**内存效率**、**成本效益**与 **推理速度**，以下小节将对这些挑战进行详细探讨。

2.1 内存效率

LLM 通常需要大量的内存资源，其内存需求每年增长超过 1000%。相比之下，高速内存（如 HBM）容量的增长率要慢得多，通常每年不足 50% [?]。尽管多节点并行是应对内存限制的有效方案，但从源头优化内存使用仍是至关重要且高效的策略。

2.1.1 **低精度模型.** 与采用 BF16 权重的模型相比，FP8 将内存消耗减半，有效缓解了 AI 内存墙挑战。关于低精度技术的详细讨论见第 3 节“低精度驱动设计”。

2.1.2 **利用 *MLA* 减少 *KV* 缓存.** 在 LLM 推理中，用户请求通常涉及多轮对话。为高效处理这些请求，先前请求的上下文会被缓存，这通常被称为 *KV 缓存*。KV 缓存通过缓存已处理 Token 的 **Key** 和 **Value** 向量来解决这一挑战，从而免除了对后续 Token 重复计算的需求。在每次推理步骤中，模型仅计算当前 Token 的 Key 和 Value 向量，并将其与历史缓存的 Key-Value 对结合以执行注意力计算。这种增量计算将生成每个 Token 的复杂度降低至 $O(N)$，在处理长序列或多轮输入时十分高效。然而，这也引入了内存瓶颈，因为计算从 GEMM 转变为 GEMV，后者的计算访存比低得多。在现代硬件提供数百 TFLOPS 算力的情况下，GEMV 很快会受到内存带宽的限制，使内存访问成为主要瓶颈。

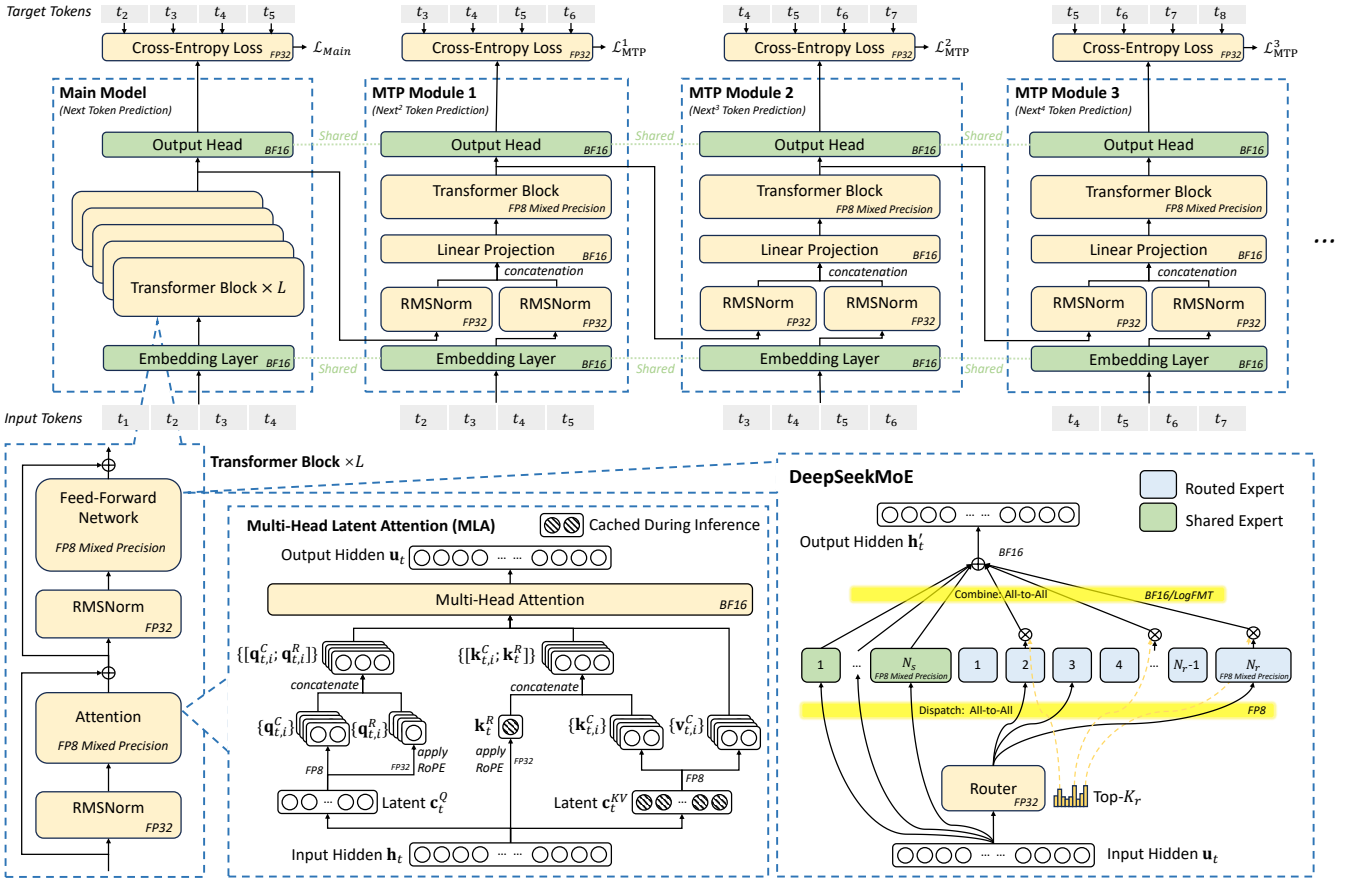

Figure 1: DeepSeek-V3 的基础架构。在 DeepSeek-V2 的 MLA 和 DeepSeekMoE 基础上，引入了多 Token 预测模块与 FP8 混合精度训练，以提升推理与训练效率。图中展示了架构不同部分计算所使用的精度。所有组件的输入与输出均采用 BF16 格式。

为应对这一瓶颈，我们采用了 **多头潜在注意力（MLA）**[?]，它通过一个与模型联合训练的投影矩阵，将所有注意力头的 KV 表示压缩为一个更小的潜在向量。在推理过程中，仅需缓存该潜在向量，与存储所有注意力头的 KV 缓存相比，显著降低了内存消耗。

除 MLA 外，学界还提出了多种其他方法来减小 KV 缓存的规模。这些方法极具价值，为内存高效注意力机制的进步提供了重要启发：

- **共享 KV（分组查询注意力 GQA；多查询注意力 MQA）：** 不再为每个注意力头维护独立的 KV 对，而是让多个头共享一组 KV 对，从而大幅压缩 KV 存储。代表性方法包括 GQA [?] 和 MQA [?]。
- **窗口化 KV：** 对于长序列，缓存中仅保留一个滑动窗口内的 KV 对，丢弃窗口外的结果。尽管这减少了存储需求，但会损害长上下文推理能力。代表性方法包括 Longformer [?] 及相关架构。
- **量化压缩：** KV 对采用低位宽表示进行存储 [???]，进一步降低内存占用。量化技术在几乎不影响模型性能的前提下实现了显著的压缩效果。

Table 1: KV 缓存大小对比（BF16 精度）：与采用 GQA 的其他模型相比，DeepSeek-V3（MLA）大幅降低了 KV 缓存大小。

模型	每 Token KV 缓存大小	倍数
DeepSeek-V3 (MLA)	70.272 KB	1x
Qwen-2.5 72B (GQA)	327.680 KB	4.66x
LLaMA-3.1 405B (GQA)	516.096 KB	7.28x

表 1 对比了 DeepSeek-V3、Qwen-2.5 72B [?] 与 LLaMA-3.1 405B [?] 每 Token 的 KV 缓存内存占用。通过采用 MLA，DeepSeek-V3 实现了 KV 缓存大小的显著降低，每 Token 仅需 70 KB，远低于 LLaMA-3.1

405B 的 516 KB 和 Qwen-2.5 72B 的 327 KB。这一降低凸显了相较于基于 GQA 的方法，MLA 在压缩 KV 表示方面的高效性。实现如此显著的内存消耗降低，使 DeepSeek-V3 特别适用于涉及长上下文处理与资源受限环境的场景，从而支持更具可扩展性且成本效益更高的推理。

2.1.3 **资源高效技术的未来方向与展望**. 尽管减小 KV 缓存规模是提升内存效率的有前景方法，但基于 Transformer 的自回归解码固有的二次方复杂度仍是一项严峻挑战，尤其是在极长上下文场景下。近期的研究工作，如 Mamba-2 [**?**] 和 Lightning Attention[**?**]，探索了线性时间复杂度的替代方案，为平衡计算成本与模型性能提供了新的可能性。此外，稀疏注意力 [**?**] 等方法试图压缩并稀疏激活注意力键与值，代表了克服注意力相关计算挑战的另一种尝试。我们期待与更广泛的社区携手合作，在该领域取得突破性进展。

2.2 MoE 模型的成本效益

针对稀疏计算，我们开发了 DeepSeekMoE，这是一种先进的 **混合专家（Mixture of Experts, MoE）** 架构，如图 1 右下角所示。MoE 模型的优势主要体现在两个方面。

2.2.1 **降低训练的计算需求**. MoE 架构的主要优势在于其能够显著降低训练成本。通过仅选择性激活部分专家参数，MoE 模型能够在保持计算需求相对可控的同时，实现总参数规模的急剧扩展。例如，**DeepSeek-V2** 拥有 236B 参数，但每个 Token 仅激活 21B 参数。同样，**DeepSeek-V3** 的参数规模扩展至 671B——几乎是 V2 的三倍——而每个 Token 的激活参数仅保持在 37B。相比之下，Qwen2.5-72B 和 LLaMa3.1-405B 等稠密模型在训练过程中需要激活所有参数。

如表 2 所示，DeepSeek-V3 的总计算成本约为每个 Token 250 GFLOPS，而 72B 稠密模型需要 394 GFLOPS，405B 稠密模型则需要 2448 GFLOPS。这表明，MoE 模型在消耗少一个数量级计算资源的情况下，能够实现与稠密模型相当甚至更优的性能。

Table 2: MoE 与稠密模型训练计算成本对比：计算成本按每个 Token 衡量，假设序列长度为 4096。

模型	规模	训练成本
DeepSeek-V2 MoE	236B	155 GFLOPS/Token
DeepSeek-V3 MoE	671B	250 GFLOPS/Token
Qwen-72B Dense	72B	394 GFLOPS/Token
LLaMa-405B Dense	405B	2448 GFLOPS/Token

2.2.2 **个人使用与本地部署的优势**. 在个性化大语言模型智能体 [**?**] 日益普及的未来，MoE 模型在单次请求场景下展现出独特优势。由于每次请求仅激活部分参数，内存和计算需求大幅降低。例如，**DeepSeek-V2**（236B 参数）在推理过程中仅激活 21B 参数。这使得搭载 AI SoC 芯片的 PC [**? ? ?**] 能够实现近 20 tokens/秒（TPS）的生成速度，甚至可达两倍于此，完全满足个人使用需求。相比之下，能力相近的稠密模型（如 70B 参数）在同类硬件上通常仅能达到个位数 TPS。

值得注意的是，日益流行的 KTransformers [**?**] 推理引擎使得完整的 DeepSeek-V3 模型能够在配备消费级 GPU 的低成本服务器（成本约 $10,000）上运行，同时仍能实现近 20 TPS。

这种高效性使得 MoE 架构非常适合硬件资源通常受限的本地部署和单用户场景。通过最小化内存和计算开销，MoE 模型无需昂贵的基础设施即可提供高质量的推理性能。

2.3 提升推理速度

2.3.1 **计算与通信重叠：最大化吞吐量**. 推理速度涵盖系统级最大吞吐量和单次请求延迟。为了最大化吞吐量，我们的模型从架构设计之初便采用了双微批次重叠（dual micro-batch overlap）技术 [**? ?**]，有意将通信延迟与计算过程重叠。正如我们的在线推理系统所演示，并得到开源性能分析数据 [**?**] 的支持，我们将 MLA 和 MoE 的计算解耦为两个独立阶段。当一个微批次执行部分 MLA 或 MoE 计算时，另一个微批次同时执行相应的分发（dispatch）通信。反之，在第二个微批次的计算阶段，第一个微批次则进行合并（combine）通信步骤。这种流水线方法实现了全对全（all-to-all）通信与持续计算的无缝重叠，确保 GPU 始终保持满负荷运行。此外，在生产环境中，我们采用了预填充（prefill）

与解码(decode)分离架构[?],将大批次预填充请求和对延迟敏感的解码请求分配给不同规模的专家并行组。该策略最终在真实服务条件下实现了系统吞吐量的最大化。

2.3.2 **推理速度上限**. 本节重点关注大语言模型服务的解码输出速度,通常以**每输出 Token 时间(Time Per Output Token, TPOT)**来衡量。TPOT 是用户体验的关键指标,同时也直接影响 OpenAI o1/o3 和 DeepSeek-R1 等推理模型的响应速度,这些模型依赖推理长度来提升其智能水平。

对于 MoE 模型而言,实现高推理速度依赖于在计算设备间高效部署专家参数。为了达到尽可能快的推理速度,理想情况下每台设备应仅负责单个专家的计算(或在必要时由多台设备协同计算单个专家)。然而,**专家并行(Expert Parallelism, EP)**需要将 Token 路由到相应的设备,这涉及网络上的 `all-to-all` 通信。因此,MoE 推理速度的上限由互联带宽决定。

考虑一个系统,其中每台设备持有一个专家的参数,并每次处理约 32 个 Token。该 Token 数量在计算访存比与通信延迟之间取得了平衡。同时,该 Token 数量确保了在专家并行期间每台设备处理相等的批次大小,从而便于计算通信时间。

对于采用 CX7 400Gbps InfiniBand (IB) 网卡互联的系统,EP 中两次 `all-to-all` 通信所需的时间计算如下:

$$\text{Comm. Time} = (1\text{Byte}+2\text{Bytes})\times 32\times 9\times 7\text{K}/50\text{GB/s} = 120.96\mu s$$

此处,分发(dispatch)使用 FP8(1 字节),而合并(combine)使用 BF16(2 字节),每个 Token 的隐藏层维度约为 7K。系数 9 表示每个 Token 被传输至 8 个路由专家和 1 个共享专家。此计算未包含网络延迟。

如第 2.3.1 节所述,最大化吞吐量必须使用双微批次重叠策略。在该策略下,我们的理论最佳情况分析假设计算开销已最小化,因此性能上限由通信延迟决定。然而,在实际推理负载中,请求上下文通常长得多,且 MLA 计算通常占据主导地位。因此,该分析代表了双微批次重叠下的理想化场景。在此假设下,每层的总时间可表示为:

$$\text{Total Time Per Layer} = 2 \times 120.96\mu s = 241.92\mu s$$

DeepSeek-V3 包含 61 层,总推理时间为:

$$\text{Total Inference Time} = 61 \times 241.92\mu s = 14.76\text{ms}$$

因此,该系统的理论上限约为 **14.76 ms TPOT**,相当于**每秒 67 个 Token**。然而,在实际中,通信开销、延迟、带宽利用率不足以及计算效率低下等因素会降低这一数值。

相比之下,如果使用 GB200 NVL72 这样的高带宽互联架构(72 块 GPU 间单向带宽达 900GB/s),每个 EP 步骤的通信时间将降至:

$$\text{Comm. Time} = (1\text{Byte}+2\text{Bytes})\times 32\times 9\times 7\text{K}/900\text{GB/s} = 6.72\mu s$$

假设计算与通信完美重叠,这将产生 **0.82 ms TPOT** 的理论上限,即约**每秒 1200 个 Token**。然而,该数值纯属理论值,未考虑小批次下 GPU 效率的大幅下降;在实际部署中,实际吞吐量将显著低于此值。尽管如此,该计算生动地展示了高带宽纵向扩展网络在加速大规模模型推理方面的变革潜力。

尽管 MoE 模型展现出良好的可扩展性,但仅靠增加硬件资源来实现高推理速度成本高昂。因此,软件与算法也必须为提升推理效率做出贡献。

2.3.3 **多 *Token* 预测**. 受 ?] 的启发,DeepSeek-V3 引入了**多 Token 预测(Multi-Token Prediction, MTP)**框架,该框架在提升模型性能的同时改善了推理速度。在推理过程中,传统的自回归模型在每个解码步骤仅生成一个 Token,导致顺序瓶颈。MTP 通过使模型能够以较低成本生成额外的候选 Token 并进行并行验证来缓解这一问题,类似于先前基于自草稿(self-drafting)的投机解码方法[??]。该框架在不牺牲准确性的前提下显著加速了推理。

如图 1 上半部分所示，每个 MTP 模块使用单层结构（比完整模型轻量得多）来预测额外 Token，从而实现对多个候选 Token 的并行验证。尽管该方法略微降低了吞吐量，但显著改善了端到端生成延迟。真实世界实践数据表明，MTP 模块在预测第二个后续 Token 时达到了 80% 至 90% 的接受率，与不使用 MTP 模块的场景相比，生成 TPS 提升了 1.8 倍。

此外，通过每步预测多个 Token，MTP 增加了推理批次大小，这对于提升 EP 计算密集度和硬件利用率至关重要。此类算法创新对于 DeepSeek-V3 实现快速且具成本效益的推理至关重要。

2.3.4 **推理模型与测试时扩展的高推理速度**. 大语言模型中的测试时扩展（Test-time scaling），以 OpenAI 的 o1/o3 系列 [? ?] 为代表，通过在推理过程中动态调整计算资源，在数学推理、编程和通用推理方面取得了显著进展。后续模型——包括 DeepSeek-R1 [?]、Claude-3.7 Sonnet [?]、Gemini 2.5 Pro [?]、Seed1.5-Thinking [?] 和 Qwen3 [?]——均采用了类似策略，并在这些任务上取得了显著改进。

对于这些推理模型而言，高 Token 输出速度至关重要。在强化学习（RL）工作流中——例如 PPO [?]、DPO [?] 和 GRPO [?]——快速生成大量样本的需求使得推理吞吐量成为关键瓶颈。同样，过长的推理序列会增加用户等待时间，降低此类模型的实用价值。因此，通过软硬件协同创新优化推理速度，对于提升推理模型的效率不可或缺。然而，如第 2.1.3 节所述，加速推理和加快 RL 训练的有效策略仍是活跃的研究领域。我们鼓励更广泛的社区共同探索并开发解决这些持续挑战的创新方案。

2.4 技术验证方法

每种加速技术都经过严格的实证验证以评估其对精度的影响，包括 MLA、FP8 混合精度计算以及网络协同设计的 MoE 门控路由。鉴于在全规模模型上进行详尽消融实验的成本过高，我们采用了一种分层且资源高效的验证流程。每种技术首先在小规模模型上进行广泛验证，随后进行最小化的大规模调优，最终在单次综合训练运行中集成。例如，在最终集成之前，我们首先在 16B 和 230B 的 DeepSeek-V2 模型上进行了细粒度 FP8 训练的消融研究。在这些受控设置下，与 BF16 相比的相对精度损失保持在 0.25% 以下，这归功于我们采用了高精度累加和细粒度量化策略。

3 低精度驱动设计

3.1 FP8 混合精度训练

诸如 GPTQ [?] 和 AWQ [?] 等量化技术已被广泛用于将位宽降低至 8 位、4 位甚至更低，从而显著减少内存需求。然而，这些技术主要应用于推理阶段以节省内存，而非训练阶段。NVIDIA 的 Transformer Engine 已支持 FP8 混合精度训练一段时间，但在 DeepSeek-V3 之前，尚无开源大模型利用 FP8 进行训练。通过我们的基础设施团队与算法团队的深度合作，以及大量的实验与创新，我们开发了一个兼容 FP8 的 MoE 模型训练框架。图 1 展示了训练流水线中利用 FP8 精度进行前向和反向过程的计算组件。我们采用了细粒度量化，即对激活值采用 tile-wise 1x128 量化，对模型权重采用 block-wise 128x128 量化。我们 FP8 框架的更多技术细节记录在 DeepSeek-V3 技术报告 [?] 中，且我们的细粒度 FP8 GEMM 实现已在 DeepGEMM [?] 中开源。

3.1.1 **局限性**: . 尽管 FP8 在加速训练方面具有巨大潜力，但需解决若干硬件限制才能充分发挥其能力：

- **FP8 累加精度：** FP8 在 Tensor Core 中使用受限的累加精度，影响了大模型训练的稳定性，尤其是在 NVIDIA Hopper GPU 上。在根据最大指数右移对齐 32 个尾数乘积后，Tensor Core 仅保留其最高的 13 位小数位进行加法运算，并截断超出该范围的位。加法结果累加至 FP22 寄存器（1 位符号位、8 位指数位和 13 位尾数位）[?]。“FP22” 一词遵循了所引文献中的命名，且截至 2024 年初，业界已普遍观察到 Hopper GPU 上的 FP8 精度问题。
- **细粒度量化挑战：** 诸如 tile-wise 和 block-wise 量化等细粒度量化技术，在将部分结果从 Tensor Core 传输至 CUDA Core 以进行缩放因子乘法时，引入了巨

大的反量化开销。这导致了频繁的数据移动，降低了计算效率并增加了硬件利用的复杂性。

3.1.2 **建议**：. 为应对现有硬件的局限性，我们对未来的设计提出以下建议：

- **提高累加精度：** 硬件应将累加寄存器精度提升至适当值（例如 FP32），或支持可配置的累加精度，从而为不同模型在训练和推理中的不同需求提供性能与精度之间的权衡。
- **原生支持细粒度量化：** 硬件应原生支持细粒度量化，使 Tensor Core 能够接收缩放因子并实现带组缩放的矩阵乘法。通过这种方式，整个部分和累加与反量化过程可直接在 Tensor Core 内部完成，直至生成最终结果，从而避免频繁的数据移动以降低反量化开销。该方法的一个显著工业实现是 NVIDIA Blackwell 对 **微缩放数据格式（microscaling data format）** [?] 的支持，这充分展示了大规模原生量化的实际收益。

3.2 LogFMT：通信压缩

在当前的 DeepSeek-V3 架构中，我们采用低精度压缩进行网络通信。在 EP 并行期间，token 的调度使用细粒度 FP8 量化，与 BF16 相比通信量减少了 50%。这显著降低了通信时间。尽管出于精度要求，`combine` 阶段仍使用较高精度（例如 BF16），但我们正在积极测试 FP8、自定义精度格式（例如 E5M6）以及 FP8-BF16 混合方案，以进一步降低通信开销。

除了这些传统的浮点格式外，我们还尝试了一种名为 **对数浮点格式（Logarithmic Floating-Point Formats, LogFMT-nBit）** 的新数据类型，其中 n 为总位数，首位 1 位作为符号位 S。通过将激活值从原始的 `Linear` 空间映射到 `Log` 空间，激活值的分布更加均匀。具体而言，给定一个元素 tile $[x_1, \cdots, x_m]$（在我们的实现中为 1x128），我们取其绝对值并计算所有元素的对数，找到最小值 $min = log(abs(x_i))$ 和最大值 $max = log(abs(x_j))$。最小值编码为 $S.00\cdots01$，最大值编码为 $S.11\cdots11$，区间表示为 $Step = \frac{max-min}{2^{n-1}-2}$。零值特别表示为 $S.00\cdots00$。其余值四舍五入到最接近的 $Step$ 的整数倍 K。解码过程很简单，只需结合符号位和 $exp^{min+Step\times(K-1)}$ 即可。

通过局部计算 min 和 $Step$，该数据类型支持不同 block 的动态表示范围，与静态浮点格式相比，能够覆盖更大的范围或提供更高的精度。此外，我们发现为了进行无偏的激活值量化，在原始的 `Linear` 空间而非 `Log` 空间中进行舍入非常重要。我们还约束 min 大于 $max - log(2^{32})$，这意味着最大表示范围类似于 E5（一种具有 5 位指数的浮点数）。我们在约 70 亿参数的稠密语言模型上验证了我们的 LogFMT-nBit，通过量化残差分支的输出以模拟 MoE 模型中的 `combine` 阶段。当设置 $n = 8$（与 FP8 位数相同）时，LogFMT-8Bit 相比 E4M3 或 E5M2 表现出更优的训练精度。将 n 增加到 10 位后，我们发现其效果与 BF16 `combine` 阶段相似。

3.2.1 **局限性**：. 使用 LogFMT 的初衷是将其应用于传输过程中的激活值或激活函数附近，因为它在相同位宽下能提供比 FP8 更高的精度。然而，后续计算需要重新转换回 BF16 或 FP8 以适应 Hopper GPU Tensor Core 的数据类型。由于 log/exp 操作的 GPU 带宽不足，以及编解码过程中的寄存器压力过大，如果将编解码操作与 all-to-all 通信融合，开销将非常巨大（50%~100%）。因此，尽管实验结果验证了该格式的有效性，我们最终并未采用它。

3.2.2 **建议**：. 为针对 FP8 或自定义精度格式量身定制的压缩和解压缩单元提供原生支持，是未来硬件的一个可行方向。这有助于最小化带宽需求并简化通信流水线。降低的通信开销在 MoE 训练等带宽密集型任务中尤为有益。

4 互连驱动设计

4.1 当前硬件架构

我们当前使用的 NVIDIA H800 GPU SXM 架构（如图 2 所示）基于 Hopper 架构构建，与 H100 GPU 类似。然而，出于合规要求，其 FP64 计算性能和 NVLink 带宽有所降低。具体而言，H800 SXM 节点中的 NVLink 带

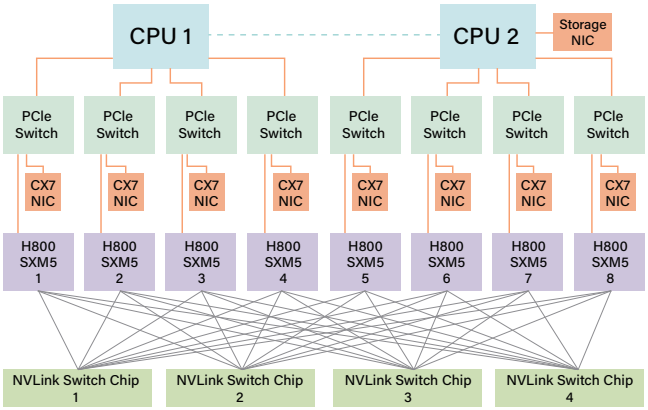

Figure 2: H800 节点互连。

宽从 900 GB/s 降至 400 GB/s。节点内 Scale-up 带宽的大幅缩减给高性能工作负载带来了挑战。为弥补这一不足，每个节点配备了八张 400G Infiniband (IB) CX7 网卡，增强了 Scale-out 能力以缓解带宽缺口。

为应对这些硬件限制，DeepSeek-V3 模型融入了多项与硬件优势及局限性相匹配的设计考量。

4.2 硬件感知并行策略

为契合 H800 架构的限制，我们考虑了以下并行策略以优化 DeepSeek-V3 的性能：

- **避免使用张量并行（TP）：**由于在有限的 NVLink 带宽下效率较低，训练期间避免了使用张量并行。然而，在推理阶段，仍可选择性使用 TP 以提升 TTFT 和 TPOT 性能。
- **增强型流水线并行（PP）：**采用 DualPipe [?] 将 Attention 和 MoE 计算与 MoE 通信重叠。这也有助于减少流水线气泡并平衡各 GPU 间的内存使用，从而提升整体吞吐量。更多细节请参阅技术报告 [?]。
- **加速专家并行（EP）：**借助八张 400Gbps InfiniBand (IB) 网卡，系统实现了超过 40GB/s 的 all-to-all 通信速度。值得注意的是，我们的 all-to-all EP 实现 DeepEP [?] 已开源，如下一小节所述，它实现了高度高效的专家并行。

4.3 模型协同设计：节点受限路由

在 H800 架构中，Scale-up（节点内）与 Scale-out（节点间）通信的带宽差异约为 4:1。具体而言，NVLink 提供 200GB/s 的带宽（实际可达约 160GB/s），而每张 400Gbps IB 网卡仅提供 50GB/s 的带宽（考虑到小消息大小和延迟的影响，有效带宽按 40GB/s 计算）。为平衡并充分利用较高的节点内带宽，模型架构与硬件进行了协同设计，特别是在 **TopK 专家选择策略**方面。Consider a setup with 8 nodes (64 GPUs in total) and 256 routed experts (4 experts per GPU). 对于 DeepSeek-V3，每个 token 会被路由到一个共享专家和 8 个路由专家。如果这 8 个目标专家分布在全部 8 个节点上，通过 IB 的通信时间将为 $8t$，其中 t 表示通过 IB 发送一个 token 所需的时间。然而，通过利用更高的 NVLink 带宽，路由到同一节点内的 token 只需通过 IB 发送一次，随后即可通过 NVLink 转发至节点内的其他 GPU。NVLink 转发机制使得 IB 流量得以去重。当给定 token 的目标专家分布在 M 个节点上时，去重后的 IB 通信开销将降低至 Mt（$M < 8$）。

由于 IB 流量仅取决于 M，DeepSeek-V3 为 TopK 专家选择策略引入了**节点受限路由（Node-Limited Routing）**。具体而言，我们将 256 个路由专家划分为 8 组，每组 32 个专家，并将每组部署在单个节点上。在此部署基础上，我们通过算法确保每个 token 最多仅被路由到 4 个节点。该方法缓解了 IB 通信的瓶颈，并提升了训练过程中的有效通信带宽。

4.4 Scale-Up 与 Scale-Out 的融合

4.4.1 **当前实现的局限性.** 尽管节点受限路由策略降低了通信带宽需求，但由于节点内（NVLink）与节点间（IB）互连带宽存在差异，它使得通信流水线内核的实现变得复杂。在实际应用中，GPU 流多处理器（SM）线程既用于处理网络消息（例如填充 QP 和 WQE），也用于 NVLink 上的数据转发，从而消耗计算资源。例如，在训练期间，H800 GPU 上多达 20 个 SM 被分配用于通信相关操作，导致可用于实际计算的资源减少。为了在在线推理中最大化吞吐量，我们完全通过 NIC RDMA 执行 EP all-to-all 通信，从而避免 SM 资源争用并提升计算效率。这凸显了 RDMA 异步通信模型在重叠计算与通信方面的优势。

以下是EP通信期间SM当前执行的关键任务，特别是针对 `combine` 阶段的 `reduce` 操作和数据类型转换。将这些任务卸载到专用通信硬件可以释放SM用于计算内核，从而显著提高整体效率：

- **数据转发：** 在IB和NVLink域之间聚合发往同一节点内多张GPU的IB流量。
- **数据传输：** 在RDMA缓冲区（已注册的GPU内存区域）与输入/输出缓冲区之间移动数据。
- **Reduce操作：** 执行EP `all-to-all` `combine` 通信所需的 `reduce` 操作。
- **内存布局管理：** 处理跨IB和NVLink域的分块数据传输的细粒度内存布局。
- **数据类型转换：** 在 `all-to- all` 通信前后进行数据类型转换。

4.4.2 **建议**：. 为解决这些低效问题，我们强烈建议未来硬件应将节点内（scale-up）与节点间（scale-out）通信整合到一个统一的框架中。通过引入用于网络流量管理的专用协处理器，并实现NVLink与IB域之间的无缝转发，此类设计可降低软件复杂度并最大化带宽利用率。例如，DeepSeek-V3中采用的节点受限路由策略可在硬件支持动态流量去重的情况下进一步优化。

我们也关注到新兴的互连协议，如超以太网联盟（UEC）[??]和超加速器链路（UALink）[?]，它们有望推动scale-up与scale-out通信的进步。最近，统一总线（UB）[?]提出了一种实现scale-up与scale-out融合的新方法。第6节进一步探讨了UEC和UALink提出的几项技术创新。然而，在本节中，我们的主要焦点在于在编程框架层面实现scale-up与scale-out的融合：

(1) **统一网络适配器：** 设计连接到统一scale-up和scale-out网络的网卡（NIC）或I/O Die。这些适配器还应支持基本交换功能，例如将数据包从scale-out网络转发至scale-up网络内的特定GPU。这可以通过使用单个LID（本地标识符）或IP地址结合基于策略的路由来实现。
(2) **专用通信协处理器：** 引入专用协处理器或可编程组件（如I/O die）来处理网络流量。该组件应从GPU SM卸载数据包处理任务，提供硬件加速的内存拷贝以实现高效的缓冲区管理，并且关键的是，以类似于TMA（张量内存加速器）的方式加速内存加载/存储操作，从而以最小的资源消耗使带宽达到饱和。
(3) **灵活的转发、广播与Reduce机制：** 硬件应支持跨scale-up和scale-out网络的灵活转发、广播操作（用于EP dispatch）和reduce操作（用于EP combine）——这与我们当前基于GPU SM的实现相呼应。这不仅将提升有效带宽，还将降低网络特定操作的计算复杂度。
(4) **硬件同步原语：** 提供细粒度的硬件同步指令，以在硬件层面处理内存一致性问题或乱序到达的数据包。这将消除对基于软件的同步机制（如RDMA完成事件）的需求，后者会引入额外延迟并增加编程复杂度。采用acquire/release机制的内存语义通信是一种很有前景的实现方案。

通过落实这些建议，未来的硬件设计可在简化软件开发的同时，显著提升大规模分布式AI系统的效率。

4.5 带宽争用与延迟

4.5.1 **局限性**：. 此外，当前硬件缺乏在NVLink和PCIe上动态分配不同类型流量带宽的灵活性。例如，在推理期间，将KV cache数据从CPU内存传输到GPU可能消耗数十GB/s的带宽，导致PCIe带宽饱和。如果GPU同时使用IB进行EP通信，KV cache传输与EP通信之间的这种争用会降低整体性能并导致延迟激增。

4.5.2 **建议**：.

- **动态NVLink/PCIe流量优先级：** 硬件应支持基于流量类型的动态优先级分配。例如，应分别为与EP、TP和KV cache传输相关的流量分配不同的优先级，以最大化互连效率。对于PCIe，将流量类别（TC）暴露给用户级编程即可。
- **I/O Die Chiplet集成：** 将NIC直接集成到I/O die中，并在同一封装内将其与计算die连接，而非通过传统PCIe，将大幅降低通信延迟并缓解PCIe带宽争用。

- **Scale-Up 域内的 CPU-GPU 互连：**为进一步优化节点内通信，CPU 与 GPU 应使用 NVLink 或类似专用高带宽互连架构进行连接，而非仅依赖 PCIe。与将 NIC 集成到 I/O die 所带来的优势类似，该方法可显著改善训练和推理期间在 GPU 与 CPU 内存之间卸载参数或 KV cache 等场景。

5 大规模网络驱动设计

5.1 网络协同设计：多平面胖树

在 DeepSeek-V3 的训练过程中，我们部署了**多平面胖树（MPFT）** scale-out 网络，如图 3 所示。每个节点配备八张 GPU 和八张 IB NIC，每对 GPU-NIC 被分配至不同的网络平面。此外，每个节点还配备一张 400 Gbps 以太网 RoCE NIC，连接到独立的存储网络平面，用于访问 3FS [?] 分布式文件系统。在 scale-out 网络中，我们使用了 64 端口 400G IB 交换机，使得该拓扑在理论上最多支持 16,384 张 GPU，同时保留了双层网络的成本与延迟优势。然而，受政策和监管限制，最终仅部署了略多于两千张 GPU。

此外，受限于当前 IB ConnectX-7 的能力，我们部署的 MPFT 网络并未完全实现预期的架构。理想情况下，如图 4 所示，每张 NIC 将具备多个物理端口，每个端口连接到独立的网络平面，但通过端口绑定共同向用户暴露为单个逻辑接口。从用户角度来看，单个队列对（QP）可无缝跨所有可用端口收发消息，类似

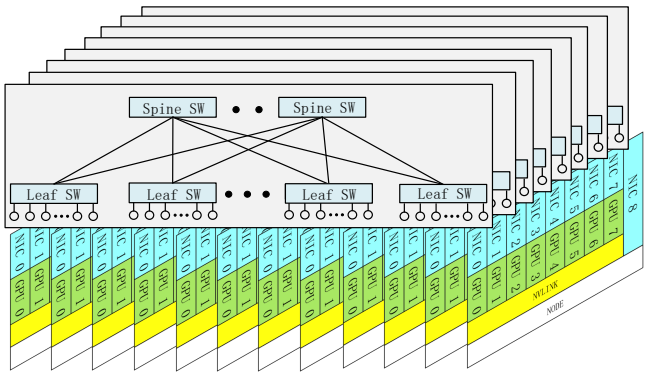

Figure 3: 八平面双层胖树 scale-out 网络：每对 GPU 和 IB NIC 属于一个网络平面。跨平面流量必须使用另一张 NIC 以及 PCIe 或 NVLink 进行节点内转发。

Table 3: 网络拓扑对比。成本估算基于 Slim Fly (SF) 论文 [?] 中的方法。DF 表示经典的 Dragonfly 拓扑 [? ? ?]。

指标	FT2	MPFT	FT3	SF	DF
端点数量	2,048	16,384	65,536	32,928	261,632
交换机数量	96	768	5,120	1,568	16,352
链路数量	2,048	16,384	131,072	32,928	384,272
成本 [百万美元]	9	72	491	146	1,522
单端点成本 [千美元]	4.39	4.39	7.5	4.4	5.8

于数据包喷洒（packet spraying）。因此，源自同一 QP 的数据包可能经过不同的网络路径，并以乱序方式到达接收端，这就要求 NIC 原生支持乱序放置，以保证消息一致性并维护正确的排序语义。例如，InfiniBand ConnectX-8 原生支持四个平面。未来 NIC 若能全面支持高级多平面能力，将有利于双层胖树网络有效扩展至更大规模的 AI 集群。总体而言，多平面架构在故障隔离、鲁棒性、负载均衡和大规模系统可扩展性方面具有显著优势。

5.1.1 多平面胖树网络的优势.

- **多轨胖树（MRFT）的子集：** MPFT 拓扑构成了更广泛 MRFT 架构的一个特定子集。因此，NVIDIA 和 NCCL 为多轨网络开发的现有优化可无缝应用于多平面网络部署中。此外，NCCL 对 PXN [?] 技术的支持解决了平面间隔离的固有挑战，即使在平面间缺乏直接互连的情况下也能实现高效通信。
- **成本效益：**如表 3 所示，多平面网络采用双层胖树（FT2）拓扑即可支持超过 10,000 个端点，与三层胖树（FT3）相比显著降低了网络成本。其单端点成本甚至比注重成本效益的 Slim Fly（SF）拓扑 [?] 更具竞争力。
- **流量隔离：**每个平面独立运行，确保一个平面的拥塞不会影响其他平面。这种隔离机制提升了整体网络稳定性，并防止了性能的级联式下降。
- **延迟降低：**实验表明，双层拓扑的延迟低于三层胖树。这使其特别适用于对延迟敏感的应用，例如基于 MoE 的训练与推理。
- **鲁棒性：**如图 4 所示，多端口 NIC 提供了多条上行链路，因此单端口故障不会中断连接，并可实现快速、透明的故障恢复。

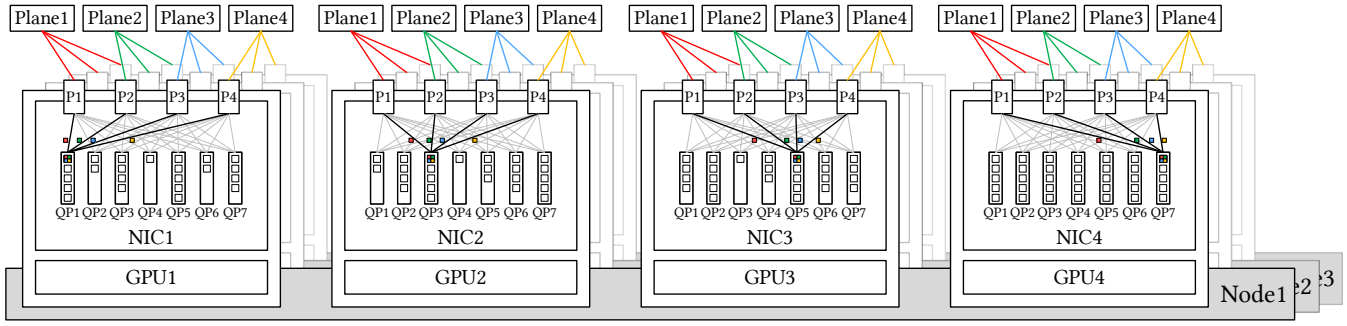

Figure 4: 理想的多平面网络：每张 NIC 配备多个物理端口，每个端口连接到不同的网络平面。单个队列对（QP）可同时利用所有可用端口进行数据包收发，这需要 NIC 原生支持乱序放置。

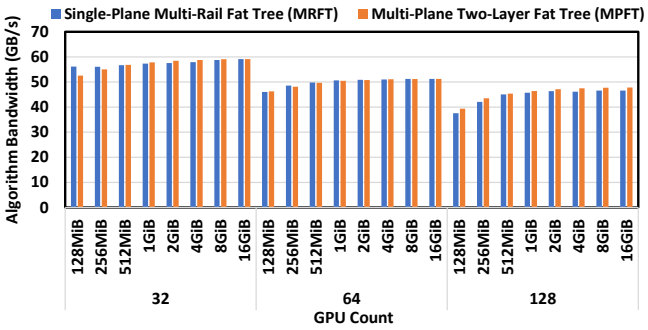

Figure 5: MRFT 与 MPFT 网络在 32 至 128 张 GPU 规模下的 NCCL All-to-All 性能对比。

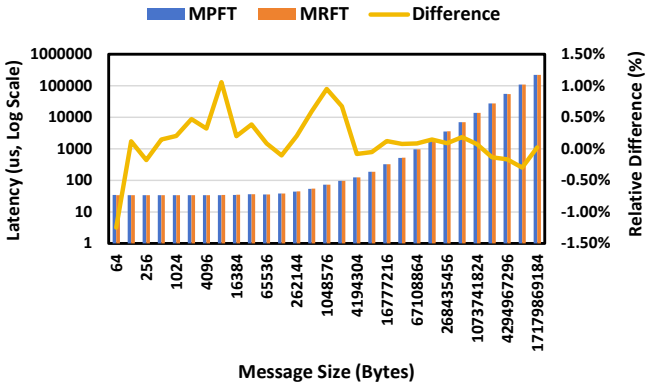

Figure 6: 不同消息大小下 MPFT 与 MRFT 网络在 NCCL All-to-All 测试中的延迟对比，结果表明两者性能几乎一致。

需要指出的是，受限于当前 400G NDR InfiniBand 的能力，跨平面通信需要节点内转发，这会在推理过程中引入额外延迟。如果未来硬件能够实现前文所述的 Scale-up 与 Scale-out 网络融合，该延迟将大幅降低，从而进一步提升多平面网络的可行性。

5.1.2 **性能分析**. 为验证多平面网络设计的有效性，我们在集群上进行了实际实验，通过修改集群网络拓扑，对比了 **多平面双层胖树（MPFT）** 与 **单平面多轨胖树（MRFT）** 的性能。以下是实验的关键发现：

1. **All-to-All 通信与 EP 场景**：如图 5 所示，多平面网络的 All-to-All 性能与单平面多轨网络非常接近。这种性能对等性可归因于 NCCL 的 PXN [?] 机制，该机制在多轨拓扑中通过 NVLink 优化了流量转发。多平面拓扑同样受益于该机制。如图 6 所示，在 16 张 GPU 上进行的 All-to-All 通信测试结果表明，MPFT 与 MRFT 拓扑之间的延迟差异微乎其微。

为评估 MPFT 在实际训练场景中 All-to-All 通信的性能，我们测试了训练过程中常用的 EP 通信模式。如图 7 所示，在多平面网络中，每张 GPU 均实现了超过 40GB/s 的高带宽，提供了满足训练需求的可靠性能。

2. **DeepSeek-V3 模型训练吞吐量**：我们还在表 4 中对比了 MPFT 与 MRFT 下 DeepSeek-V3 模型的训练指标。MFU（模型算力利用率）基于 BF16 峰值性能计算。Causal MFU 仅考虑注意力矩阵下三角部分的 FLOPs（与 FlashAttention[? ?] 一致），而非 Causal MFU 则包含整个注意力矩阵的 FLOPs（与 Megatron [?] 一致）。1F、1B 和 1W 分别表示前向传播时间、输入反向传播时间和权重反向传播时间。在 2048 张 GPU 上训

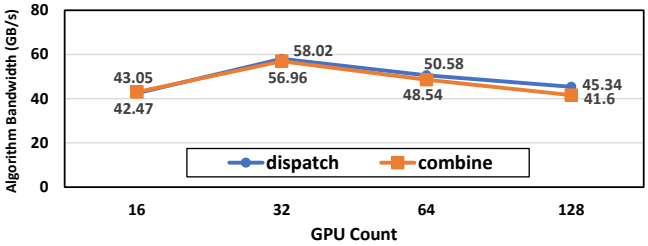

Figure 7: MPFT 上的 DeepEP 性能：EP 的 dispatch 和 combine 内核通过 All-to-All 在 16 至 128 张 GPU 间进行通信。每张 GPU 处理 4096 个 token。观测到的吞吐量几乎饱和了 400Gps NIC 带宽。

Table 4: MPFT 与 MRFT 网络的训练指标对比。

指标	MPFT	MRFT
tokens/天 (B)	272.80	272.52
单步耗时 (s)	19.926	19.946
1F (s)	1.13	1.13
bubble (s)	2.06	2.03
1B (s)	1.99	1.99
1W (s)	0.48	0.48
1F1B (s)	13.95	14.00
opt (s)	0.29	0.31
TFLOPS (非因果)	432	432
TFLOPS (因果)	385	385
MFU (非因果)	43.73%	43.68%
MFU (因果)	38.94%	38.90%

练 V3 模型时，MPFT 的性能与 MRFT 几乎完全一致，观测到的差异均在正常波动和测量误差范围内。

5.2 低延迟网络

在我们的模型推理中，大规模 EP 严重依赖 `all-to-all` 通信，该通信对带宽和延迟均高度敏感。考虑第 2.3.2 节中讨论的典型场景，在 50GB/s 的网络带宽下，数据传输理想情况下应耗时约 120 μs 。因此，微秒级的固有网络延迟会对系统性能产生关键影响，其作用不可忽略。

5.2.1 IB 与 RoCE **的选择**. 如表 5 所示，IB 始终能实现更低的延迟，使其成为分布式训练和推理等延迟敏感型工作负载的首选。尽管 IB 在延迟性能上优于融合以太网上的 RDMA（RoCE），但它也存在一定的局限性：

- **成本：** IB 硬件的成本显著高于 RoCE 解决方案，这限制了其广泛部署。

Table 5: IB、RoCE 与节点内 NVLink 在传输 64B 数据时的 CPU 端端到端延迟对比。

链路层	同 Leaf	跨 Leaf
RoCE	3.6us	5.6us
InfiniBand	2.8us	3.7us
NVLink	3.33us	-

- **可扩展性：** IB 交换机通常每台仅支持 64 个端口，而 RoCE 交换机通常提供 128 个端口。这限制了基于 IB 的集群的可扩展性，尤其是在大规模部署场景下。

5.2.2 RoCE **改进建议**. 尽管 RoCE 有潜力成为 IB 的性价比替代方案，但其在延迟和可扩展性方面的当前局限性使其难以完全满足大规模 AI 系统的需求。以下我们提出针对 RoCE 改进的具体建议：

(1) **专用低延迟 RoCE 交换机：** 我们建议以太网厂商通过移除不必要的以太网功能，开发专门针对 RDMA 工作负载优化的 RoCE 交换机。Slingshot 架构 [?] 展示了基于以太网的设计如何实现与 IB 相当的延迟性能。同样，Broadcom [?] 近期的创新，包括 AI 转发头（AIFH）和即将推出的低延迟以太网交换机，证明了为 AI 定制的高性能以太网交换架构的可行性。我们期待该方向持续创新。

(2) **优化的路由策略：** 如图 8 所示，RoCE 默认的等成本多路径（ECMP）路由策略难以在互联链路上高效分配流量，导致在 NCCL 集合通信测试中出现严重的拥塞性能下降。LLM 训练流量（如 DP 数据并行）往往缺乏随机性，导致多个流汇聚到同一条互联链路上。相比之下，自适应路由（AR）[?] 可通过在多路径间动态喷洒数据包来显著提升网络性能。虽然基于手动配置路由表的静态路由可以避免特定目的地的链路冲突，但缺乏灵活性。对于大规模 All-to-All 通信，自适应路由提供了更优的性能和可扩展性。

(3) **改进的流量隔离或拥塞控制机制：** 当前 RoCE 交换机仅支持有限数量的优先级队列，不足以应对涉及 EP 的 All-to-All 和 DP 的 All-Reduce 等并发通信模式的复杂 AI 工作负载。在此类混合负载中，All-to-All 流量可能因突发性的多对一传输引发入站拥塞（incast congestion），进而降低整体网络性能。为缓

解入站拥塞对其他流量的影响，一种方法是采用虚拟输出队列（VOQ），为每个 QP 分配专用的虚拟队列以隔离流量。另一种方法是采用更有效的拥塞控制（CC）机制，如基于 RTT 的 CC（RTTCC）或用户可编程 CC（PCC），从而实现 NIC 与交换机的协同优化，在动态流量条件下保持低延迟和高吞吐量。

5.2.3 InfiniBand GPUDirect Async (IBGDA).

5.2.4 InfiniBand GPUDirect Async (IBGDA). We utilize IBGDA [? ?] to reduce latency in network communications. Traditionally, network communication involves the creation of a CPU proxy thread: once the GPU has prepared the data, it must notify the CPU proxy, which then populates the control information for the work request (WR) and signals the NIC via a doorbell mechanism to initiate data transmission. This process introduces additional communication overhead.

IBGDA addresses this issue by allowing the GPU to directly fill the WR content and write to the RDMA doorbell MMIO address. By managing the entire control plane within the GPU, IBGDA eliminates the significant latency overhead associated with GPU-CPU communication. Moreover, when sending a large number of small packets, the control plane processor can easily become a bottleneck. Since GPUs have multiple parallel threads, the sender can leverage these threads to distribute the workload, thereby avoiding such bottlenecks. A range of works—including our DeepEP [?]—have leveraged IBGDA and reported substantial performance gains [? ? ?]. We therefore advocate for such capabilities to be widely supported across accelerator devices.

6 未来硬件架构设计的讨论与启示

基于前述章节，我们总结了关键的架构启示，并概述了面向大规模 AI 工作负载的硬件设计未来方向。

第 2.3.2 节强调了大规模 Scale-up 网络在加速模型推理方面的重要性。第 3 节讨论了高效支持低精度

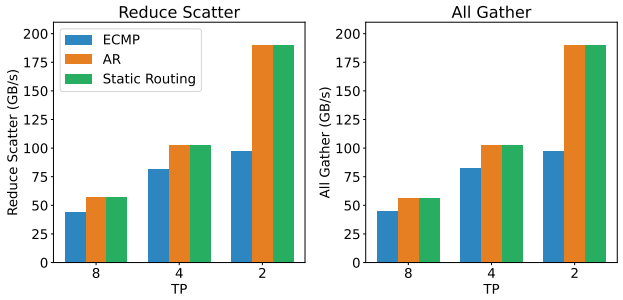

Figure 8: 在不同路由方法（ECMP、AR、静态路由）和 TP 维度下，AllGather 和 ReduceScatter 通信原语的 RoCE 网络带宽。

计算与通信的必要性。第 4 节探讨了 Scale-up 与 Scale-out 架构的融合，并提出了几项改进建议。第 5 节聚焦于多平面网络拓扑，并指出了基于以太网的互连所需的关键改进。

综合来看，这些章节在具体应用背景下指出了硬件的局限性，并提供了相应的建议。在此基础上，本节将讨论扩展到更广泛的考量，并为未来硬件架构设计提出前瞻性的方向。

6.1 鲁棒性挑战

6.1.1 **局限性**：.

- **互连故障：** 高性能互连（如 IB 和 NVLink）容易出现间歇性断开，这会破坏节点间的通信。在 EP 等通信密集型工作负载中，这一问题尤为致命，即使短暂的中断也可能导致性能显著下降或任务失败。
- **单点硬件故障：** 节点崩溃、GPU 故障或 ECC（纠错码）内存错误会破坏长时间运行的训练任务，通常需要代价高昂的重启。在大规模部署中，此类故障的影响会加剧，因为单点故障的概率与系统规模成正比增加。
- **静默数据损坏：** ECC 机制未能检测到的错误（如多位内存翻转或计算不精确）对模型质量构成重大风险。在长时间运行的任务中，这些错误尤为隐蔽，因为它们可能在未被察觉的情况下传播并破坏下游计算。当前的缓解策略依赖于应用层启发式方法，不足以确保系统级的鲁棒性。

6.1.2 **高级错误检测与纠正建议**. 为缓解静默损坏相关的风险，硬件必须引入超越传统 ECC 的高级错误检测机制。基于校验和的验证或硬件加速的冗余检查等技术可为大规模部署提供更高的可靠性。

此外，硬件供应商应向终端用户提供全面的诊断工具包，使其能够严格验证系统完整性，并主动识别任何潜在的静默数据损坏。将此类工具包作为标准硬件套件的一部分嵌入，有助于提升透明度，并在整个运行生命周期内实现持续验证，从而增强整体系统的可信度。

6.2 CPU 瓶颈与互连

尽管加速器设计往往占据核心地位，但 CPU 在协调计算、管理 I/O 以及维持系统吞吐量方面仍然不可或缺。然而，当前架构面临几个关键瓶颈：

首先，如第 4.5 节所述，CPU 与 GPU 之间的 PCIe 接口经常成为带宽瓶颈，尤其是在大规模参数、梯度或 KV 缓存传输期间。为缓解这一问题，未来系统应采用直接的 CPU-GPU 互连（如 NVLink 或 Infinity Fabric），或将 CPU 和 GPU 均集成到 Scale-up 域中，从而消除节点内瓶颈。

除了 PCIe 的限制外，维持如此高的数据传输速率还需要极高的内存带宽。例如，跑满 160 条 PCIe 5.0 通道需要每个节点超过 640 GB/s 的带宽，这意味着每个节点的内存带宽需求约为 1 TB/s——这对传统 DRAM 架构构成了重大挑战。

最后，内核启动和网络处理等延迟敏感型任务需要极高的 CPU 单核性能，通常要求基础频率高于 4 GHz。此外，现代 AI 工作负载要求每个 GPU 配备充足的 CPU 核心，以防止控制侧出现瓶颈。对于基于 Chiplet 的架构，还需要额外的核心来支持缓存感知的负载划分与隔离。

6.3 面向 AI 的智能网络

为满足延迟敏感型工作负载的需求，未来互连必须同时优先考虑低延迟与智能网络：

- **共封装光学（CPO）：** 集成硅光子技术可实现可扩展的高带宽与增强的能效，这两者对大规模分布式系统至关重要。
- **无损网络：** 基于信用的流量控制（CBFC）机制可确保无损数据传输，但盲目触发流量控制可能导致严重的队头阻塞。因此，必须部署先进的、端点驱动的拥塞控制（CC）算法，以主动调节注入速率并避免病态拥塞场景。
- **自适应路由：** 正如第 5.2.2 节所强调的，未来网络应标准化采用动态路由方案（如数据包喷洒和拥塞感知路径选择），持续监控实时网络状况并智能重分配流量。这些自适应策略在缓解热点和减轻集合通信工作负载（包括 All-to-All 和 Reduce-Scatter 操作）期间的瓶颈方面尤为有效。
- **高效容错协议：** 通过部署自愈协议、冗余端口和快速故障转移技术，可显著增强对故障的鲁棒性。例如，链路层重试机制和选择性重传协议在提升大型网络可靠性方面不可或缺，能够最大限度地减少停机时间，并确保在间歇性故障下无缝运行。
- **动态资源管理：** 为有效处理混合工作负载，未来硬件应支持动态带宽分配和流量优先级管理。例如，在统一集群中应将推理任务与训练流量隔离，以确保延迟敏感型应用的响应能力。

6.4 内存语义通信与排序问题讨论

使用 Load/Store 内存语义的节点间通信高效且对程序员友好，但当前实现受限于内存排序挑战。例如，写入数据后，发送方必须在更新标志以通知接收方之前发出显式的内存屏障（fence），以确保数据一致性。这种严格的排序引入了额外的往返时间（RTT）延迟，并可能导致发出线程停滞，阻碍在途存储操作并降低吞吐量。类似的乱序同步问题也出现在消息语义 RDMA 中；例如，在 InfiniBand 或 NVIDIA BlueField-3 上执行常规 RDMA 写入后，再进行带数据包喷洒的 RDMA 原子加操作，会产生额外的 RTT 延迟。

为解决这些问题，我们主张提供硬件支持，为内存语义通信提供内置的排序保证。这种一致性应在编程层面（例如通过 acquire/release 语义）和接收方硬

件层面同时强制执行，从而实现无额外开销的按序交付。

存在多种可行的方法。例如，接收方可以缓冲原子消息并使用数据包序列号（PSN）确保按序处理；该方法简单直接，能有效维持正确性。另一种更具吸引力的方法是基于区域的 acquire/release（RAR）机制，因为它能为远程内存访问提供真正的 acquire/release 语义，并为各类工作负载提供更大的灵活性。在此方法中，接收方硬件维护轻量级元数据（如位图或基于区域的计数器）以跟踪内存区域的状态。Acquire 和 release 操作限定于特定的地址范围，从而无需发送方显式 fence 即可实现高效的硬件强制排序。值得注意的是，这两种方法均易于在 NIC 或 I/O 芯片上实现，且适用于内存语义和消息语义的 RDMA 原语，从而拓宽了其实际应用价值。

6.5 网络内计算与压缩

EP 涉及两个关键的 `all-to-all` 阶段——**分发（dispatch）**和 **合并（combine）**，这为网络内优化提供了重要契机。**分发**阶段类似于小规模组播操作，需要将单条消息转发至多个目标设备。一种支持自动数据包复制并转发至多目的地的硬件级协议可大幅降低通信开销并提升效率。

作为小规模归约操作的 **合并**阶段，可从网络内聚合技术中受益。然而，由于 EP 合并阶段的归约范围较小且负载不均衡，以灵活的方式实现网络内聚合颇具挑战。

此外，如第 3.2 节所述，LogFMT 能够在对模型性能影响极小的情况下实现低精度 Token 传输。在网络硬件中原生集成 LogFMT 可通过提高熵密度和降低带宽使用率进一步优化通信。硬件加速的压缩与解压将使 LogFMT 无缝集成到分布式系统中，从而提升整体吞吐量。

6.6 以内存为中心的创新

6.6.1 **内存带宽的局限性**. 模型规模的指数级增长已超越了高带宽内存（HBM）技术的进步。这种差距导致了内存瓶颈，尤其是在以注意力机制为主的架构（如 Transformer）中尤为明显。

6.6.2 **建议**：.

- **DRAM 堆叠加速器：**借助先进的 3D 堆叠技术，DRAM 芯片可垂直集成于逻辑芯片之上，从而实现极高的内存带宽、超低延迟以及实用的内存容量（尽管受限于堆叠层数）。这种架构范式对 MoE 模型的超快推理具有显著优势，因为内存吞吐量在此类模型中是关键瓶颈。SeDRAM[?] 等架构充分展现了该方法的潜力，为内存密集型工作负载带来了前所未有的性能提升。
- **晶圆级系统（SoW）：** 晶圆级集成 [?] 能够最大化计算密度与内存带宽，满足超大规模模型的需求。

7 结论

DeepSeek-V3 充分展现了软硬件协同设计在提升大规模 AI 系统可扩展性、效率与鲁棒性方面的变革潜力。通过剖析当前硬件架构的局限性并提出切实可行的建议，本文为下一代 AI 优化硬件的发展提供了路线图。随着 AI 工作负载在复杂度和规模上持续增长，这些创新对于推动智能系统的未来发展至关重要。