H3C
数字化及AI解决方案领导者

超节点技术白皮书 2026

新华三集团

目 录

前言

当前，大模型训练已从单机计算演进为超大规模集群的协同工程。随着模型参数量向万亿级迈进，万卡级集群已成为头部算法厂商的标配。在这一演进过程中，底座架构的瓶颈正在从单卡的算力峰值（TFLOPS）转向集群维度的通信效率（MFU-Model FLOPs Utilization）。当训练任务分布在数千甚至上万个加速节点上时，All-to-All、All-Reduce 等集合通信操作产生的流量呈几何倍数增长，如何降低长尾延迟、提升有效算力利用率（MFU），成为大规模工程落地中首要解决的硬核课题。

传统的服务器架构在应对超大规模并发训练时，正面临严重的扩展性挑战。受限于标准机架的物理空间与功耗限制，单机内部的互联带宽虽高，但跨机通信仍主要依赖于 RDMA 网络。在万卡规模下，多级交换机带来的跳数增加（Hops）以及网络拥塞，会导致计算单元在等待参数、数据的同步时出现严重的“气泡”。传统“1 机 8 卡”的结构，机内 TB 级总线与机间 Gbps 网络存在量级带宽差，且协议转换引入微秒级延迟。这种内存共享空间（HBM P2P）和互联带宽之间明显的物理断层，难以支撑千亿级以上参数模型对低延迟数据对撞的苛刻要求，制约了集群算力扩展。

“超节点”架构的出现，本质上是对计算、网络与存储边界的一次系统性重构。它不再局限于传统的单机物理框限制，而是通过高密度的背板互联或高速互联协议（如高性能 NVLink 扩展或自研互联架构），将多个计算单元在逻辑上整合成一个超大规模的计算实例。超节点的技术核心在于将“Scale-up”域从单机扩展至机柜级甚至更大规模，利用拓扑结构设计和统一的内存寻址能力，显著缩短通信路径。这种架构在保持高吞吐的同时，有效缓解了大规模线性扩展时的性能损耗。

本白皮书旨在通过对超节点产品的系统性拆解，从硬件拓扑、互联协议、系统工程以及软件栈协同等维度，探讨如何构建下一代 AI 算力底座。我们希望通过对技术细节的深入剖析，为构建高性能、高可靠的 AI 基础设施提供切实可行的参考路径，支撑大模型走向超大规模生产化应用。

超节点技术基础

1 定义与核心特征

1.1 超节点技术背景与演进必然性

AI 算力需求爆发式增长驱动

1. 算力规模持续攀升

国际数据公司（IDC）发布的《2025 年中国人工智能计算力发展评估报告》显示，中国智能算力将保持高速增长态势：2025 年规模预计达到 1037.3 EFLOPS，同比增长 43%；2026 年将达到 1460.3 EFLOPS，为 2024 年的两倍。2023-2028 年期间，智能算力五年复合增长率预计达到 46.2%，远超通用算力 18.8%的增长率。这一差距反映出 AI 专用计算需求的独特性和紧迫性。

与此同时，推理算力占比将持续提升，预计到 2028 年达到 73%的市场份额。这一结构性变化表明，AI 应用正从训练为主向训练推理并重转型，对计算基础设施的功耗、时延等指标提出了更高要求。Epoch 的《AI in 2030》预测更为激进：按照当前发展趋势，2030 年最大的人工智能模型将需要数千亿美元投资，其计算量将是当今最大模型的 1000 倍，训练运行所需的电力将需要数十亿瓦特的电力，接近整个大型城市的平均需求。

2. 大模型技术演进推动架构变革

从数十亿到万亿级，大模型参数规模的快速扩张已超出单卡甚至单服务器的承载能力。2024 年 12 月发布的 DeepSeek V3 参数量达到 671B，2025 年 7 月的 Kimi K2 突破万亿大关。以 GPT-4 的 1.8 万亿参数为例，模型部署需要超过 10TB 显存占用，远超单卡显存 HBM 容量。这种规模的部署必须依赖多维混合并行策略，如张量并行（TP）、流水线并行（PP）、专家并行（EP）、序列并行（SP）和数据并行（DP）的组合使用。如此规模的并行化带来的巨大通信挑战，巨大的集合通信量对网络基础设施提出极端要求：既要大带宽（单批次数百 GB），又要低时延（并行域内通信更为迫切）。

算力与网络联接必须协同发展、相互匹配，才能支撑新一代 AI 系统。英伟达在 GTC 2026 中提出了 7 芯片极致协同的集群架构，正是这一思想的典型体现。这 7 类芯片包括：核心算力芯片 Rubin GPU、系统调度与控制芯片 Vera CPU、芯片间 Scale UP 高速互联芯片 NVLink 6 Switch、芯片间 Scale OUT 高速互联网卡芯片 ConnectX-9 SuperNIC、负责存储与基础设施卸载芯片 BlueField-4 DPU、大规模高速 Scale OUT

交换芯片 Spectrum-X CPO Switch、低时延推理专用 FFN 引擎 Groq 3 LPU。通过多芯片分工协作、深度融合，可以释放出远超单一芯片的整体算力。

超节点正是这一理念的工程化实现：一方面，提供极致性能的 AI 算力；另一方面，构建大规模、高带宽、低时延的 AI 芯片高速互联网络。通过让 AI 芯片算力与 AI 芯片互联能力最优匹配，形成“强算力 × 强互联”的高效架构，从而稳定支撑超大参数模型的训练、PD 分离推理、AF 分离解码（Attention-FFN 分离）等。这一架构最终指向智算中心的两大核心目标：

- 更低的单 Token 成本。
- 更低的单 Token 能耗。

大模型后训练与强化学习的通信挑战

1. RLHF 多模型协同通信

大模型后训练的人类反馈强化学习（RLHF）流程包括监督微调（SFT）、奖励模型训练（RM）和强化学习训练（RL Training，强化学习算法通常选择近端策略优化 Proximal Policy Optimization，PPO）三个阶段，每个阶段都需要维护百亿甚至万亿参数规模的大型模型。以 ChatGPT 和 Claude 等系统为例，RLHF 过程需要同时运行策略模型、价值模型、奖励模型和参考模型，总参数量可达数万亿级别。

RLHF 需要在策略网络推理、奖励计算、优势估计和策略更新之间进行复杂的数据流转需要进行大量模型参数传输和梯度同步。以 GPT-3 的训练配置为例，当使用 175B 参数的策略模型和 6.7B 参数的奖励模型时，单次 PPO 迭代的模型间通信量可达 500GB 以上。传统服务器堆叠集群的网络带宽瓶颈往往导致整体训练效率下降 60%（NSDI'23 论文《TOPOOPT》）。

2. 大规模样本生成与并行推理

大规模 RLHF 训练需要为每个提示生成数十个候选响应，可达数万个样本。这要求系统能够高效处理大规模的序列生成和并行推理任务，同时奖励模型的评分结果需要及时反馈给策略优化过程。任何通信时延都会直接影响训练的收敛速度和稳定性，对通信基础设施的时延特性提出极高要求。

3. 通信墙

传统 Scale-out 集群面临跨节点通信开销（网络延迟>1μs vs 单机<100ns）、分布式训练并行效率衰减（Amdahl 定律约束）以及资源碎片化与能耗问题。大模型预训练的 Scaling Law 曲线持续攀升，SP/TP/EP 总通信数据量相较中小模型提升近百倍，单批次达到数百 GB，但模型跨节点通信带宽增长速率仅为算力增长速率的 1/5，造成通信墙，通信瓶颈愈发突出。

多模态 AI 的算力与通信挑战

多模态模型正从静态生成向实时生成演进，部署形态从单卡单机走向多机并行，对并行带宽的需求达到百 GB/s 级别。此外，不同任务（生成/编辑/理解）对算力及访存的需求差异较大，驱动部署方式走向高算力与高带宽 AI 芯片组合的异构部署。在自动驾驶和具身智能等场景，3DGS（3D Gaussian Splatting，三维高斯泼溅）技术成为主流，其 Tokenization、以及其 3D 识别和 3D 生成应该对 GPU 并行提出了更高要求。

千万级高斯球重建场景下，单轮迭代需压缩至数十毫秒，同步百万级椭球梯度（位置/协方差/透明度等）需要 GPU 间通信达到百 GB/s。梯度数据分散在离散显存空间，需要支持数十个字节级别的细粒度通信，这对小包低时延通信能力提出严峻挑战。

2026 年 1 月，北京智源人工智能研究院推出的多模态大模型“悟界 · Emu”登上 Nature 正刊，成为继 DeepSeek 之后第二个达成此成就的国内大模型团队研究成果，也是中国首篇围绕多模态大模型路线的 Nature 论文。悟界 · Emu3.5 进一步通过大规模长时序视频训练，学习时空与因果关系，展现出随模型与数据规模增长而提升的物理世界建模能力，并观察到多模态能力随规模扩展而涌现的趋势，实现了“预测下一个状态”的范式升级。多模态研究领域也将继续遵循规模扩展定律，对算力的需求将继续不断攀升。

推理算力与智能体的新需求

1. LLM 推理的性能优化需求

如前所述，IDC 的报告中预测，推理算力的占比将在未来几年大幅增长，2028 年达到 73%。MoE 模型推理场景下，以 DeepSeek V3 为例，每 Token 推理需要 58 轮 Dispatch 和 Combine 跨节点动态通信，最小粒度仅为 KB 级，通信量随 Batch Size 线性增加，最大可达百 MB。推理服务需要在令牌生成时间（Time Per Output Token）TPOT 10ms 甚至更低情况下做大 Batch Size，对跨节点网络通信构成巨大挑战。阿里在 SPC 2026 的演讲中指出，超级节点方案在 LLM 推理的解码阶段带来显著的性能提升（30%）。

PD 分离是大模型推理中一种"分工协作"的优化策略。简单来说，就是把生成回复的两个阶段拆开：先让一部分 GPU 负责"快速理解"（Prefill 阶段），另一部分 GPU 专门负责"逐字生成"（Decode 阶段）。这样做的好处是，可以针对两个阶段的不同特点分别配置资源——前者需要强算力，后者需要大内存，从而在保证响应速度和流畅度的前提下，大幅提升整体处理效率。目前，vLLM、Mooncake、Dynamo 等主流推理框架都已支持这一方案，成为大规模部署大模型服务的高效选择。但是，在 Prefill 阶段和 Decode 阶段两个阶段之间需要传输中间数据（KV Cache），需要有高速网络和优化传输，这点超节点相比服务器节点集群有更大的优势。

2. 智能体的通信与缓存需求

智能体的成熟使 AI 从对话进展到工作层面。长期记忆以键值缓存（KV Cache）形式实现，保存推理阶段的上下文信息。

随着序列长度增长，KV Cache 呈线性扩张，需在多个步骤中复用扩展。智能体的出现对 GPU 的压力大增，大量智能体所在机器以远高于人类的调用速度，大量调用 LLM，与传统的人机交互对 LLM Token 的生成速度、成本提出了更高要求，对超节点系统与大模型的极致协同提出了更高要求。

多智能体系统（LLM-MA）将 LLMs 专业化为具有不同能力的智能体，通过互动模拟复杂现实世界环境。智能体间的通信主要采用合作、辩论和竞争三种范式，通信内容通常以文本形式存在。多智能体系统间的通信。比单智能体内部的通信复杂度更高，通信带宽和时延对系统性能产生重大影响，单一节点内的高速网络为 LLM-MA 提供更好选择。

资源池化需求的驱动

云计算使得资源池化成为运营的关键。包括计算、存储、数据处理、高性能计算等都需要高效的资源池化方式。当前的资源池化仍然存在一些挑战有待提升。

当前通用计算业务主要部署在虚拟机或容器，传统虚机 30%-50%内存未被使用，设备无法在节点间池化，整体资源利用率低。虚拟机热迁移时间若达到 100-200ms 量级，客户将感知到业务中断。业务连续性要求 RPO 指标为 0，RTO 指标小于 50ms，以实现虚机热迁移场景业务无损。资源碎片化、内存资源利用率低下、固定资源配置限制灵活性等问题，进一步加剧了资源浪费。传统云平台采用预留分配机制，导致接近半数虚拟机内存未充分利用。

传统分布式存储存在硬件资源离散化、软硬协同能力薄弱等问题。每台服务器的 SSD 容量被独立分配，利用率很低。以 CPU 为中心的架构设计存在根本性缺陷：EC 编解码、CRC 校验、数据跨节点搬移等关键任务抢占主机 CPU 算力，使其成为系统性能瓶颈。分布式存储系统垃圾回收任务中，CPU 将数据从源盘读出并重新写入目的盘，产生多次网络栈和 I/O 栈开销，导致有效 IOPS 下降很多。

数据处理的大数据平台和数据库也存在资源池化挑战。传统大数据平台面临资源配置失衡与浪费、数据交换效率制约整体性能两大瓶颈。跨节点数据混洗（Shuffle）过程中的数据读写与交换效率成为系统性能的主要瓶颈，导致 CPU 平均使用率低。为提升数据库线性度，达到峰值性能最优，数据库需要分层解耦的资源池化架构，需要大规模和低时延 的内存共享。只有实现百 ns 量级的跨节点内存访问时延（低于当前 RDMA 的 1/10），才能匹配数据库架构发展趋势。

近十年 AI 技术快速发展催生 AI4S（AI for Science）新范式，带来高性能计算与智算融合。通信性能挑战、超大规模集群网络拥塞和组网可靠性挑战、存储 I/O 多次内存拷贝问题，共同构成对高性能计算架构的新要求。

新兴 AI 模型的算力需求

以世界模型为首的新兴 AI 模型对算力有更多的需求。世界模型与大语言模型 LLM、多模态模型存在范式层面的根本差异，本质上是从"理解和生成"向"模拟和交互"的跨越。大语言模型的核心任务是在离散符号空间中进行统计建模，通过给定历史文本来预测下一个 token 的概率分布，本质上是一个语言生成器。世界模型则要求构建一个能够生成或预测环境完整状态的概率模型，包括观测（完整的视觉/听觉等多感官数据）、奖励（行为产生的价值反馈）以及隐藏状态（环境的潜在表征）。这种转变使得世界模型从单一的语言生成工具升级为完整的环境仿真器。

世界模型必须满足因果连贯性硬指标，即任何动作产生的后果必须符合物理逻辑。当前的大语言模型和多模态模型主要学习数据中的统计相关性，能够生成看似合理但可能违背物理规律的内容，而世界模型则需要理解牛顿力学的基本规律、掌握因果关系的时间依赖性并建立连贯的物理世界模型。这种因果推理能力要求模型具备更深层次的抽象和推理能力，而不仅仅是模式匹配。同时，世界模型要求环境具有三维层面的持久性。多模态模型主要处理 2D 图像序列，缺乏对三维空间的持久性理解，而世界模型需要建立内在的 3D 表征，能够从不同角度识别同一物体、在被遮挡后恢复其完整状态。3D 几何处理的计算密集性是世界模型算力需求的另一个重要来源。世界模型则需要处理 3D 几何计算，包括 3D 高斯溅射的梯度计算、点云或体素网格的处理、光线追踪和物理模拟等。这些 3D 操作的计算密集度远超 2D 操作，对计算架构的并行能力和内存带宽都提出了更高要求。

在交互性方面，世界模型需要实时响应来自外部的动作指令，形成"动作→状态更新→新观测"的持续交互循环，这与大语言模型和多模态模型主要进行的离线生成形成鲜明对比。这种交互性要求模型具备毫秒级的推理延迟、支持在线学习和持续更新，并能够处理不确定性和随机性，对系统的实时性和动态适应性提出了更高要求。

世界模型的算力需求显著增长，主要源于状态空间规模的指数级扩张。大语言模型的输出空间相对有限，词表大小通常在 5 万到 10 万之间，每次只需预测一个 token。而世界模型需要生成完整的高维状态表征，观测空间可能是百万级的像素向量（如 512×512×3 的 RGB 图像），隐藏状态可能包含位置、速度、物理属性等数百个连续变量，奖励信号虽然通常是标量但需要基于复杂状态计算。这种状态空间的扩张带来了计算复杂度的指数级增长，使得世界模型在单次推理中的计算量远超传统模型。

时间推理的深度依赖是算力需求增长的另一个关键因素。大语言模型的推理主要依赖上下文窗口内的历史信息，通常涉及几千到几万 token。世界模型则需要进行多步预测和长期推理，为了评估一个动作序列的效果，可能需要模拟数十甚至数百步的环境演化，每步预测都涉及复杂的状态转移计算，同时需要维护长期记忆来跟踪环境的持续演化。这种时间依赖性导致计算量随预测深度线性甚至指数增长，对算力的持续消耗提出了极高要求。

多模态融合的复杂度也在世界模型中显著提升。当前多模态模型主要处理静态对齐（如图文配对），而世界模型需要处理动态多模态融合，包括视觉、听觉、触觉等多种感官信息的实时整合、不同模态之间的时空对齐以及多模态状态的一致性维护。这种实时动态融合要求模型具备更复杂的融合机制和更大的参数规模，进一步推高了算力需求。

因果推理的计算开销同样不容忽视。大语言模型的注意力机制主要捕捉统计关联，而世界模型需要进行显式的因果推理，包括构建因果图或结构化模型、进行反事实推理（例如"如果当时做了另一个选择会怎样"）以及维护物理定律的约束满足。这类推理需要结构化的计算（如图神经网络、符号推理），其计算复杂度远超纯神经网络的统计学习，对算力资源构成了巨大压力。

实时交互的延迟约束进一步加剧了算力挑战。大语言模型推理可以容忍数百毫秒甚至更长的延迟（用户打字本身就慢），但世界模型需要满足实时交互约束，智能体决策通常需要 10ms 级的响应时间，在交互过程中需要持续推理而不能离线批处理，同时支持在线学习和参数更新。实时约束要求更高效的并行计算和更低的通信开销，这使得高带宽低延迟互联特性成为必需。

综合来看，LLM 只是让 AI 会说话，世界模型则是构建完整的世界，目前其研究虽然还处在早期，但其算力需求可能是当前大语言模型的 10 到 100 倍，在极端场景下甚至可能达到 1000 倍以上。这种跨越式的算力需求增长，对计算基础设施提出了革命性的要求。世界模型在输出空间、推理深度、状态维度、多模态融合、推理类型和延迟要求等多个维度上都对传统架构构成了挑战，需要更强大的计算密度、更高效的互联技术、更灵活的编程模型以及更精细的资源管理能力。

更大规模的超节点架构，凭借其高密度集成、高速互联、全局编址、资源池化等核心技术特征，能够有效破解传统架构面临的通信墙和内存墙瓶颈。通过提供 1TB/s 以上的互联带宽、低于 500ns 的通信延迟、TB 级统一内存空间以及细粒度的资源池化能力，超节点为大模型训练和推理提供了高效的计算平台。对于世界模型这类对通信时延和带宽要求极高的应用，超节点的高带宽低延迟互联成为必需，而其多 OS 协同架构、统一编程模型和高效的跨组件资源调度，则能够支撑世界模型所需的复杂计算任务。

1.2 超节点架构演进

架构演进路径

超节点的出现不是偶然，而是 AI 计算需求持续增长和架构持续优化的必然结果。其演进路径清晰地体现了从分散到集中、从松耦合到紧耦合的技术趋势。

第一阶段（2020-2022）：基础互联探索期

这一阶段以 PCIe 和 NVLink 早期版本为代表，实现基础的 GPU 互联能力，主要支持 8 卡机配置，集群规模限制在 32 卡以内。这一时期的架构特征是：节点间通信依赖标准网络协议，互联带宽有限，主要满足小规模模型的训练需求。虽然初步实现了多卡并行，但由于缺乏系统级的架构设计，通信效率低下，无法应对大规模分布式训练的需求。

第二阶段（2023-2024）：专用协议崛起期

随着大模型参数规模从百亿级迈向万亿级，专用互联协议开始崛起。NVLink 4.0 等专用互联总线等技术成为主流，集群规模扩展到 64-512 卡。这一阶段的特征是：引入专用于 GPU-GPU 通信的高带宽低延迟协议，显著提升节点内和节点间的通信效率，开始支持中等规模的大模型训练。然而，这一阶段更大规模的集群仍然以 Scale-out 为主，节点间的通信瓶颈尚未完全解决。

第三阶段（2025+）：开放生态与万卡级互联

进入 2025 年后，超节点架构进入百家争鸣阶段。NVLink 5.0、UB 2.0、SUE、OISA、ETH-X、ESUN 等多种互联技术并存，形成异构兼容与开放生态。这一阶段的标志性特征是：支持万卡级互联，实现真正的"单节点化"管理，多 OS 协同成为可能。超节点不再只是硬件集成，而是形成了完整的系统级解决方案，包括统一的编址空间、资源池化能力、以及跨 OS 的协同调度机制。

超节点架构通过破解传统架构局限，实现全局资源抽象，超节点内硬件逻辑一体化，OS 成为算力统一底座：

- **通信架构从“软件定义”到“硬件原生”，应对大模型并行计算的通信刚需：**采用"紧耦合"通信设计，使用芯片直出的高速总线和硬件级内存语义，解决延迟瓶颈、带宽不足和一致性问题，提升 GPU 间的参数交换效率。从 RDMA/PCIe 演进到 NVLink / 灵衢 UB 硬件互联，以及 OS 内核级通信优化。从而支持超节点域内带宽达 Tbps 级，时延纳秒级，支撑千亿参数模型训练效率提升。
- **内存管理从“本地隔离”到“全局池化”,提升内存使用效率：**通过统一内存编址、内存语义访问等技术，使得系统内部可以共享统一内存资源池，减少数据在网络间的过多转换和传递，提升内存使用效率，降低数据处理开销。从“本地隔离”到“全局池化”演进，从单节点内存管理，到异构内存共享（GMEM），再到 全局内存池化。KV Cache 可跨节点动态迁移，支持百万 Token 长上下文，推理并发提升。
- **调度与虚拟化从“节点级”到“超节点级”演进：**从传统 KVM / 容器演进到超节点感知虚拟化，到全局资源编排。从而支持虚拟机热迁移中断时长下降，数据库性能提升，集群稳定运行时间从“天级”到“月级”。

- **降低集群复杂度、平衡性能与成本：**本质是高度集成的小型集群，具备部署简化、运维便捷和便于扩展的优势，应对智算集群规模向万卡/十万卡升级的挑战。通过局部优化降低全局带宽依赖，提升能耗效率优势，用更低成本获得更高的收益。

超节点在计算范式变革中的定位

1. 从 Scale-out 到 Scale-up 的范式转变

传统分布式计算主要依赖 Scale-out 横向扩展，通过增加节点数量来提升整体算力。这种方式的优势是灵活性好、成本低，但缺点是通信开销大、编程复杂度高、资源利用率低。随着大模型对通信时延和带宽要求的不断提高，Scale-out 的局限性愈发明显。

超节点代表了 Scale-up 纵向扩展的新范式。通过在单节点内集成更多的计算资源，并提供高速的互联机制，超节点能够在保持单一系统逻辑的前提下提供大规模计算能力。这种转变不仅是硬件形态的变化，更是编程模型和系统架构的根本性变革。

2. 编程模型的统一化

传统集群采用消息传递（MPI，Message Passing Interface）编程模型，开发者需要显式管理节点间的通信和数据分发，开发复杂度高且容易出错。超节点采用统一内存地址空间的编程模型，开发者可以像访问本地内存一样访问远程资源，Load/Store 语义极大简化了并行编程的复杂度。这种统一化的编程模型是超节点区别于传统集群的关键特征，也是其能够提升开发效率的重要原因。

与传统集群的深度对比

超节点与传统集群在多个维度上存在本质差异，这些差异决定了它们各自适用的场景和边界。

1. 扩展方式

传统集群（Scale-out）：通过横向扩展增加节点数量，将任务分配到多个独立的物理服务器上。这种方式的优势是理论上可以无限扩展，但缺点是随着节点数量增加，通信开销和协调成本呈指数级增长，扩展效率快速衰减。

超节点（Scale-up）：通过纵向扩展在单节点内增加计算资源，在逻辑上保持单一系统。这种方式的优势是通信开销小、资源调度高效，缺点是单节点的扩展上限受限于物理集成密度和散热能力。

2. 互联技术

传统集群：主要采用 InfiniBand 或 RoCE 等标准网络协议，单向带宽通常低于 400Gb/s，网络延迟在微秒量级。这些协议设计用于通用网络通信，没有针对 GPU-GPU 通信进行特殊优化。

超节点：采用 NVLink/UB 等专用互联技术，已经商用的第 5 代 NVlink 双向带宽达到 1.8TB/s，延迟在亚微秒量级。这些协议针对 GPU 通信特性进行了深度优化，在联接上支持高密度的高速连接和低时延交换，功能上支持 Load/Store 语义，能够提供接近片内通信的效率。通过专用互联技术，提供了单一逻辑巨型 AI 计算节点，比如在 NVIDIA Vera Rubin NVL72 超节点系统中，通过 NVLink Switch 将 72 颗 GPU 互联为单一

计算单元，实现 130 TB/s 的总带宽，极大提升了万亿参数模型的训练与推理效率。同时配合集合通信加速功能 SHARP™技术，在网络层直接处理聚合运算，使 All-Reduce 等集合通信性能提升 4 倍，显著减少了 GPU 等待同步的时间。

3. 编程模型

传统集群：采用消息传递编程模型，不同节点间的数据交换需要显式的发送和接收操作。这种模型适合任务级并行，但对于数据级并行和细粒度同步支持不足，编程复杂度高。

超节点：采用统一内存地址空间编程模型，所有计算资源共享同一个地址空间，支持 Load/Store 语义进行数据访问。这种模型极大简化了并行编程，支持更细粒度的并行和更高效的资源共享。

维度	Scale-up（超节点）	Scale-out（传统集群）
扩展哲学	纵向扩展：单机柜内高密度集成	横向扩展：多机松耦合堆叠
互联技术	NVLink 5.0（双向＞1800GB/s，亚μs延迟）	InfiniBand NDR/RoCE v2(400Gb/s，1-10μs延迟）
编程模型	统一内存地址空间（UMA），Load/Store内存语义	消息传递接口（MPI），显式网络通信
一致性机制	硬件级缓存一致性（可选）	软件管理，弱一致性模型
资源粒度	GPU/内存/NVMe细粒度池化（可切分至1%级别）	整机/整卡分配，资源碎片化严重
适用场景	万亿参数以上模型训练、实时推理、内存密集型计算	中小规模模型(或大模型，但需要更多的GPU管理和任务调度）、批处理任务、通用云计算
拓扑特征	收敛比1:1	大规模集群时收敛比通常小于1:1

超节点的适用边界

1. 突破"单点有效算力"阈值的应用场景

并非所有场景都需要超节点，超节点的适用存在明确的"单点有效算力"阈值。当任务的通信占比超过一定比例（通常为 20%-30%）时，超节点的高带宽低延迟互联才能带来明显的性能收益。如果任务的通信占比很低，超节点的优势无法充分发挥，反而可能因为成本和复杂度增加而不经济。

2. 成本效益临界点分析（TCO 对比）

超节点的硬件成本通常高于传统集群，但在长期运行成本上可能具有优势。需要进行 TCO （Total Cost of Ownership）综合评估：包括硬件采购成本、电力成本、运维成本、软件适配成本等多个维度。一般来说，对于持续运行的大模型训练任务，超节点通过提升资源利用率和降低通信开销，可以在短期内收回额外的硬件投资 ；对于间歇性或小规模任务，传统集群可能更具成本优势。

3. 超节点资源虚拟化与云计算的协同关系

超节点和云计算并非替代关系，而是可以形成互补协同。超节点可以作为云服务商提供的高性能计算实例，为特定用户提供极致性能；同时，云计算的弹性资源调度和按需付费模式，可以降低用户使用超节点的门槛。未来的方向是形成"超节点即服务"（Super Pod as a Service）的新模式，让用户能够像使用虚拟机一样便捷地使用超节点资源。

1.3 超节点的技术定义与核心特征

超节点的技术定义

超节点是 AI 时代的新型基础设施，支持高密度算力芯片（CPU、GPU/NPU、DPU、内存、外存中的一种或多种组件）通过 AI 优化高速互联的特定交换机进行互联，从而实现资源池化和平等协同，组建起来的逻辑上的统一内存语义与一体化基础设施构成的机柜级紧耦合算力单元。 超节点凭借超高带宽互联、内存统一编址等技术特征，以及大规模灵活组网、高可靠运行等系统优势，同时支撑大模型高效训练与低时延高并发推理等场景，突破算力、内存、通信三大瓶颈。

包含了几个关键要素：

- 硬件集成：支持多种计算和存储组件的物理集成。
- 资源池化：通过池化技术实现资源的统一调度和高效利用。
- 平等协同：所有计算资源地位平等，协同工作。
- 逻辑单一：对用户和编程者呈现为单一系统。

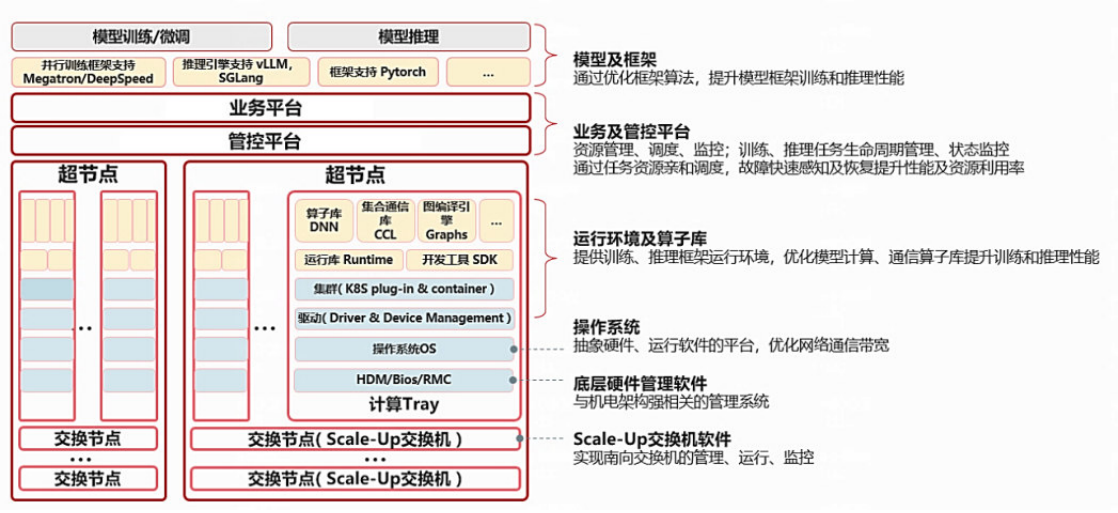


图1 超节点系统架构

相对传统单机的升级

超节点虽然逻辑上表现为单一节点，但在技术实现上相比传统的单机系统做了较多升级和扩展。

1. 硬件层：高密度物理集成

传统单机通常集成 1-8 块 GPU 卡，而超节点的计算板卡物理集成密度远超传统单机，单节点可集成数十块、数百块 GPU/NPU 卡，甚至成千上万块。这种高密度集成对物理设计提出极高要求：包括供电能力、散热能力、信号完整性、机械结构等多个方面。

高密度集成的优势在于缩短物理距离、降低互联延迟、提升通信带宽，但同时也带来了散热和供电的挑战，需要采用液冷、高功率密度供电（>50kW/机柜）等先进技术。

2. 系统层：多 OS 协同架构

传统单机采用单一操作系统进行统一管理，所有硬件资源由同一个 OS 内核调度。超节点则采用多 OS 协同架构，不同的计算组件可运行独立的操作系统，通过管控平台进行协调和管理。

多 OS 协同的优势在于：更高的灵活性、更好的隔离性、更强的容错能力。不同组件可以选择最适合的操作系统版本和配置，某个组件的故障不会导致整个系统崩溃。但也带来了跨 OS 资源调度和状态同步的新挑战，需要设计专门的调度策略和安全隔离机制。

3. 内存层：全局编址与 Load/Store 语义

传统单机的内存空间是独立的，跨节点访问需要通过网络传输。超节点实现全局编址（Global Address Space），将所有计算组件的内存统一编址到同一个地址空间中，支持 Load/Store 语义直接访问远程内存。

这种机制的实现依赖于高速互联协议，部分场景还需要缓存一致性协议的支持。全局编址的优势在于简化编程模型、提升访问效率、降低通信开销，通过预取等手段减少访存等待等，但也需要处理页错误、内存映射、缓存一致性、访存顺序等复杂的技术问题。

4. 扩展性：节点级扩展的光滑过渡

传统单机的扩展受限于物理边界，扩展到更大规模需要增加独立的节点，导致系统逻辑复杂度增加。超节点实现了从"设备内扩展"到"节点级扩展"的光滑过渡，可以在不改变编程模型的前提下，通过增加计算组件平滑扩展算力。

这种扩展性依赖于模块化设计和灵活的拓扑结构，支持计算层、互联层、存储层的独立扩展和灵活组合。

核心技术特征矩阵

超节点的核心技术特征可以从多个维度进行刻画，形成完整的技术特征矩阵。

1. 高密计算

超节点提供极高的计算密度，单节点算力可达千 PFLOPS 以上。高密计算不仅意味着更多的 GPU/NPU 数量，更意味着这些计算资源能够高效协同，避免通信瓶颈制约性能发挥。

高密计算的关键技术包括：紧凑的板卡设计、高效的散热方案、高功率密度的供电系统、以及优化的拓扑结构。

2. 高速互联

高速互联是超节点的核心优势，互联带宽超过 1TB/s，延迟低于 500ns。如此高的带宽和如此低的延迟，使得跨组件通信接近片内通信的效率，为大规模并行计算提供了坚实基础。

高速互联技术栈包括：NVLink 5.0（双向 1800GB/s）、CXL 3.0（内存语义扩展）等。这些技术共同构成了超节点的互联基础设施。

3. 全局编址

全局编址提供 TB 级统一内存空间，将所有计算组件的内存统一到一个地址空间中，支持 Load/Store 语义直接访问。全局编址的优势在于简化编程模型、提升访问效率、降低通信开销，是超节点区别于传统集群的关键特征。

4. 资源池化统一管理

资源池化实现计算/存储/内存的解耦，将不同类型的资源统一管理，按需分配给任务。资源池化的关键在于细粒度的资源管理、动态调度策略、以及跨组件的状态同步。

资源池化提升了资源利用率和系统弹性，但也带来了更复杂的资源调度和状态管理需求，需要专门的调度算法和管理软件。

技术成熟度与生态建设

超节点作为新兴技术，其技术成熟度和生态建设情况对其推广应用至关重要。

1. 技术成熟度与场景需求匹配

超节点的各项技术特性需要与具体的应用场景需求相匹配。对于通信密集型的大模型训练推理任务，超节点的优势最为明显；对于万亿规模的大模型训练和推理，需要大规模的并行技术，突破算力、内存、通信墙，也是超节点的优势所在；对于计算较离散且通信需求较低的任务，超节点的优势可能无法充分发挥。因此，在应用超节点技术时，需要仔细评估场景需求的匹配度，避免技术能力的浪费或不足。

2. 建立开放互联标准

超节点的长期发展依赖于开放互联标准的建立。目前市场上存在多种互联技术标准（NVLink、CXL、UB、SUE 等），各有优势但也存在兼容性问题。建立统一的开放互联标准，促进不同厂商设备的互操作性，对于构建健康的超节点生态至关重要。

开放互联标准的建立需要产业界的共同努力，包括技术标准组织、硬件厂商、软件开发商和用户等多个利益相关方。

1.4 核心指标体系

超节点的性能和能力可以通过多个维度的指标进行评估，形成完整的指标体系。

算力密度

算力密度衡量单位空间或单位功耗提供的算力能力，是超节点性能的重要指标。

单位体积算力（PFLOPS/m^3）：表示在单位体积内提供的算力，反映硬件集成的紧凑程度。高算力密度意味着可以在有限的数据中心空间内部署更多的计算资源，这对于寸土寸金的数据中心尤为重要。超节点通过高密度集成和优化的物理设计，可以显著提升单位体积算力。

单位功耗算力（PFLOPS/kW）：表示在单位功耗下提供的算力，反映能源利用效率。在碳中和和能源成本上升的背景下，能效比成为评估计算基础设施的重要指标。超节点通过优化供电和散热系统，降低单位算力的功耗，提升能效比。

互联带宽

互联带宽衡量不同计算组件之间的数据传输能力，是超节点性能的关键瓶颈。

单向/双向带宽（GB/s vs Gbps）：表示在单位时间内可以传输的数据量。带宽越高，数据传输越快，对通信密集型任务的性能提升越明显。超节点的互联带宽通常超过 1TB/s，远超传统集群的 400Gb/s 水平。

带宽密度（GB/s/GPU）：表示平均每块 GPU/NPU 可用的带宽。带宽密度越高，每块卡的通信能力越强，系统的并行扩展性越好。超节点通过优化拓扑结构和互联协议，提升每块卡的可用带宽。

延迟指标

延迟衡量数据传输的时间开销，对于实时性要求高的任务至关重要。

P2P 延迟（GPU 间）：表示两块 GPU 之间进行点对点通信的延迟。低延迟能够更快速地完成同步和协作，对细粒度并行任务尤为重要。超节点的 P2P 延迟通常低于 500ns，远低于传统集群的微秒级延迟。

内存访问延迟（NUMA 距离）：表示访问远程内存的延迟，反映全局编址的效率。内存访问延迟越低，跨组件内存访问的效率越高，对需要频繁共享内存的任务性能提升越明显。

扩展能力

扩展能力衡量系统能够支持的计算规模和扩展效率。

纵向扩展极限（单机卡数）：表示单节点能够集成的最大 GPU/NPU 卡数。纵向扩展极限越高，单节点的算力越强，能够处理的任务规模越大。

横向扩展效率（多超节点互联）：表示多个超节点互联时性能扩展的效率。横向扩展效率越高，系统能够支持的规模越大，能够处理的任务越复杂。超节点通过优化的互联协议和拓扑结构，实现高效的多节点互联。

能效比

能效比衡量能源利用效率，对运营成本和环境影响有重要影响。

PUE（电能利用效率）：数据中心总能耗与 IT 设备能耗的比值，PUE 越低，供电、散热等基础设施的能耗占比越小，整体能效越高。超节点通过优化散热和供电系统，降低 PUE 值。

算力功耗比（每瓦性能）：单位功耗提供的算力，反映硬件的能效特性。算力功耗比越高，每度电能够完成的计算量越大，运营成本越低。

生态兼容性

生态兼容性衡量系统能够支持的软件生态和迁移成本。

软件栈支持：主流深度学习框架和科学计算库的支持程度。良好的软件栈支持意味着开发者可以使用熟悉的工具和框架，降低学习和使用成本。

迁移成本评估：从传统集群迁移到超节点需要投入的成本，包括代码改写、软件适配、人员培训等。迁移成本越低，超节点的推广越容易。

2 核心应用场景

2.1 超大规模 AI 模型训练

随着模型进入千亿至万亿参数、万卡级集群时代，传统分布式架构面临跨节点通信带宽不足、时延过高导致算力利用率长期偏低，单卡显存无法承载超大模型等诸多问题，大模型预训练是高算力密集、高通信密集、高内存密集等特点，对基础设施提出了挑战。

通信瓶颈：跨节点数据同步的效率挑战

在万亿参数模型训练中，梯度同步和参数交换的数据量呈指数级增长，大模型训练依赖 All-Reduce、All-to-All 等集体通信操作，千亿参数模型单次梯度同步会产生 TB 级数据量，传统基于 PCIe + 以太网的架构，其带宽（通常<400Gbps）远低于计算单元的吞吐能力，导致 GPU 大量时间处于等待数据的空转状态，有效算力利用率 MFU 往往难以突破 50%，尤其在 MoE 等稀疏模型训练中，负载动态变化加剧了通信不均衡问题，进一步放大了带宽压力。

内存瓶颈：内存墙制约模型规模与训练灵活性

大模型训练对内存容量的需求呈指数级增长，而单卡显存容量如 H100 HBM3 仅 80GB，模型参数、优化器状态和激活值所需的显存容量远超单卡甚至单机上限，单 GPU 算力不足、内存不足，大模型必须通过大规模并行计算实现，增加了跨 GPU 的通信负载，同时跨卡访问显存需经过复杂的拷贝流程，速度极慢，数据搬运延迟成为训练效率的核心瓶颈。在训练过程中，频繁的权重读取、梯度回传和激活值交换导致数据搬运时间远超实际计算时间，导致 GPU 需要等待数据从显存或其他节点传输到位。

针对上述主要挑战，超节点通过硬件架构革新与软件协同优化，实现超大规模模型高效训练：

极致通信性能：超节点采用高速互联架构，通过 NVLink 或 HCCS 等私有互联技术，节点内聚合带宽达到 TB/s 级别，例如 NVIDIA NVL72 达 130TB/s，华为昇腾超节点达 PB/s 级聚合带宽；通信延迟从微秒级 μs 降低至纳秒级 ns，实现了内存级的远程访问体验；在万亿参数 MoE 模型训练中，模型算力利用率（MFU）从传统的 40%-50%提升至 60%-65%以上，同等规模模型训练时间缩短 30%-40%，加速了模型迭代速度。

内存池化突破：超节点构建了以全局共享异构内存池为核心的内存访问体系，基于 NVLink/HCCS 等专用低延迟高速互联总线，为节点内所有并行计算单元提供连续、统一编址的 PB 级全局虚拟地址空间，各计算单元通过 Load/Store 原生指令实现全局内存透明访问，消除跨节点数据拷贝开销，实现跨 GPU 的远程内存直接访问与统一调度。使得万亿甚至十万亿参数的稠密模型或稀疏 MoE 模型可以在更少的节点数上运行，减少了模型切分带来的通信开销，优化了显存利用率，显存碎片率降低 90%，支持更大的 Batch Size，提升训练收敛速度。

整体应用效果来看，超节点全面突破传统集群的技术瓶颈，将万亿参数模型训练周期从数月缩短至数周，通信损耗从 30%-40%降至 10%以内，算力利用率从 60%-70%提升至 90%以上，模型微调效率提升 80%、开发适配成本降低 90%；可高效支撑 LLM、MoE、多模态等各类超大规模模型的训练与迭代，同时为政务决策、应急预测等行业专属超大规模模型，提供低延迟、高效、高可靠的核心算力支撑，成为超大规模模型训练的优选算力底座。

2.2 高性能计算（科学模拟、气象预测、药物研发）

高性能计算在科学模拟、气象预测、药物研发三大场景中，核心需求是大规模并行计算、海量数据实时处理、高精度模拟迭代，随着人工智能发展，带来了 AI+HPC 超智融合的需求，对超节点也提出了更多挑战：

- **通信带宽与延迟的极致要求：**随着模型参数量突破万亿级及科学计算网格的细化，节点间数据交换量呈指数级增长。传统基于 MPI 的 Scale-Out 架构中，受限于 Infiniband / 以太网跨节点互联的带宽与时延瓶颈，已成为算力释放的核心短板。
- **海量数据处理与访问延迟瓶颈：**三大场景均涉及 PB 级甚至 EB 级海量数据（如气象卫星观测数据、分子结构数据、多物理场模拟中间数据），传统 HPC 集群采用计算与存储分离架构，数据需在存储节点与计算节点间频繁迁移，受限于互联带宽，数据传输延迟达毫秒级，同时，模拟过程中需高频访问海量中间数据，传统分布式内存无统一编址，数据访问效率低。
- **算力密度瓶颈：**现代高性能计算需要处理从天气预报、流体力学到基因测序、材料模拟等海量任务。这要求系统在有限的物理空间内集成尽可能多的计算核心，以提供巨大的并行计算能力，而单纯堆砌核心会导致通信、供电和散热的物理极限。

- **精度适配挑战：**传统高性能计算的核心任务要求双精度（FP64），以保证结果的数值稳定性和可复现性。而 AI 负载则大量使用半精度（FP16）甚至 8 位浮点（FP8）来提升速度和能效。一套高性能计算系统需要同时为这两类任务提供最优的精度支持。

针对上述 HPC 与智算融合场景的核心挑战，超节点的应用效果如下：

通信性能数量级跃升，破解带宽时延瓶颈：超节点通过专用高速互联总线，超高速正交架构或全光互联提供 TB 级带宽，打破通信瓶颈，实现百纳秒级端到端时延、TB/s 级双向带宽，较传统超算集群的 Infiniband 集群时延降低 90%、带宽提升 10–20 倍。

全局内存池化重构访存体系，破解数据访问瓶颈：超节点构建 PB 级统一编址全局异构内存池，通过硬件级地址映射，实现跨设备内存的原生指令级透明访问，不规则随机访存延迟降低 80% 以上，降低数据搬运开销降低，支撑 PB 级海量数据集的全内存级处理，减少频繁磁盘 IO 带来的性能损耗。

高密度异构算力集成，突破物理极限约束：超节点采用单机柜高密度异构算力集成设计，单柜可整合近百个 AI 芯片（GPU/NPU）与海量 CPU 核心，异构算力密度较传统风冷 HPC 集群提升 8–10 倍；通过算力池化，实现 CPU 与 GPU/NPU 的无缝协同，提升异构算力协同效率。

异构算力灵活调度，破解精度适配难题：超节点通过统一的 Scale up 互联架构（如 UB-Mesh），将支持不同精度的 CPU、GPU、NPU 等异构芯片池化。系统可以按需调度，让高精度任务跑在 FP64 强的芯片上，低精度任务跑在 FP16/FP8 强的芯片上；让一套系统既能高效支持 AI 训练，又能精确完成科学模拟，避免了为不同任务建设多套系统的浪费。

整体上，超节点通过专用高速互联、全局异构内存池化、高密度算力、异构算力智能调度的架构级革新，相比传统超算通信性能、访存效率、算力密度的数量级跃升，同时兼顾 HPC 双精度高精度模拟与 AI 低精度训练的双重需求，将成为智算融合时代超算基础设施的核心底座。

2.3 关键行业 AI 基础设施（政务、金融、能源自主可控算力集群）

政务、金融、能源等关键行业的 AI 基础设施，这类行业往往涉及敏感数据与核心业务流程，不仅需满足大规模 AI 训练、推理及数据处理的高算力需求，更需具备自主可控的核心特性，以保障核心业务的合规性和数据安全性，防范核心技术受限风险与业务中断带来的重大损失，对软硬件自主化、安全防护能力及运行稳定性的要求远高于普通场景。

自主可控是这些关键行业 AI 基础设施共同面临的根本性挑战。超节点技术的核心价值之一，就在于它提供了一个完全自主、开源开放的技术体系。开放的超节点参考架构，目标是构建一个不受单一厂商制约的、开放的智算生态，这使政务、金融、能源等行业能够基于国产技术，打造从芯片到系统、从硬件到软件的全栈自主可控的 AI 基础设施。

政务领域，政务数据涉及公民隐私与国家机密，要求算力设施必须自主可控，高度重视数据安全风险，同时在业务高峰期能够高效处理海量市民的并发请求，对推理系统的吞吐能力和响应时延要求高；超节点可基于鲲鹏、昇腾、海光等国产芯片，结合国产操作系统、与自研高速互联协议，构建全栈自主可控的算力集群；超节点在模型训练及微调中的性能表现比传统 AI 服务器产品提升 2.5–3.5 倍，可以作为政务智能化升级的强劲算力引擎，总体看，超节点采用国产芯片和全栈技术，满足国家信息安全和自主可控的要求的同时确保了高并发下业务系统的流畅运行，让市民享受到更快捷的服务。

金融领域作为国家经济命脉，面临核心交易系统自主化率达 80% 的刚性政策要求，同时叠加强监管合规与业务连续性的极致约束，针对金融场景的核心痛点，超节点深度整合国产异构算力资源，依托自主研发的高速互

联技术与全局共享内存池，实现国产异构算力的无缝协同，解决国产芯片适配性差、算力碎片化问题；算力调度与响应延迟方面，超节点构建无阻塞全对等高速互联架构，精准适配高频交易、实时风控的低延迟需求；通过整机柜冗余设计与硬件级故障自愈机制，将 MTTR 从小时级缩短至分钟级，核心系统可用性达 99.999%，满足金融核心业务连续性要求；通过芯片级软硬件协同优化，国产芯片算力利用率提升至 85% 以上，千亿参数金融风控大模型训练效率提升 3 倍，实时交易反欺诈推理时延从 50ms 缩短至 5ms，欺诈识别率提升 40%

能源行业，传统基于通用服务器堆叠的算力集群，在面对新型电力系统、智能勘探、流程优化等复杂场景时，遭遇性能、效率、安全与可靠性的多重瓶颈；在电网调度领域，超节点提供的超高带宽和极低时延的互联能力，使得电网调度中心能以 15 毫秒内的极快速度完成全网状态感知，结合自主研发的电力智能调度引擎，实现电网负荷实时分析、故障精准预判、调度方案快速生成的技术突破。国家电网相关研究院的测试显示，基于超节点架构的 AI 服务器，在电网最优潮流计算任务上，性能可达传统服务器的 3–10 倍，有效支撑了“沙戈荒”大基地等新能源外送场景下的动态稳定控制。

2.4 混合负载场景（训练 + 推理协同、异构芯片集群）

混合负载场景核心需求是实现训练与推理协同调度、异构芯片集群高效适配，兼顾算力利用率、任务响应速度与资源集约化，支撑 AI 模型快速迭代与实时服务落地。当前混合负载场景普遍面临负载适配、异构协同、资源复用等核心瓶颈，具体挑战如下：

- **训练与推理负载适配瓶颈。**AI 训练与实时推理的负载特征差异极大，异构芯片的算力特性也各不相同，传统调度机制无法实现资源的精细化匹配与高效利用，无法根据负载类型、精度需求实现算力精准匹配，导致异构芯片算力利用率比较低。
- **异构芯片集群协同瓶颈。**混合负载场景普遍采用 CPU/GPU/NPU 异构芯片集群，各类芯片算力优势不同，GPU/NPU 擅长张量计算，CPU 擅长通用调度。传统集群协同协议不完善，负载分配不均，无法充分发挥各类芯片算力优势，且异构通信损耗大，拖累整体效率。
- **异构内存协同与数据交互的效率瓶颈。**超节点往往融合通用 CPU、专用 AI 加速器及存算一体单元。不同架构间的内存一致性协议（Cache Coherence）难以统一，数据在 HBM、DDR 及 NVMe 之间的频繁搬运造成严重的内存墙效应。
- **训练与推理资源复用瓶颈。**训练过程中产生的中间数据、模型参数，与推理过程中所需的模型权重、输入数据无法高效复用，需重复传输、存储，不仅增加通信与存储开销，还延长任务响应时间，影响协同效率。

针对上述核心挑战，超节点通过架构级革新实现全维度突破，核心应用效果如下：

针对训练与推理负载适配瓶颈，超节点搭建智能负载感知调度引擎，依托硬件级任务监测模块与实时数据采集单元，实时捕捉训练、推理任务的算力消耗、带宽占用、时延需求等核心指标，结合业务 SLA 优先级算法，动态区分任务优先级（推理任务优先保障低时延，训练任务采用弹性优先级），支持负载动态感知与资源自适应调整，空闲时段推理并发量低、算力冗余时，自动将冗余算力调度至训练任务，提升训练效率、缩短训练周期；高峰时段推理并发激增时，优先为推理任务分配足额算力与带宽资源，锁定推理时延稳定性，从技术层面彻底避免负载间资源抢占与闲置浪费，实现混合负载的高效协同。

超节点优化异构算力协同协议，深度优化异构算力协同协议，先后出现 NVLink、UB、HSL、PCIe/CXL 等主流异构芯片通信标准，实现 x86/ARM 架构 CPU、GPU、NPU 等多品类、多架构算力芯片的统一纳管与无缝适配，解决传统异构集群协议不互通、适配门槛高、协同兼容性差的行业痛点。

训练与推理资源复用瓶颈，超节点构建全局共享内存池，实现训练中间数据、模型参数与推理所需资源的统一存储、高效复用，无需重复传输与存储；构建 PB 级全局异构内存池，实现 CPU/GPU/NPU 等异构内存统一编址与统筹调度，实现异构芯片间内存的原生指令级透明访问，内存利用率提升至 75%以上，破解训练内存不足与推理内存闲置矛盾。依托内存池实现训练–推理数据实时同步，模型迭代效率提升 60%以上，推理精度提升 30%

超节点核心技术架构

1 硬件架构设计

随着大模型参数规模迈入万亿级（Trillion-parameter），智算基础设施的架构设计逻辑，正发生自通用计算时代以来最为深刻的系统性重构。计算单元的互联架构已完成核心范式转移，由传统以中央处理器（CPU，Central Processing Unit）为核心的互联模式，转变为以图形处理器（GPU，Graphics Processing Unit）为核心的新型互联架构。

在该新型计算架构中，GPU 作为核心算力载体，其功能定位已超越传统服务器单一硬件组件的物理范畴。超节点（Super Pod）组网方案是突破高性能计算领域内存墙与通信墙核心技术瓶颈的关键手段。该方案依托超高带宽、超低延迟的纵向扩展（Scale-Up，南向组网）互联技术，将数十至数百个 GPU 计算模组在逻辑层面整合为统一的高性能计算单元，最终实现算力资源的深度池化与一体化调度。

本章将系统剖析超节点硬件技术架构：研究维度覆盖宏观整机柜级互联形态、超节点内部硬件架构设计；技术层面将对比封闭的 NVLink 协议与开放式标准化互联协议的技术路径；同时重点分析超高功率密度场景下，超节点在高速互联、供电系统、散热设计三大方向的基础设施工程化挑战与核心设计要求。

1.1 超节点硬件架构

AI（Artificial Intelligence）超节点是高性能计算与 AI 大模型基础设施的核心架构形态。在组网架构层面，超节点互联架构与传统 CPU 集群组网存在本质差异：传统 CPU 集群以通用计算为核心设计目标，采用以太网或 InfiniBand（IB）网络为主的横向扩展（Scale-Out）架构，聚焦通用算力的分布式聚合与多业务通用调度，计算单元间依赖 CPU 调度、操作系统内核完成数据包级数据交换。而 AI 超节点以突破大模型算力瓶颈为核心目标，其内部 GPU 集群采用纵向扩展（Scale-Up）南向互联架构，这也是超节点最核心的组网特征；超节点通过基于内存语义（Memory Semantics）的极高带宽、极低时延内部互联协议，将数十甚至数百个 GPU 模组在逻辑上整合为一个统一系统集群 [1,2]。其内部业务带宽密度通常是传统通用计算以太网组网方案的 10 倍以上，核心定位是专为解决大模型训练（Training）和推理（Inference）场景中内存墙与通信墙技术瓶颈而构建的高密度算力集群。

超节点作为 AI 时代新的算力范式，在技术实现层面通常具备以下特征：

- **逻辑单一性与资源池化：**采用全互联硬件架构，依托高带宽内存介质实现算力资源池化，保证 GPU 间通过高效互联完成协同计算。超节点内全部 GPU 在逻辑上构成单一计算资源池，通过硬件级统一地址映射技术，任意计算单元可直接通过 Load/Store 指令访问集群内其他单元的高带宽内存（HBM, High Bandwidth Memory），无需 CPU 执行显式数据搬运操作，也无需操作系统内核参与，实现跨算力芯片的零拷贝通信。
- **计算与通信解耦：**为适配大模型多样化并行策略，包括数据并行（Data Parallelism, DP）、流水线并行（Pipeline Parallelism, PP）、张量并行（Tensor Parallelism, TP）、专家并行（Expert Parallelism, EP），超节点通常采用计算单元与交换单元分离的解耦设计。该模块化架构可根据实际业务需求，灵活平衡算力规模与通信带宽的配比关系。
- **更大规模分级组网：**超节点内部依托南向 Scale-Up 架构实现 GPU 高密度整合；而在超节点与超节点之间，可通过以太网或 IB 网络实现横向 Scale Out 扩展，支撑更大规模的智算集群部署。

1.2 硬件互联架构

超节点从定义上，属于逻辑上的集群系统，业界超节点的具体形态也非完全一致，但其演进上呈现出两大主流趋势。

整机柜超节点（Rack as a Computer）

整机柜超节点形态是当前高性能 AI 算力的主流载体，这类系统通过定制化的铜缆背板（Cable Tray）实现机柜内的算力全互联。

作为面向万亿级参数模型训推场景的超节点产品，H3C UniPoD S80000 实现了更高性能、更高密度、更高效率的三重进化。面对客户追求极致性能的需求，S80000 以网强算，全面释放算力矩阵动能，柜内卡间全互联通信。密度方面，S80000 单柜支持部署 64 卡，采用液冷方式散热，整柜功率可支持到 120KW，同时兼容下一代高性能 AI 加速模组设计。此外，三总线全盲插、全面的漏液检测等方面的设计，能极大简化运维流程、提升能效产出。

H3C UniPoD S80000 系列通过 2 层无阻塞交换架构，支持 256 卡集群组网。未来，新华三还将基于超节点整机柜形态推出 1024 卡超节点集群。

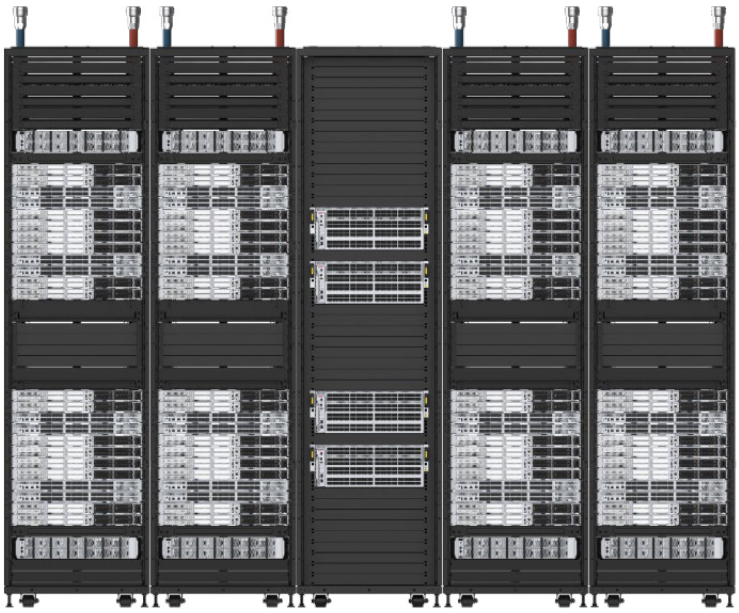


图2 新华三 256 卡整机柜超节点集群

整机柜超节点 Rack as a Computer 的设计，创造性地解决了单机 GPU 的内存容量与通信带宽瓶颈，是当前追求极致性能最成熟的规模化路径。

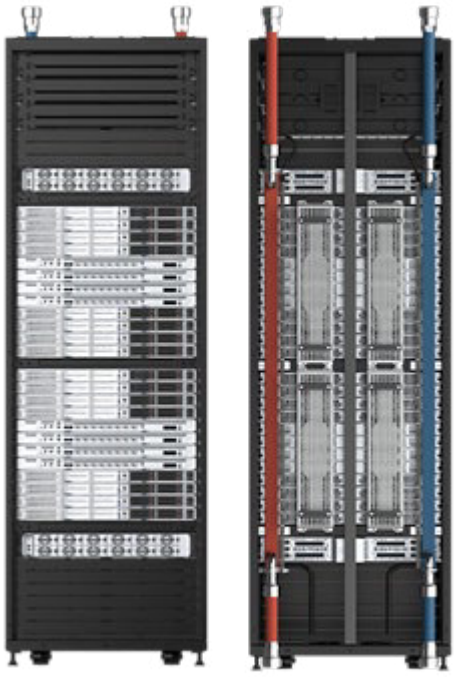


图3 新华三 S80000 整机柜超节点

如图所示，整机柜超节点系统通常由计算节点（Compute Tray）、交换节点（Switch Tray）、互联铜缆（Cable Tray）组件、电源插框（Powershelf）供电系统和液冷管路（Manifold）组成。

正视图：通常采用对称式布局。计算节点围绕交换节点上下分布，以确保所有计算节点到交换节点的走线长度保持较高一致性，保证对时延敏感的信号时序统一。顶部或底部配置高密度的 Powershelf 集中供电，支持 N+N 或 N+1 冗余及动态均流。

后视图：核心是盲插（Blind-mate）接口区域。中央为直流汇流排（BusBar）为柜内设备提供输入电源，两侧分布高密 Cable Tray 模组及 Manifold 接口，支持计算/交换节点统一插拔维护。

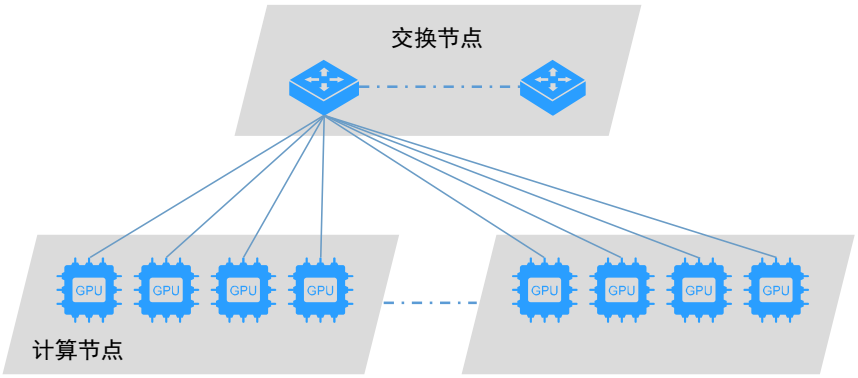


图4 整机柜超节点组网示意图

业界如 NVIDIA 基于 BlackWell 和 Rubin 系列的 NVL72 系统架构、AMD 基于 MI4XX 系列的 Helios 整机柜、阿里云基于 UALink 推出的 ALS 整机柜架构、昆仑芯天池架构、字节跳动大禹架构等都属于整机柜超节点形态。

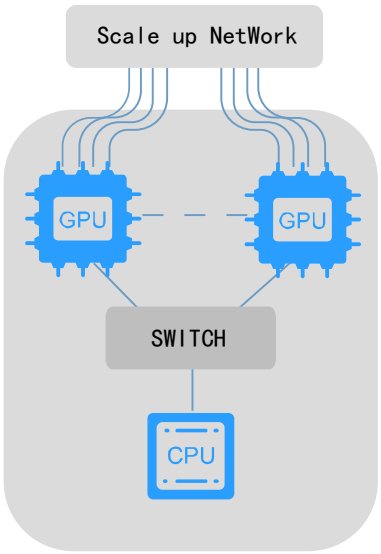


图5 计算节点示意图

如图所示，为业界通用整机柜超节点的计算节点板，物理形态上通过 Cable Tray、正交背板或正交网板直连方案实现与柜内交换节点互联，形成更大规模的整机柜高带宽域（HBD, High Bandwidth Domain）。未来随着 Serdes 速率进一步突破 224GT/s，光互联也可能成为超节点集群扩展的形态选项。

整机多框超节点（Disaggregated Chassis）

整机多框超节点形态，通常由整机算力节点、交换机节点、互联传输介质（铜缆或光缆）构成。该架构在机框内部实现算力资源的高密度集成部署，通过外部交换单元完成跨机框的远程组网；亦可采用机内交换汇聚后二级扇出的拓扑方式，实现分级远程扩展，从而为大规模纵向扩展（Scale-Up）集群的建设，提供了高弹性的物理部署方案。

图6　新华三 F80000 超节点

依托国产算力平台，H3C UniPoD F80000 采用创新的光互联技术拉远集群组网，突破整机内高速互联瓶颈，实现了跨机箱/跨机柜的超节点集群互联，实现模型训练性能提升 35% 以上。基于灵活开放的产品理念，H3C UniPoD F80000 支持基于不同形态的 AI 整机服务器及 AI 加速模组灵活构建超节点产品，支持灵活按需部署。

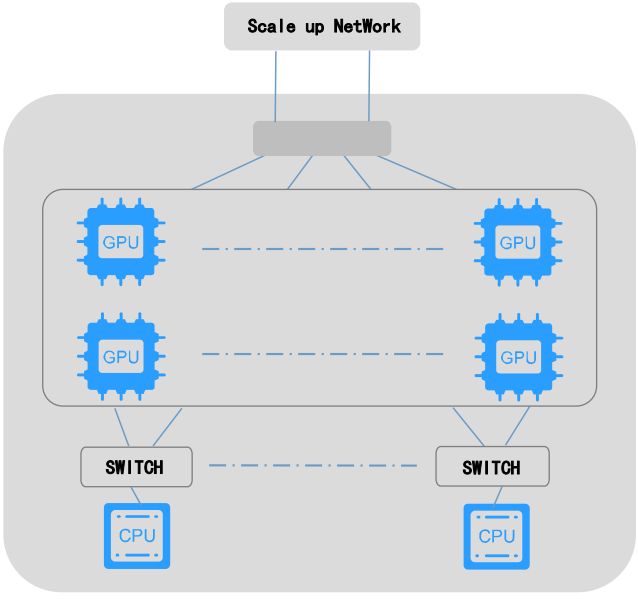


图7 超节点机内示意图

如图所示为业界整机多框超节点常见的机内计算模块，整机系统内通过 Scale Up 交换芯片的高速信号全部扇出，再通过外部交换机实现拉远组网，完成更大规模 Scale Up 集群构建。

华为基于 Ascend 910C 系列 CM384 正交架构、NVIDIA Rubin Ultra 正交背板架构等超节点，在整机系统中机内通过交换汇聚后再扇出，也是整机多框超节点组网的另一种物理形态选择。

未来超节点架构的发展演进

在具体的底层硬件互联层面，超节点的发展正面临着严峻的物理制约。随着 SerDes（串行器/解串器）传输速率逐步向 224GT/s 的超高标准演进，传统的铜缆传输介质在信号完整性保障与整体功耗控制方面的挑战，已经无限逼近其物理极限 。 在此背景下，为了突破算力传输的物理瓶颈，未来数据中心及超节点集群内的光互联技术将完成从“可选项”向“必选项”的根本性转变 [3]。

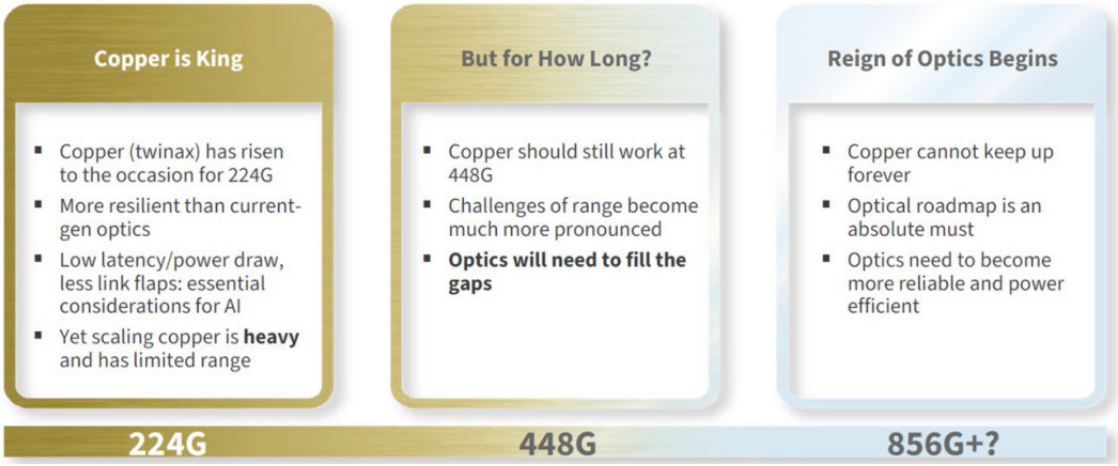


图8 未来高速 Serdes 的演进以及传输介质的变革 [4]

除了互联介质的转变，超节点的物理形态演进同样深度受制于供电与散热这两项核心工程约束。传统的高密度集中式算力单体，往往受困于这两大物理极限而难以实现规模的无限扩展。引入先进的光互联技术，不仅能够在物理层面实现大规模组网的服务器拉远部署，更具战略意义的是，它能够有效地将高度集中的供电与散热工程设计约束进行物理分离。这种硬件形态的解构与重组，为打破传统物理单体的空间限制提供了切实可行的路径，是支撑未来构建更大规模、更高效率算力资源池化的核心技术手段。

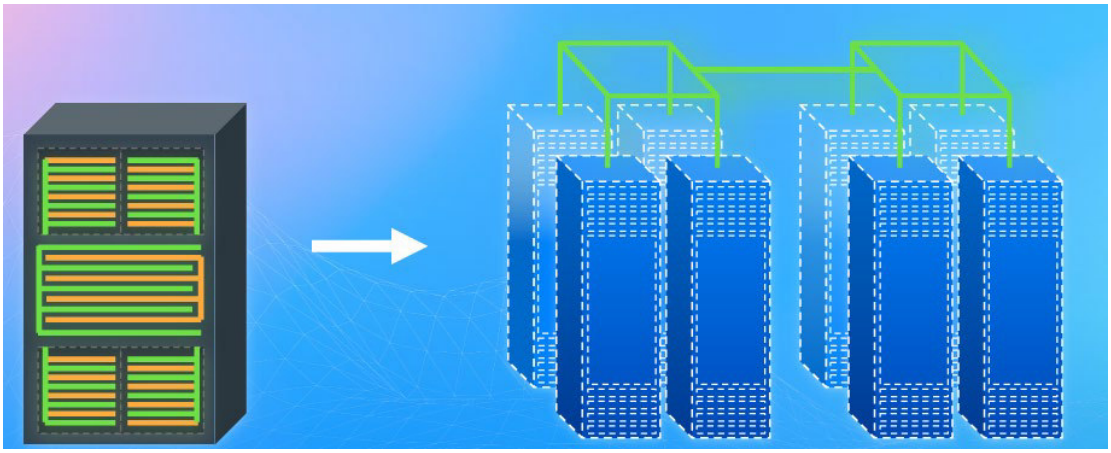


图9 光互联组网解耦架构演进 [5]

超节点技术的持续演进过程，在很大程度上映射了通用计算集群的发展规律。在这一宏观演进周期中，通用服务器的组网模式呈现出明显的“集中”与“分布”交替演进的特征。这种架构形态的更迭，其本质并非盲目的技术堆叠，而是在不同的技术历史阶段下，系统设计在性能释放、建设成本、系统扩展性以及部署复杂度等多维度指标之间所进行的综合权衡（Tradeoff）。因此，未来超节点技术的发展必然不是依赖于某一项单一技术的孤立突破，而是需要依托多项技术的深度融合，从而实现算力集群向着体系化、可持续的方向进行迭代。

1.3 计算节点硬件设计

计算节点作为超节点系统的核心算力单元，其架构已从通用计算转向极致的并行加速与异构协同。CPU 的角色正经历从控制者到协同者的转变。

GPU 算力模组

在深度学习爆发初期，数据中心主要依赖标准的 PCIe 插卡（Add-In Card, AIC）来扩展 GPU 算力。然而，随着模型训练从单纯的计算密集型任务转向通信密集型任务（如 All-Reduce 集合通信），金手指插卡形态暴露出了难以克服的物理缺陷。

首先，AIC 的接入方式和 PCB 边缘连接器（Edge Connector）限制了高速接口的扩展。传统服务器中 GPU 依赖 PCIe P2P 特性的交互，也严重影响系统通信效率。其次，AIC 形态对散热器的体积限制使得单卡功率很难突破 400W-500W 的物理极限，制约了算力密度的进一步提升。

1. OAI 生态系统：标准化之路

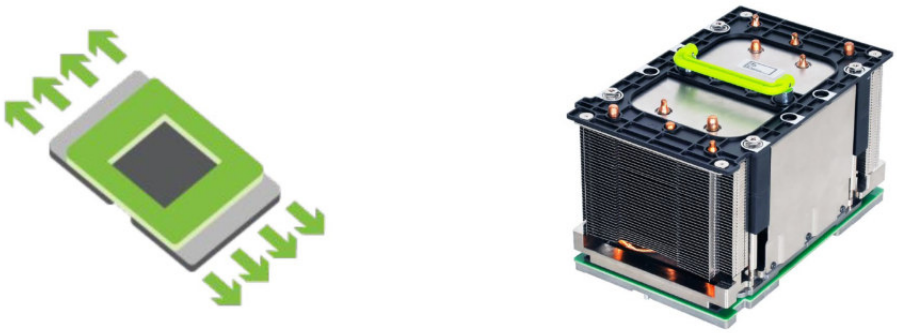


图10 OAM 模组示意图

开放计算项目（OCP）推出的开放加速器基础设施（OAI）[6]不仅仅是定义了一个算力模组形态，而是构建了一个完整的硬件生态层级，确保了不同供应商芯片的物理互操作性：

- **OAM（OCP Accelerator Module）：** 包含 ASIC 芯片、HBM 显存及 VRM 供电的核心算力单元[7]。
- **UBB（Universal Baseboard）：** 通用基板，作为承载 8 个 OAM 模组的物理平台，定义了模组间的互联拓扑（如全互联或混合立方网格）。
- **HIB（Host Interface Board）：** 主机接口板，负责将 UBB 与主机 CPU 连接，实现了计算模组与控制节点的解耦。

- **SCM (Secure Control Module)**：安全控制模组，独立处理系统管理、固件验证和热遥测，将带外管理（OOB）与数据平面分离。

这种分层设计使得超节点集群能够独立升级 OAM 算力单元或 UBB 基板，而无需更换整个机架基础设施，极大地降低了 TCO（Total Cost of Ownership）成本。

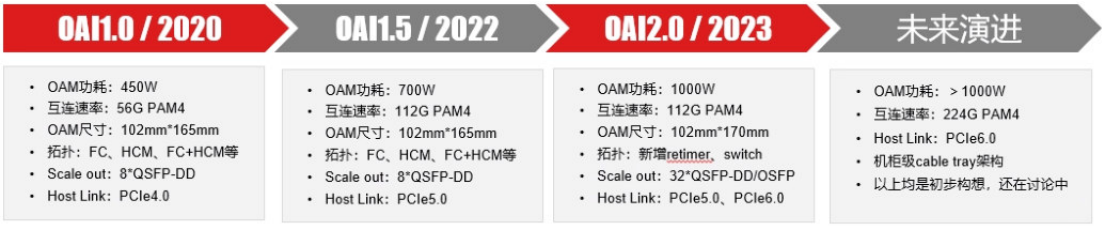


图11 OAI 标准的演进

2. OAM 模组的技术演进：从 OAI 1.0 到 OAI 2.0

OAM 规范的演进史，本质上是对抗物理极限（功耗、信号完整性、散热）的工程实现。参考 OAI 规范演进路线图，我们可以清晰地看到技术指标的指数级跃迁。

3. OAI 1.0 (2020)：确立基准

OAI 1.0 规范奠定了 OAM 的物理形态基础，主要针对当时主流的 AI 加速器需求。

- **物理尺寸与互联**：定义了 102mm x 165mm 的标准尺寸，采用 Mirror Mezz 连接器。这种扣卡式设计相比金手指连接器提供了更高的引脚密度和更好的信号完整性。
- **功率与散热**：设定了 450W 的功耗基准。在这一阶段，风冷仍然是主流解决方案，同时规范中预留了液冷的支持规格。
- **互联速率**：互联速率支持 56G PAM4，对应单通道 56Gbps。这在当时足以支持 PCIe Gen4 和早期的专用互联协议。
- **拓扑结构**：UBB 设计支持全互联（Fully Connected, FC）和混合立方网格（Hybrid Cube Mesh, HCM），主要针对 8 卡系统进行优化，机间扩展能力通过 8 个 QSFP-DD 端口实现。

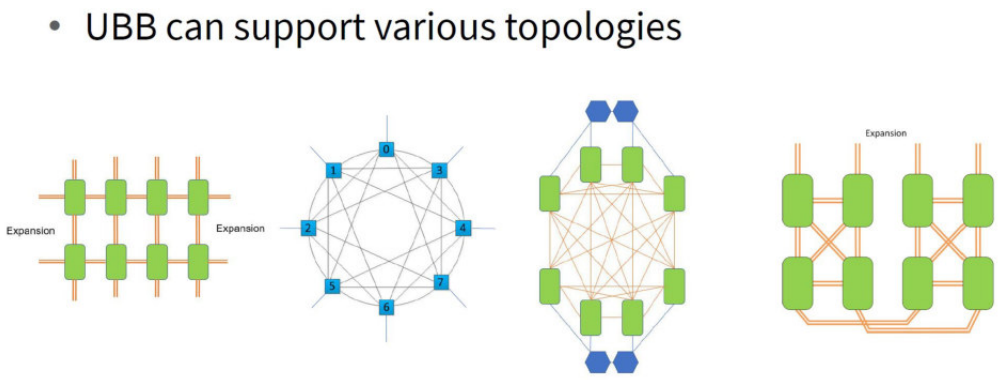


图12 UBB 互联拓扑示意图

4. OAI 1.5 (2022)：过渡与增强

随着算力需求的增长，OAI 1.5 作为中间版本，重点解决了高功耗下的供电与信号挑战。

- **功率提升：**OAM 功耗上限提升至 700W。为了应对电流增加带来的压降损耗，输入电压开始从传统的 12V/48V 混合向更高比例的 48V/54V 偏移。
- **速率演进：**互联速率提升至 112G PAM4，这标志着 PCB 布线难度的显著增加。为了维持信号完整性，规范对 UBB 的层叠结构和材料损耗提出了更严苛的要求。
- **主机链路：**Host Link 升级至 PCIe 5.0，相比于 PCIe4.0 带宽翻倍，以匹配加速器吞吐量的增长。

5. OAI 2.0 (2023)：千瓦级时代的开启

OAI 2.0 是为了应对以 ChatGPT 为代表的大模型训练需求而生，是一次质的飞跃 [8,9]。

- **极致功耗：**单模组功耗正式达到 1000W。这一功率密度使得风冷无法有效支持散热，液冷（特别是冷板式液冷）成为该规范下的推荐配置。
- **尺寸微调：**为了容纳更大的散热结构和供电模组，OAM 尺寸在长度上微调至 102mm x 170mm。
- **互联与拓扑：**全面支持 112G PAM4，并引入了重定时器（Retimer）和交换芯片作为 UBB 上的标准组件。由于 112G 速率下，信号在 PCB 上的衰减损耗过大，仅靠无源走线难以覆盖大尺寸 UBB 的互联距离。
- **可扩展性**：扩展能力大幅增强，支持 32 个 QSFP-DD/OSFP 端口，意味着全面支持单机对外可扩展互联，为构建更大规模的超节点集群提供底座支持。

6. 未来演进：迈向 224G 与机柜级架构

展望未来，技术演进将触及现有材料科学的边界，推动物理架构的彻底重构。

- **功率密度：**单模组功耗突破 1000W。供电电压可能进一步固化在 54V 甚至更高，可能需要支持垂直供电方案，以减少电流传输损耗。
- **224G PAM4 互联：**这是下一代的通信基石。然而，在 224GT/s 速率下，传统的 PCB 材料将表现出极大的介质损耗和趋肤效应损耗。
- **机柜级架构：**为了解决 224GT/s 信号在 PCB 上传输距离受限的问题，未来演进可能更聚焦超节点集群的互联架构，原来支持 8 卡机的 OAI 标准朝着更小颗粒度的单机节点 4 卡 OAM 模组转变，模组本身需要考虑 NPO 或者 CPC/CPO 的先进封装方案克服 PCB 的损耗，集群互联可能更聚焦于 Cable Tray 的紧耦合或者通过光拉远方案实现物理解耦组网部署。

表1 OAI 标准下的拓扑及硬件规格

特性	OAI 1.0	OAI 1.5	OAI 2.0	未来演进
时间	2020	2022	2023	2026+
OAM功耗	450W	700W	1000W	> 1000W (1800W+)
互联速率	56G PAM4	112G PAM4	112G PAM4	224G PAM4

特性	OAI 1.0	OAI 1.5	OAI 2.0	未来演进
OAM尺寸	102mm * 165mm	102mm * 165mm	102mm * 170mm	保持或更大
拓扑结构	FC, HCM, FC+HCM	FC, HCM, FC+HCM	新增Retimer, Switch	无阻塞全交换
集群扩展	8 * QSFP-DD	8 * QSFP-DD	32 * QSFP-DD/OSFP	机柜级Cable Tray或机柜间光互联
Host Link	PCIe 4.0	PCIe 5.0	PCIe 5.0 / 6.0	PCIe 6.0 / 7.0

CPU 与 GPU 的异构组合

超节点中 CPU 与 GPU 的组合，需要从硬件互联总线形态以及具体 AI 业务形态的维度展开。

业界 GPU 与 CPU 互联总线中，主要有以下几种形态：

PCIe (Peripheral Component Interconnect Express) 接口：每个 GPU 模组作为从设备 (Device) 挂载在 CPU 或者以 CPU 为根节点 (Root Complex，RC)的 PCIe Switch 主机设备 (Host) 下，CPU 主要负责预处理与任务调度。

NVLink-CC 私有接口协议：以 NVIDIA Grace Hopper 为代表的架构[10]，打破了传统的 PCIe 接口连接方案，通过 C2C（Chip-to-Chip）互联技术实现了 CPU 与 GPU 的统一内存寻址，该架构彻底消除了 Host 与 Device 之间的 PCIe 带宽吞吐瓶颈，支持高达 900 GB/s 的芯片间双向带宽，使得 GPU 可以直接访问海量系统内存，为万亿参数模型的单机加载提供了可能。未来基于 NVLink Fusion 进一步实现与 X86 架构的 CPU 对接，共享通算侧业务生态。

此外，AMD 基于 Infinity Fabric 高速互联总线协议[11]、海光基于 HSL (Hygon System Link) 高速互联总线协议[12]，实现自家的 GPU 和 X86/C86 处理器对接；华为基于统一总线 (Unified Bus，UB) 架构[13]，提出异构资源 CPU/GPU/内存等资源池化的概念。

从业务角度，伴随着 AI 智能体（Agent）应用兴起，其将进一步加快大模型基础设施架构的演进和发展。这一过程中，推理过程不再是单纯的矩阵计算，还包含复杂的逻辑判断、工具调用、环境交互和检索增强生成 (Retrieval-Augmented Generation，RAG)，进而导致推理负载中 CPU 密集型任务（如 Prompt 解析、API 调度、向量数据库检索）比例显著上升，尤其是需要支持超大规模并发的 Agent 执行和独立操作（通常 Agent 代理在通算的 CPU 侧部署业务），这就要求多租户隔离以及以 GPU 为中心的集群与 CPU 通算集群间实现更为弹性的交互，随之而来的是超节点中就近部署的 CPU 可用资源出现瓶颈。

未来的超节点设计可能需要更弹性的 CPU:GPU 配比，甚至出现专门的 Agent 处理节点，以处理复杂的推理规划（Reasoning/Planning）任务。

1.4 计算单元：算力芯片选型标准

人工智能迈向万亿参数大模型的时代，AI 计算单元作为超节点基础设施的核心，其选型标准早已超越了单纯的理论算力比拼。当前，芯片的效能上限还受到物理制程、封装工艺、算力/IO 带宽/存储介质间 Tradeoff 等多重制约。

本章节旨在介绍算力芯片的一些选型指标的演进历程。首先从封装技术的演进切入，探讨后摩尔时代算力提升的底层逻辑。随后展开 GPU 与领域专用架构 DSA (Domain Specific Architecture) NPU (Neural Processing Unit) 芯片在生态兼容性与能效上的架构博弈，进而尝试介绍混合精度如何通过算法与硬件的协同实现算力倍增。最后聚焦打破内存墙的关键——HBM 技术的代际演进以及 HBF 新介质形态的业界探索。

GPU 芯片的封装技术迭代

回顾以 NVIDIA 及 AMD 为首的过往 GPU 发展历程，可清晰地看到一条由封装技术主导的演进轨迹。每一个架构的飞跃，本质上都伴随着封装技术的突破。AI 计算单元的最终效能，正日益受限于芯片的算力密度、显存频宽、接口规格及功耗 [14]。

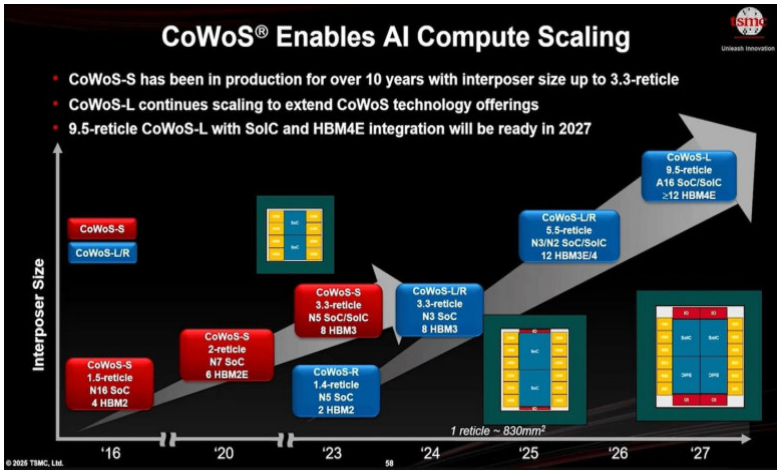


图13 先进封装制程的演进 [15]

第一阶段：2.5D 封装与 HBM 的起步 (2016–2020)

在此阶段之前，GPU 的内存通常以 GDDR (Graphics Double Data Rate) 等形式表贴焊接于印制电路板 PCB (Printed Circuit Board) 板卡，这种传统方式导致带宽吞吐性能严重受限于布线长度与信号完整性 。

2016 年，NVIDIA Pascal 架构 [16]的 Tesla P100 成为封装技术的转折点。作为首款采用 CoWoS（Chip-on-Wafer-on-Substrate）封装并合封 HBM2 的 GPU，成功将存储介质的带宽从 GDDR5 时期的约 300 GB/s 跃升至约 720 GB/s 。这一变革标志着 GPU 正式由图形渲染核心转变为数据吞吐型 AI 加速器。

随后的 2017 年，NVIDIA Volta (V100)[17]延续了 CoWoS 架构并引入 Tensor Core，封装技术的成熟使得 HBM 堆叠成为可能，进一步提升了高性能内存介质的吞吐性能 。

第二阶段：光罩极限与封装面积的挑战 (2020–2023)

掩膜版（Photomask）又称光罩 (Reticle)，是微电子制造过程中的图形转移工具或母版，是图形设计和工艺技术等知识产权信息的载体。在光刻过程中，掩膜版是设计图形的载体。通过光刻，将掩膜版上的设计图形转移到光刻胶上，再经过刻蚀，将图形蚀刻到衬底上，从而实现图形到硅片的转移，功能类似于传统照相机的底片。

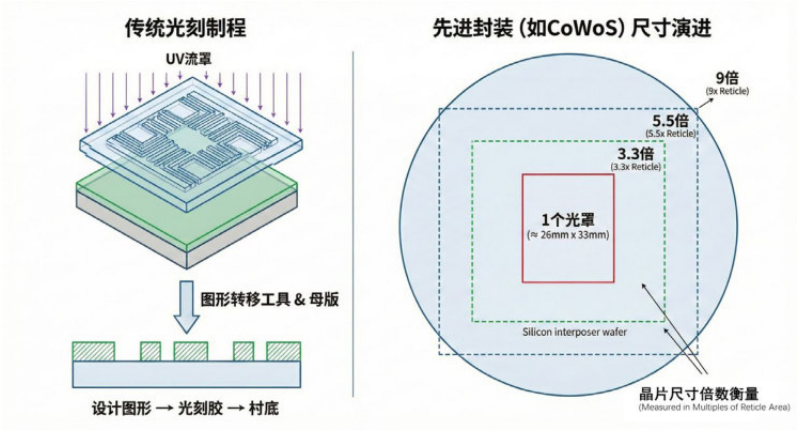


图14 先进封装制成下的光罩原理示意图

台积电基于其先进制程工艺，将 1 个光罩定义为光刻机一次曝光的最大有效面积，约为 830～858mm^2（通常指约 26mm×33mm 尺寸面积）。这是目前光刻机制造能力的极限，先进封装（如 CoWoS）常以该面积的倍数（如 3.3 倍、5.5 倍、9 倍）来衡量其封装晶片的尺寸。

随着 AI 模型参数量呈指数级增长，对算力与显存的需求迫使晶片面积不断增大，直至逼近物理制造的极限——光罩设计极限。

2020 年 NVIDIA 发布 Ampere (A100) [18]采用了台积电 7nm 制程，其晶片面积达 826mm^2，已极限接近光罩允许的最大曝光面积。为提升性能，A100 80GB 版本利用 CoWoS-S 技术整合了 6 颗 HBM2e，将 HBM 带宽推向 2TB/s，此时 CoWoS 中介层的尺寸已扩大至接近 2 倍光罩大小。

到了 2022 年，NVIDIA Hopper (H100)[19]采用了台积电 4N 制程，整合了惊人的 800 亿个晶体管，晶片面积达 814mm^2。H100 利用 CoWoS-S 新一代技术继续整合 6 颗 HBM3，尺寸达到光罩的 3.3 倍。这被视为单晶片（Monolithic）设计的巅峰之作，预示着单晶片封装已经达到极限。

与 NVIDIA 坚持单晶片不同，AMD 在 MI200 系列上率先探索了 MCM（Multi-Chip Module）设计，将两颗算力芯片进行合封，为后续的 Chiplet 路线积累了经验。

第三阶段：Chiplet 封装技术演进 (2023-至今)

这一阶段标志着 GPU 合封技术正式成为架构设计的核心驱动力，多晶片互联成为主流。

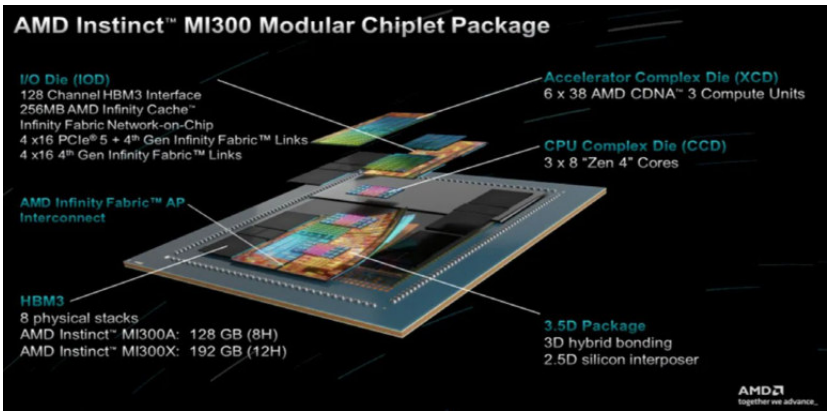


图15 AMD MI300 的 Chplet 封装示意图 [20]

AMD 在 MI300 系列[21]上为了追求性能极致，采用了 3D Hybrid Bonding + 2.5D Silicon Interposer 的复杂封装技术。

- **垂直堆叠 (3D SoIC)：**利用台积电 SoIC（System on Integrated Chips）混合键合技术，将 8 个 5nm 制程的算力 Die 垂直堆叠在 4 个 6nm 制程的 I/O Die (IOD) 之上，实现了逻辑电路的 3D 立体化。
- **水平互联 (2.5D CoWoS)：**将上述的 3D 堆叠模组与 8 颗 HBM3 存储模组，通过 CoWoS 技术封装在同一基板上，实现更高的集成度。

面对物理极限，NVIDIA 在 B200[22]上最终放弃了对单晶片的坚持，转向 Chiplet 架构。B200 由 2 颗全光罩尺寸的计算晶片组成，二者通过高达 10 TB/s 的 NVLink-C2C 进行互联 。为了承载这两颗巨大的芯粒及 8 颗 HBM3e，传统 CoWoS-S 技术因光罩面积需大于 3.5 倍而面临良率瓶颈。因此，NVIDIA 转向了 CoWoS-L 技术，利用有机基板提供更大的光罩面积支撑，进而打破物理尺寸的束缚。

展望未来

展望未来，GPU 封装技术将在材料、结构与光学整合三维度上发生革命性变化，随着先进制程的进一步演进以及 chiplet 技术的持续成熟，GPU 的算力指标/HBM 规格/IO 带宽还将继续螺旋式提升。

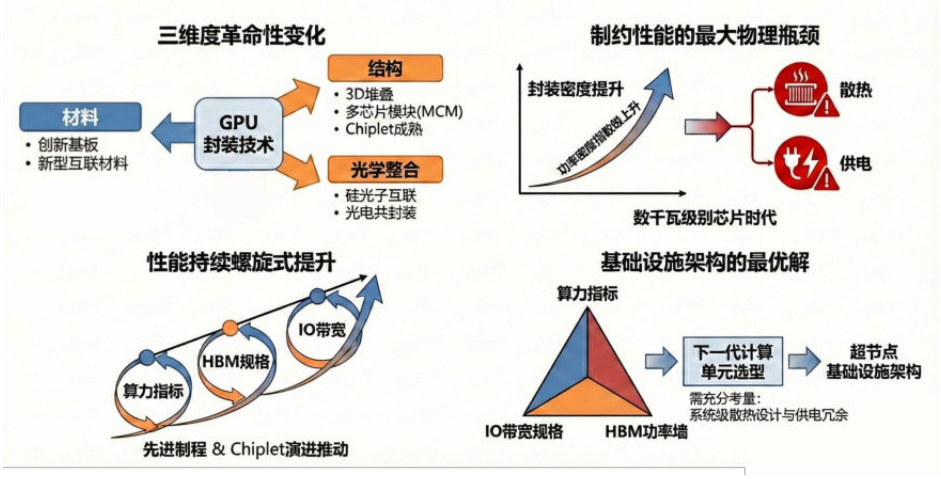


图16 未来 GPU 封装技术演进与挑战

然而，随着封装密度的提升，功率密度也呈指数级上升，散热与供电也必将成为制约 GPU 性能的最大物理瓶颈。未来的超节点基础设施架构，必须在封装极限（Reticle Limit）、高吞吐存储介质功率墙（Power Wall）与 IO 带宽规格之间寻求最优解。因此，下一代计算单元的选型，必须充分考量系统级的散热设计与供电冗余，以应对即将到来数千瓦甚至数千瓦级别芯片时代。

架构对比：通用 GPU vs 专用 NPU

AI 计算单元是超节点的算力核心。选型不能仅关注理论峰值算力 (FLOPS) 也需要关注算力的生态产业发展。

- **GPGPU 架构 (通用图形处理器)：**以 NVIDIA Hopper/Blackwell 为代表，强调通用性与生态兼容性。
    - **技术特点：**采用 CUDA Cores + Tensor Cores 的混合设计。CUDA 核心处理复杂的控制逻辑和非规则计算，Tensor Core 专攻矩阵乘加运算。

- **核心优势：**凭借极其丰富的通信算子库和灵活的编程模型（CUDA），GPGPU 能够快速适应 Transformer、MoE、Mamba 等新算法架构的快速迭代。

- **DSA NPU 架构（专用神经网络处理器）：**

  此处主要指以 Google TPU、AWS 亚马逊 Trainium 及华为昇腾系列（Ascend）为代表的领域专用架构。

  - **技术特点：**其核心通常采用专用计算单元。这种设计不像 GPGPU 那样通过大量线程并行堆叠算力，而是专为张量高维矩阵乘法进行了硬件级固化。
  - **核心优势：**在单位晶圆面积内，NPU 能提供极高的矩阵乘法（GEMM）算力密度。对于计算图固定的静态图网络（Static Graph），NPU 通过编译器优化可实现数据流的极致流水线化，往往能获得比 GPU 更高的能效比（Perf/Watt）。
  - **挑战：**由于硬件逻辑相对固化，NPU 在处理动态图（Dynamic Graph）、复杂控制流以及需要自定义新算子的场景下，开发难度较高，对编译器和算子库的依赖性极强。

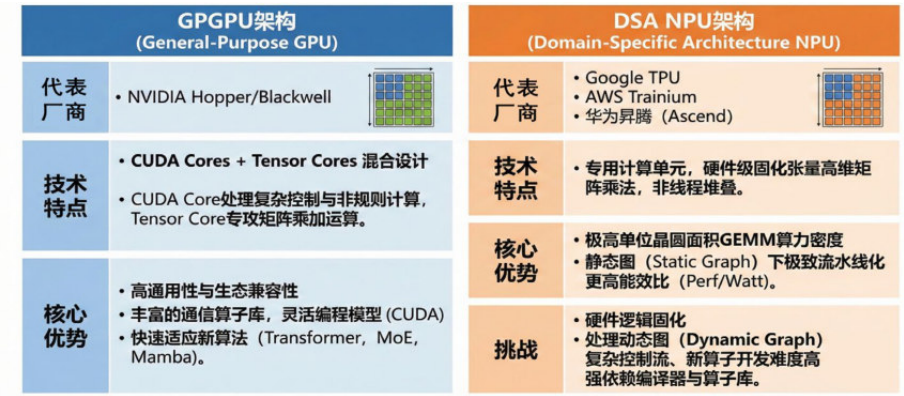


图17 AI 芯片架构对比（GPGPU vs DSA NPU）

算力精度演进

随着大模型参数量从千亿向万亿迈进，单纯依赖晶体管微缩带来的算力增长已无法匹配模型规模的指数级膨胀。算力的提升路径已从暴力堆叠逻辑单元转向算法与硬件协同的数值精度优化。通过降低计算精度来换取吞吐量的倍增，成为提升 AI 算力的关键路径。

- **总体演进主线：**
  - 早期阶段：FP32 / FP64 为主。这一阶段更强调数值稳定性和科学计算能力，双精度性能是高性能计算的重要指标。
  - 深度学习兴起：FP16 / BF16 成为主流，训练任务对数值范围和吞吐更敏感，因此半精度开始广泛用于训练，在保障精度的情况下提升吞吐和节省显存。
  - 推理优化：INT8 / FP8，随着 Deepseek V3 和 R1 的发布，使得 FP8 成为其核心精度，同时配合 BF16 等更高精度做混合精度处理。进一步提升性能并降低带宽压力。
  - 最新趋势：FP4 进入探索和落地阶段。这是进一步压缩精度换取吞吐和能效，主要面向超大模型推理。

- **混合精度训练：**

  尽管 FP32 (Single-precision floating-point) 依然是高精度科学计算的基准，但在 AI 大模型训练中，混合精度（Mixed Precision）已成为主流。

  相比于传统的 FP16 (Half-precision floating-point)，BF16 (Brain Floating Point 16) 通过截断尾数位（Mantissa）保留了与 FP32 相同的 8-bit 指数位（Exponent）。这种设计确保了足够的动态范围（Dynamic Range），有效避免了深度神经网络在梯度累加时的数值溢出或下溢问题。

- **FP8 的训练加速：**

  随着 NVIDIA Hopper 架构引入 Transformer Engine，FP8（8-bit Floating Point，遵循 IEEE 754 或 OCP 标准）开始在训练环节发挥作用。

  - E4M3（4 位指数，3 位尾数）：提供更高的精度，适用于权重（Weights）和激活值（Activations）的前向传播。
  - E5M2（5 位指数，2 位尾数）：提供更大的动态范围，适用于反向传播中的梯度（Gradients）计算。

  相比 BF16，FP8 在同等显存容量下支持更大的 Batch Size，实现了理论算力翻倍的效果。

- **推理算力的极致演进：FP4 与微缩放（Micro-scaling）技术**

  在推理侧，为了应对超大模型部署带来的内存墙与 IO 带宽瓶颈，业界正激进地向 4-bit 甚至更低精度演进。

  传统的整数量化（INT4）在处理大模型（如 LLM 特征值）时会导致显著的精度崩溃，而标准的 FP4 由于位宽极窄，其动态范围和精度极难平衡。

针对这些挑战，Blackwell 等新一代架构引入了微缩放技术（Micro-scaling）。其核心思想不再是对单个数值进行量化，而是引入块浮点（Block Floating Point）概念，实现：

- **块级共享指数（Block Shared Scale）：** 将一组连续数据（如 16 个或 32 个元素构成一个 Block）共享一个高精度的缩放因子（Scale Factor，通常为 8-bit）。
- **细粒度量化：** 在 Block 内部，每个元素仅存储低精度的尾数。

这种技术在数学上实现了伪高精度，即在保持模型精度几乎无损的前提下，将内存占用压缩至 FP16 的 1/4，并将理论推理吞吐量提升至 FP8 的 2 倍。这可能成为未来芯片架构设计中，考量其是否具备大模型原生推理能力的关键指标。

HBM 与内存墙的博弈

AI 负载中，内存带宽往往比计算峰值更早成为瓶颈。在这种背景下，高带宽内存 HBM 标准的发展成为了行业焦点。

随着技术的演进，HBM4 将成为 2026 年推出的下一代 AI GPU 的核心内存。AMD 和 NVIDIA 均已确认在其新一代产品（如 NVIDIA 的 Rubin 和 AMD 的 Instinct MI400）中采用 HBM4 技术。NVIDIA 的 Rubin GPU 规划封装了 8 颗 HBM4，而高端的 Rubin Ultra 则会配备 16 颗 HBM4E，其搭载的 HBM4E 内存总容量可能超越 512GB。预计 Rubin Ultra 功耗将超 2200W，其中 HBM4 的功耗接近 600W（高性能内存功耗占比接近 30%）。

表2 HBM 的技术演进 [23]

规格	HBM2E	HBM3	HBM3E	HBM4	演进
时间	2020	2022	2024	2026	2029（预计）
单引脚速率（Gbps）	3.2	6.4	8	8	8
接口位宽（bit）	1024	1024	1024	2048	4096
单Stack带宽	460 GB/s	819 GB/s	1 TB/s	2 TB/s	4 TB/s
Capacity/die	16 Gb	16 Gb	24 Gb	24 Gb	40 Gb
#of die Stack	4/8-Hi	8/12-Hi	8/12-Hi	12/16-Hi	16-Hi
容量	8/16 GB	16/24 GB	24/36 GB	36/48 GB	80 GB
Power/HBM	19 W	25 W	32 W	43/75 W	100 W

JEDEC 关于 HBM 下一代技术演进还在讨论和定义中，未来 HBM 的规划不仅是速率和容量提升，更涉及到底层架构的根本性变革 [24]：

- **混合键合（Hybrid Bonding）的强制化：** 在 HBM4 阶段，JEDEC 依然允许通过放宽厚度限制来使用传统的微凸点技术。但对于 HBM5，为了在有限的 775 μm 高度内塞进 20 层甚至更多的芯片，JEDEC 可能将无凸点直接铜-铜键合作为核心标准。
- **定制化基础层（Custom Base Die）：** HBM5 将深度推进定制化趋势。JEDEC 也正与台积电等晶圆厂协作，定义 HBM5 的 Base Die 逻辑层。这允许 AI 厂商将部分逻辑电路甚至简单的计算单元集成在内存底部的逻辑层中。
- **低摆幅与先进均衡技术：** 随着数据速率可能达到 10Gbps+，信号完整性挑战巨大。JEDEC 规划在 HBM5 接口中引入更先进的决策反馈均衡器（DFE）和小摆幅接口技术，以在维持高带宽的同时显著降低每比特功耗。

HBF 新介质的发展和演进

当前，HBM 是解决介质带宽问题的绝对主力，但其物理扩展性、成本及功耗限制了其在超大规模模型推理场景中的普及。正是在这一背景下，基于 NAND Flash 技术的高带宽闪存（High Bandwidth Flash, HBF）介质也被业界厂商所提及 [25]。HBF 并非传统 SSD 的简单加速，而是一种全新的存储层级，它试图通过 3D 堆叠、TSV（硅通孔）互联以及异构键合技术，将非易失性存储的密度优势与 DRAM 级别的接口带宽相结合。其旨在填补 DRAM 与传统 SSD 之间巨大的性能鸿沟。

- **定义：** HBF 是利用 3D NAND 技术，通过 HBM 式的先进封装（如 TSV 和 Wafer-to-Wafer Bonding）实现极高并行度的存储介质 [26]。
- **关键指标：**
  - **带宽：** HBF Gen 1 目标读取带宽约为 1.6 TB/s，与 HBM3E 持平或接近；
  - **容量：** 单颗 HBF 堆栈容量可达 512GB 至 1TB，是 HBM 的 16 倍以上；

- **延迟：**HBF 的物理介质仍然是 NAND，其随机读取延迟在微秒级 (μs)，远高于 DRAM 的百纳秒级 (ns)。但对于 AI 推理这种吞吐受限型 (Throughput-bound) 而非时延受限型 (Latency-bound) 的负载，高带宽可以掩盖高延迟。

表3 三种存储介质的技术指标对比

技术指标	HBM3E (当前主流)	HBF Gen 1 (HBF1)	NVMe SSD
介质类型	DRAM (易失性)	3D NAND (非易失性)	3D NAND (非易失性)
单堆栈/设备容量	24GB – 36GB	512GB – 1TB	3.2TB – 30TB+
读取带宽	1.2 TB/s	~1.6 TB/s	~14 – 28 GB/s
介质响应延迟	~14 ns	~10 – 20 μs	~70 – 100 μs
接口技术	TSV / 1024-bit Wide I/O	TSV / CBA Logic Interface	PCIe / NVMe 协议
功耗特性	需持续刷新 (高静态功耗)	零待机功耗	低
应用场景	训练热数据、模型激活值	推理 KV Cache、冷权重、CheckPoint	数据集存储、冷归档

HBF 之所以能被称为高带宽，并非仅仅因为其堆叠了更多层数，其核心在于对 NAND 架构的颠覆性重构。传统的 NAND 设计受限于 IO 接口速度和内部并行度，而 HBF 引入了以下关键技术突破。

CBA (CMOS directly Bonded to Array) 技术

在传统 3D NAND 中，负责控制读写电压、逻辑寻址的外围电路，通常位于存储单元阵列周边。这种设计迫使外围电路必须使用与存储单元相同的制造工艺，这通常是针对存储密度优化而非逻辑性能优化的传统制造工艺，导致 I/O 接口速度受限。

HBF 采用 CBA 也称晶圆键合技术。

- **分离制造：**将 3D NAND 存储阵列晶圆与 CMOS 逻辑控制晶圆分别在不同的生产线上制造。存储晶圆专注于堆叠层数 (如 300+ 层 BiCS 技术) 以提高密度；逻辑晶圆则采用先进的逻辑工艺节点 (如 14nm 或更先进)，以实现极高的开关速度和驱动能力。
- **混合键合：**两个晶圆通过数百万个微小的金属触点进行面对面键合 (Hybrid Bonding)。
- **性能跃升：**这使得 HBF 的 I/O 接口速度大幅提升。例如，基于 BiCS9 技术的 HBF 接口速度可达 4.8 Gbps，结合极宽的总线位宽，实现了总带宽的飞跃。

并行子阵列架构 (Parallel Sub-Array Architecture)。

为了达到 TB/s 级的带宽，HBF 彻底改变了 NAND 的内部数据通路。

- **传统架构瓶颈：**普通 SSD 的 NAND Die 通常被划分为 4 或 8 个平面。在同一时刻，只支持有限个平面执行读写操作。
- **HBF 创新：**HBF 将存储阵列细分为多个子阵列。每个子阵列都拥有独立的数据读写通路。

- **大规模并行：**底部的逻辑控制芯片（Logic Die）可以同时从众多子阵列中提取数据。虽然单个 NAND 单元的物理读取时间仍然是微秒级，但通过并行发起数千个读取指令，HBF 能够以流水线的方式填满高速接口，从而在外部呈现出类似 DRAM 的吞吐量。

TSV 垂直互联与封装。

与 SSD 采用引线键合（Wire Bonding）不同，HBF 直接沿用了 HBM 的封装工艺。

- **16-Hi 堆叠：**HBF 支持采用 16 层 NAND Die 堆叠。
- **硅通孔 (TSV)：**通过在硅片上打孔并填充金属，实现了 Die 与 Die 之间、Die 与底部逻辑 Die 之间的垂直电气连接。TSV 的互联长度极短，寄生电容小，有利于高频信号传输并降低功耗。
- **热管理挑战：**由于 16 层 NAND 加上高速逻辑层会产生显著热量，且 NAND 对高温（导致数据保持力下降）敏感，HBF 封装必须引入先进的散热设计。

表4 传统存储介质与 HBF 的内部原理对比

技术支柱	传统 NAND/SSD	HBF	收益
外围电路	CuA (Circuit Under Array)	CBA (Wafer-to-Wafer Bonding)	解耦存储与逻辑工艺，提升 I/O 速度
内部架构	2-4 Planes / Die	Massive Parallel Sub-Arrays	极大提升数据并发度，以空间换时间
互联方式	Wire Bonding	TSV (Through-Silicon Via)	缩短信号路径，降低功耗，提升频率
封装形式	BGA	HBM-like Stack on Interposer	极高密度集成，靠近 GPU 部署

在 Transformer 架构中，为了生成下一个 Token，模型需要计算当前输入与所有历史 Token Attention。为了避免重复计算，历史 Token 的 Key 和 Value 矩阵会被缓存下来。

- **线性增长：**KV Cache 的大小与 Context Length（上下文长度）成正比。
- **数据量级：**对于一个 70B 参数的模型，在 128K 上下文长度下，仅 KV Cache 就需要占用超过 150GB 的显存 (超过 H100 单卡 80GB 的容量上限)。
- **并发瓶颈：**在服务高并发用户时，GPU 显存迅速被 KV Cache 填满，导致系统被迫通过“换入换出”来服务不同用户，导致严重的性能抖动。

HBF 大容量大带宽的特性与 KV Cache 的访问模式相契合：

- **多读少写特性：**在推理的预填充（Prefill）阶段生成 KV Cache 后，后续的解码（Decode）阶段主要是反复读取。规避了 NAND 写入寿命和写入速度的弱点。
- **吞吐敏感：**解码阶段是内存带宽受限的（Memory Bound）。HBF 提供的 1.6 TB/s 读带宽可以快速将巨大的 KV Cache 块流式传输给 GPU 计算单元。
- **容量需求：**4TB 的 HBF 足够存储数百个并发用户的长上下文 KV Cache，实现真正的“无限上下文”推理，而无需频繁地重新计算。

在 AI 超节点架构中，HBF 未来支持扮演关键的分级存储角色，特别是作为 KV Cache 的巨型容器，释放被存储束缚的计算潜能。对于架构设计而言，未来的关注点也支持从如何封装更多 HBM 转变为如何优化 HBM 与 HBF 之间分级存储，HBF 有望成为继 HBM 之后，AI 硬件领域的又一次革命性突破[27]。不过从新技术走向产品的成熟应用，HBF 还需要解决 NAND 介质目前 10 万次写入寿命的问题。

小结

后摩尔时代，AI 算力芯片的选型已彻底脱离了单点性能的线性比拼，转而进入系统工程的综合比拼。当前的技术演进呈现出清晰的制约关系：

- **封装定义的算力边界：**随着单晶片面积逼近光罩（Reticle）物理极限（约 858mm^2），CoWoS-L/R 等先进封装技术进行 Chiplet 合封已成为延续摩尔定律的关键路径。未来的算力单元实际上是由计算芯粒、IO 芯粒与 HBM 堆栈通过 2.5D/3D 技术异构集成的系统级芯片。
- **精度换取计算能力：**从 FP16 到 FP4 Micro-scaling 的演进，本质上是利用大模型整体稳定性，通过降低单点计算精度来换取系统级吞吐量的指数级跃升。
- **打破内存墙的高性能存储介质持续演进：**面对模型参数的指数级增长，内存带宽不仅是性能瓶颈，更是功耗瓶颈。随着 HBM4 及后续标准的引入，3D 垂直堆叠（HBM 直接键合于逻辑 Die 之上）将成为解决带宽与能效比矛盾的可演进方案。同时随着如 HBF 等新介质和新技术的出现，也将为大语言模型的持续发展注入新的动力。

1.5 超节点互联拓扑

人工智能训练需求正推动高性能计算基础设施经历范式转移的这一进程中，除了核心的算力节点，互联带宽（Interconnect Bandwidth）与网络拓扑（Network Topology）也已成为决定集群训练效率、可扩展性及经济可行性的关键变量。当单一大模型训练任务需要跨越数百甚至千卡集群协同工作时， 智算中心网络架构面临前所未有的“通信墙”挑战。

本章节旨在探讨支撑现代 AI 算力底座的几种互联拓扑——CLOS（多级交换架构）、Torus（环面直接互联）与 DragonFly（高基数分层互联）。

拓扑设计的核心评价指标

评估一个互联架构是否适用，通常需考察以下几个关键指标，在 AI 组网中，网络拓扑也与其上层业务的开展形式相关：

- **路由最短路径（Diameter）：**数据包从源节点到目的节点所需经过的最少跳数（Hops）。路径越短意味着越低的端到端延迟。
- **对分带宽（Bisection Bandwidth）：**将网络切分为均等两部分时，切面上所有链路的总带宽。全对分带宽（Full Bisection）意味着网络无阻塞。
- **基数（Radix）：**单个交换芯片可提供的端口数量。高基数交换是实现低跳数网络（如 DragonFly）的物理基础。

CLOS 架构：无阻塞交换

CLOS 网络[28]，尤其是其折叠形式（Folded Clos）或胖树（Fat-Tree），是目前大多数商业 AI 集群的业界主流选择（如 NVIDIA 的超节点集群解决方案、AMD 即将发布的基于 UALink 的组网解决方案、博通主导的 ESUN 的 Scale Up 组网解决方案）。

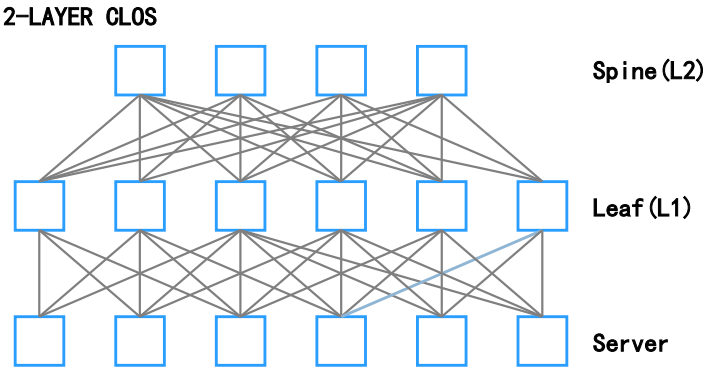


图18 CLOS 架构组网拓扑示意图

1. 架构特征

采用单级或者多级交换机（Leaf，Spine）构建。在理想的无阻塞（Non-blocking）设计中，任意两台服务器之间的带宽恒定，通信交互路径具有高度的多样性。考虑到链路端到端时延（包括芯片多级转发时延）的综合影响，业界 AI 集群 Scale up 网络部署交换通常不超过两级。考虑到集群组网的流控机制的复杂性，支持故障场景下更加高效的重传能力，超节点在协议层面更聚焦在单级交换的实现，技术上宣称更大规模组网条件下支持二级交换。

2. 优势分析

- **流量无感：**由于提供全对分带宽，CLOS 对流量模式不敏感，无论是随机流量还是 All-to-All 流量，均能较好应对业务挑战。
- **路由简单：**支持更合理的负载均衡实现。
- **故障隔离：**单一链路或交换机故障一定程度上只影响吞吐带宽大小，不破坏连通性。

3. 劣势与瓶颈

- **组网规模受限：相比**于下一小节要介绍的 Torus/DragonFly 拓扑，在端到端转发时延受限条件下，CLOS 最大组网规模直接受限于交换芯片的 Radix 支持的交换端口数量。
- **成本较高：**随着规模扩大，交换机数量和节点间接线数量呈超线性增长。

不过 Scale up 域内的 GPU，在高带宽吞吐和低时延转发诉求的客观背景下，节点规模一般受限在千卡以内，最大支持两层交换实现全互联，一层交换实现百卡以内规模全互联规模，支持超节点训推业务。

4. 典型应用：NVIDIA NVL576 互联

NVIDIA GB200 NVL576 采用机架级液冷设计：柜内 NVL36 实现 18 颗 Grace CPU 和 36 颗 Blackwell GPU 组网。该架构采用第五代 NVLink 互联技术，配合专用的 NVSwitch 交换芯片，支持 36 颗具有 1.8 TB/s 的双向互联带宽的 GPU 实现 NVIDIA NVLink 高带宽域，作为一个 GPU 大集群开展业务部署。

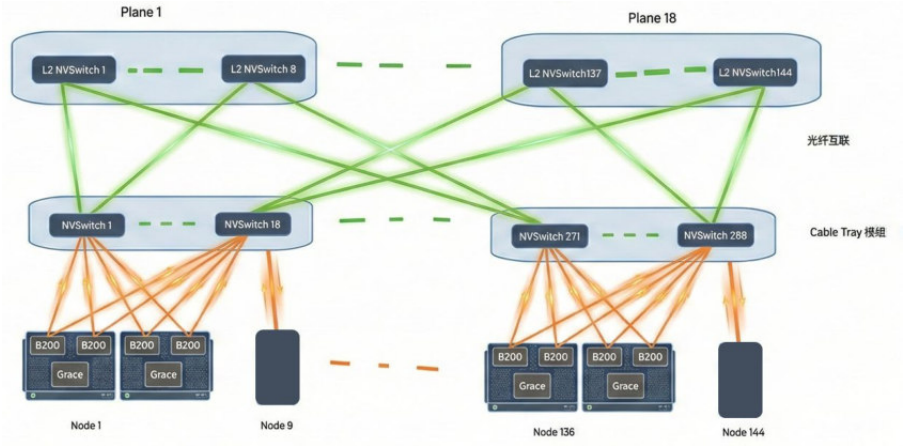


图19 GB200 NVL576 组网拓扑示意图

NVIDIA 基于 NVL36 构建 NVL576 超级集群，支持 HBD 域内扩展至 576 颗 GPU。在其超级集群架构中，通过 NVSwitch 的高密度端口交换能力，系统利用 2 层 Clos 网络拓扑实现 BlackWell GPU 的全互联：

- **整机柜：**单机柜集成 36 颗 B200 GPU 和 18 颗 NVSwitch 芯片。利用 NVSwitch 的 72 个端口扩展能力，其中 36 个端口下行连接柜内 GPU，剩余 36 个端口作为上行链路，构建无阻塞的互联通信基座。
- **机柜间集群：**整个集群包含 16 个整机柜。二层的 NVSwitch 网络充当脊层（Spine），其 72 个端口分别连接至 16 个机柜中对应的一层 NVSwitch，实现跨机柜的无损带宽聚合。

通过高效的两层 Clos 组网，构建了一个统一的高带宽 NVLink 域，确保集群内任意两个 GPU 之间均能以 1.8 TB/s 的高速互联，最大化释放 Blackwell 架构在超大规模模型训练中的效率。

Torus 架构：极致的局部性能与组网成本收益

Torus（环面）是一种直接互联拓扑 [29]，节点直接连接到其在 n 维空间中的相邻节点。

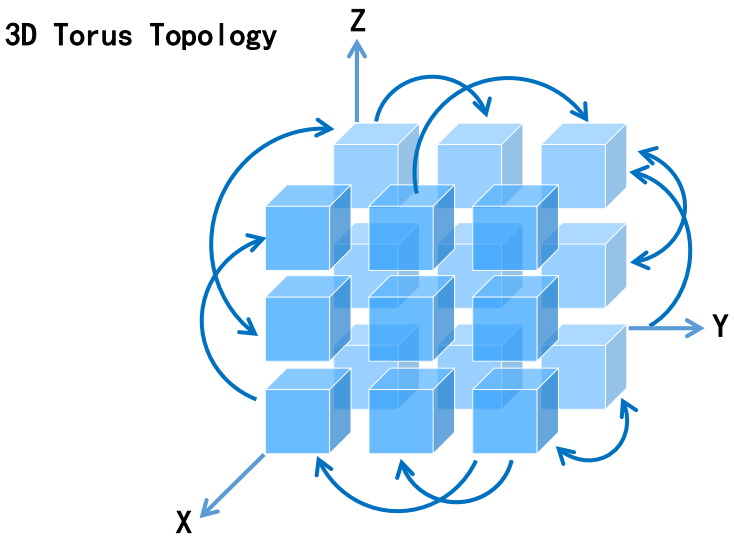


图20 3D Torus 架构组网拓扑示意图

1. 架构特征

每个节点有 2n 个相邻节点，其中 n 代表维度（例如 3D Torus 中 n=3，就有 6 个相邻节点，包括上下/前后/左右）。边缘节点通过环绕链路（Wrap-around links）连接到对侧，形成闭环。

2. 优势分析

- **布线成本低：** 绝大多数互联是近邻节点通信，可大量使用低成本电缆，不涉及交换。
- **邻近节点通信效率高：**AI 业务中，对于 TP 或 PP 这种具有强局部性的通信，Torus 拓扑可实现单跳直达。

3. 劣势与瓶颈

- **整体拓扑点到点路径长：** 跳数随节点数 N 的 1/n 次方增长。在大规模集群中，长距离通信（如 DP）延迟较高。
- **对分带宽低：**相比 CLOS，Torus 的对分带宽受限于切面链路数，处理 MoE 的 All-to-All 流量时极易拥塞，需要业务面协同调度。
- **灵活性不足：** 节点间互联固定，任意节点通信存在需要中间节点多轮转发传递，难以适应不同要求的并行任务诉求，训练/推理业务需要预先编排，且单点故障可能造成环路完整性受损，造成局部训练/推理业务中断。

4. 典型应用：Google TPU Pod

Google 的张量处理器单元（Tensor Processing Unit，TPU）集群（v2/v3/v4）采用了 2D/3D Torus 互联。Google 通过编译器优化，将模型切片放置在相邻节点，利用 Torus 的邻节点高带宽特性，避开了全局通信的短板，同时创新性利用 OCS 光交换机，实现业务故障快速恢复[30]。

TPU v4 集群物理架构：超大规模集成。

- **基本单元（Board）：** 每个 Tray 盘包含 4 个 TPU 芯片。
- **机架（Rack）：** 每个机架包含 16 个 Tray 盘，即 64 个 TPU。
- **集群（Pod）：** 一个完整的 TPU v4 Pod 由 64 个机架组成，总计包含 4096 个 TPU 芯片。

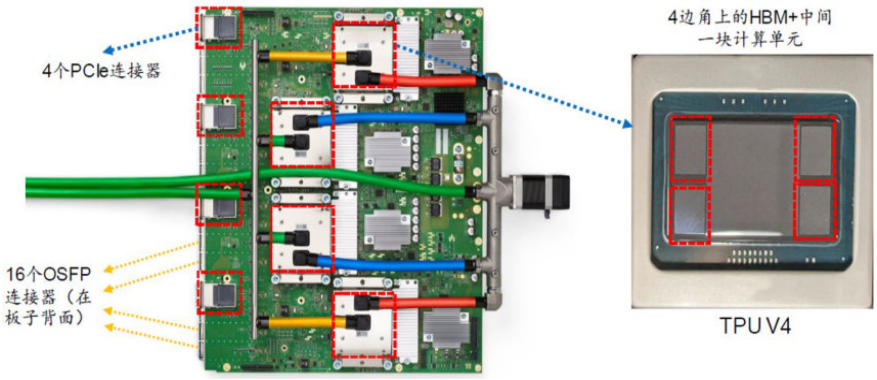


图21 TPU v4 板级接口示意图

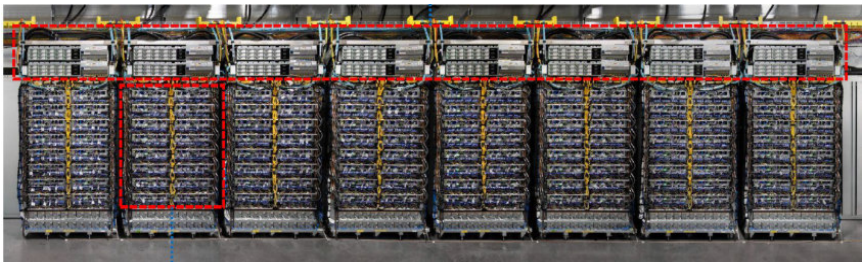


图22 TPU v4 集群示意图

OCS 光电路交换。

作为 TPU v4 网络的核心，OCS 光交换机具备以下技术特点：

- **技术原理：** 不同于传统的电交换机，OCS 基于 MEMS（微机电系统）阵列，通过调整镜面角度在光层直接进行物理路径切换，无需光电转换，实现了极低的功耗和延迟。
- **端口规格：** Palomar OCS 提供 136 x 136 的端口规模，为非阻塞式架构。在实际部署中，使用了 128 个端口，预留 8 个作为备份（Spare Ports）。

3D-Torus 拓扑组网方案详解。

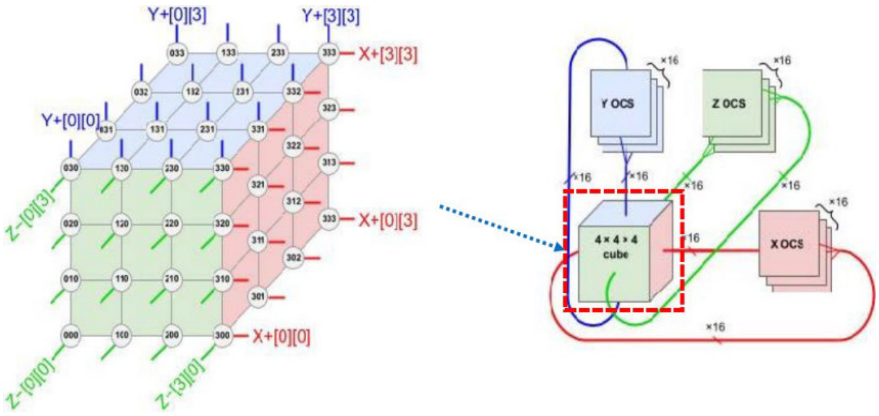


图23 Google 3D Torus 组网示意图

- **机架内的“立方体”抽象。**

  每个机架内的 64 个 TPU 在逻辑上被互联为一个 4×4×4 的立方体（Cube），这意味着机架内部主要依靠铜缆（DAC/AEC）或背板 PCB 互联，形成一个紧密的计算节点。

- **机架间的“环面”构建。**

  为了将 64 个机架互联成一个巨大的 3D-Torus 网络，Google 采用了以下连接策略：

  - **六面出线：** 每个 4x4x4 的立方体有 6 个面。每个面有 4×4=16 条链路需要与外部连接。
  - **总链路数：** 一个机架总共有 6×16=96 条对外光纤链路。
  - **OCS 聚合：** 理论上需要连接 96 个方向，但为了优化布线，Google 将同一面上的链路成对连接到同一台 OCS 交换设备。整个集群需 48 台 OCS 交换机完成互联组网。

规模	互联介质	TPU互连(ICI)
TPU v4 芯片	ICI接口规格	6 x 56Gbps X8
TPU v4 节点板	板内PCB互连 OSFP-400G	**板内：** 2 x 56Gbps X8 芯片间 **板外：** 4 x 4 x 56Gbps X8
CPU 计算节点板	100Gbps TOR + Leaf/Spine走以太交换	intel Casecade Lake 2P，PCIe 3.0 x16 (128Gbps)，双卡2 x 100Gbps规格 Ethernet
intra-SuperPOD（机柜内）	DAC电缆	**POD内：** 4 x 56Gbps X8 节点板间 **POD外：** 4 x 56Gbps X8 POD间
inter-SuperPOD（机柜间）	DAC电缆+光互联	**SuperPOD机柜间带宽：** 4 x 16 x 56Gbps X8 光交换+ 机柜间电缆

图24 Google TPU v4 集群接口示意

- **动态拓扑重构。**

  得益于 OCS 的特性，TPU v4 集群可以动态改变拓扑结构（例如从 3D Torus 变为 Twisted Torus），以绕过故障节点或为特定模型训练优化数据流路径。

架构优势与光模块效率。

- **光模块用量节省：** 相比于 NVIDIA 采用的 InfiniBand 胖树架构（GPU 与光模块比例约为 1:2.5），TPU v4 的架构极其节省光器件。整个集群中，TPU 数量与光交换端口数量之比为 4096:6114 ≈ 1:1.5。
- **优势归因：** 这种高效率归功于 3D-Torus 拓扑 的邻近节点互联特性（大部分流量只在相邻节点间传输）以及 AI 训练数据流的局部性特点。

DragonFly 架构：高基数下的转发优化

DragonFly 由 John Kim 等人于 2008 年提出 [31]，旨在利用高基数路由器打破延迟与成本的权衡，在 HPC 网络中有大量应用（如 Cray Slingshot 互联）。

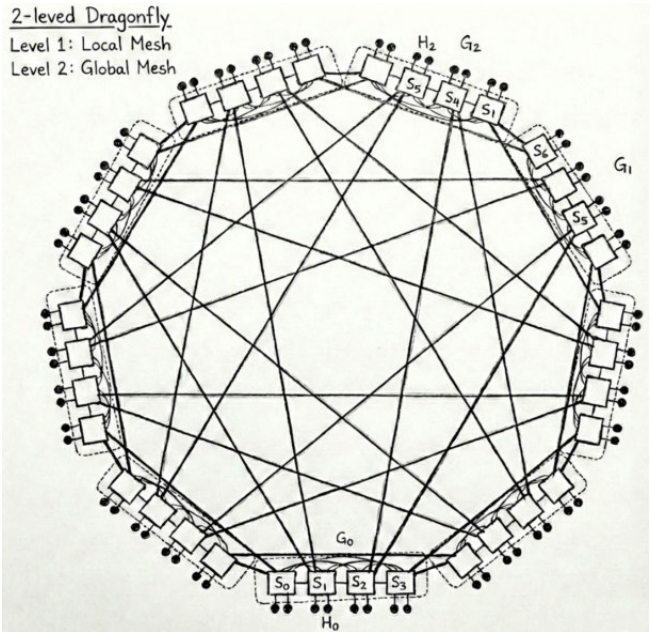


图25 DragonFly 架构组网拓扑示意图

1. 架构特征：分层结构

- **组内（Intra-group）：**多个交换机形成一个全互联或高密度互联的“超级节点”（Group）。
- **组间（Inter-group）：**所有 Group 之间形成全互联，即任意两个 Group 之间至少有一条直连链路，任意卡间带宽收敛。

2. 优势分析

- **高可扩展：**通过分层级全域互联设计，支持超大规模集群组网。
- **极低直径：**无论规模多大，理论上任意两点间最多只需 3 跳（本地-全局-本地）。
- **成本效益：**充分利用高基数交换芯片的能力，能极大减少了昂贵的全局线缆部署（相比 Fat-Tree 减少约 50%）。

3. 劣势与瓶颈

- **路由复杂：**最小路由（Minimal Routing）容易导致全局链路拥塞。必须采用非最小自适应路由（Adaptive Routing），将流量偏转到非直连组以平衡负载，这给业务部署带来死锁处理和保序的挑战。
- **组网复杂：**新增集群节点，需要重新布线组网，灵活性较差。

4. 典型应用

华为发布的 UB-Mesh（Unified Bus Mesh）架构 [32]，提出了分层局部化 nD-FullMesh（Hierarchically Localized nD-FullMesh）概念，在大规模组网条件下，该拓扑理论上与 DragonFly 有着一定层度的同源性，但在工程实现上基于 AI 业务调度进行了局部优化，其利用 AI 训练流量的局部性特征，在机柜内、机柜间构建多维全互联网络。

UB-Mesh 利用自研 Switch 芯片的高端口密度，构建了一个扁平化网络实现高性能、大规模和高可靠组网，其通过时空均衡技术充分利用最短和非最短路径、逐包/逐流动态路由等技术，减少网络拥塞，在处理 All-to-All 等复杂流量模式时，能获得较高有效带宽。在同等规模下，相比于 CLOS 拓扑，UB-Mesh 减少交换机层级和光模块数量的同时，通过编排调度支持提供更低的端到端延迟。

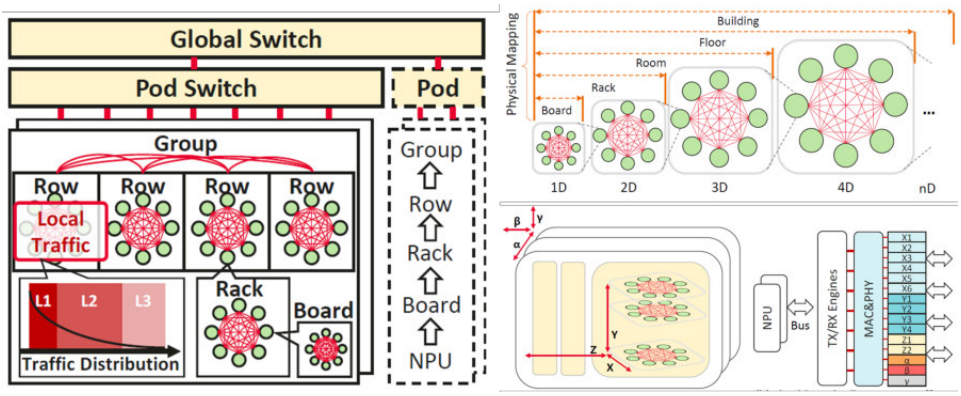


图26 Huawei UB-Mesh 架构组网拓扑示意图

三种互联拓扑综合对比：

特性维度	CLOS (Fat-Tree)	Torus (3D)	DragonFly
互联类型	间接互联 (Switch Fabric)	直接互联 (Direct)	分层直接互联 (Hierarchical Direct)
网络直径	低且恒定 (3-5跳)	高 ($O(N^{1/3})$)	极低 (最大3跳)
对分带宽	极高 (Full Bisection)	较低 (受限于维度切面)	中等 (依赖自适应路由)
扩展成本	高 (超线性增长)	低 (线性增长)	中低 (利用高基数芯片)
光缆依赖度	较高 (尤其是二级跨柜交换)	极低 (相邻节点主要用铜缆)	较低 (仅组间互联需要)
AI负载适应性	通用性强，MoE (All-to-All) 最佳	TP/PP (局部通信) 最佳	平衡型 (需配合路由优化)

CLOS 架构：作为 NVL72 等超节点的核心拓扑，其优势在于全对分频宽带来的无阻塞交换性能。尽管随着规模扩大其交换芯片成本呈超线性增长，但在 Scale-Up 域内，它是处理 MoE 等 All-to-All 流量的最佳方案。

Torus 架构：以 Google TPU 为代表，其极致的近邻节点通信效率在 TP 与 PP 中表现优异。然而，由于其网络转发跳数较长，在应对 MoE 这种需要全局大频宽交换的流量时，极易产生拥塞，需要特殊的业务预编排规划实现性能最优。

DragonFly 架构：该架构利用高基数交换芯片实现极低跳数，主要应用于 HPC 场景。华为 UB-Mesh 引入了 nD-FullMesh 概念，针对 AI 业务调度进行了局部优化，平衡了组网成本与 TP/PP 通信的局部性需求。

从通用互联到算网一体

- **机柜即节点：**无论是 NVL72 还是 Google 以及未来的其他超节点解决方案，都在物理层面上将一个机柜或者一个集群定义为不可分割的计算单元。
- **网络扁平化：**传统的多层交换架构通用性强，但 GPU 间基于总线交换的内存语义通信对端到端时延的要求，一定程度上约束了超节点业务交换的转发层数，高 Radix 和大交换带宽的交换机依然是超节点组网的架构趋势。
- **拓扑的软件定义与动态化：**Google OCS 证明了根据模型结构调整物理拓扑的巨大价值。未来，网络不再是静态的管道，同样也是编译器可优化的资源。

1.6 交换芯片技术

在以互联为中心的时代，交换芯片的基数与频宽直接定义了超节点的扩展边界。在这种架构中，数十甚至上百个加速器通过高带宽、低延迟的内存语义互联网络紧密耦合。此过程反映了算力需求的爆发式增长与数据传输瓶颈之间的博弈，整体呈现出 GPU 的互联协议从通用兼容向私有协议追求极致性能，再到开放生态的演进探索。

Year	2010-2013	2014-2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028
PCIE	8G NRZ (3.0)		16G NRZ (4.0)		32G NRZ (5.0)		64G PAM4 (6.0)				128G PAM4 (7.0)			
Ethernet	10G NRZ	25G NRZ			53G PAM4		106G PAM4				212.5G PAM4			
InfiniBand	10G NRZ	25G NRZ		53G PAM4			106G PAM4				212.5G PAM4			
NVLink		20G NRZ (1.0)		25G NRZ (2.0)		50G NRZ (3.0)	100G PAM4 (4.0)				200G PAM4 (5.0)			
UALink											128G PAM4 (1.0)		224G PAM4 (TBD)	

图27 交换协议底层 SERDES 技术演进

通用时代的瓶颈：PCIe 总线

在早期，GPU 仅作为外设存在，PCIe 是唯一的选择。但随着 AI 模型参数量激增，PCIe 固有 Serdes 速率低以及对应交换芯片带宽吞吐受限，成为多卡并行训练的瓶颈。而且传统 PCIe 总线协议报文头部开销大，不支持原子操作和缓存一致性，导致其在处理细粒度、高频次的 GPU 间同步时效率低下，也无法支持 GPU 之间实现高效显存共享，数据必须在 CPU 和内存间反复搬运，造成算力资源的极大浪费。与此同时 CXL (Compute eXpress Link) 作为 PCIe 物理层上构建的缓存一致性系统协议，旨在解决数据中心内存扩展和性能瓶颈问题的 GPU 生态尚未成熟。

不过 PCIe 作为 GPU 与 CPU 处理器的关联纽带，依然会持续影响 AI 的生态发展。未来，PCIe 技术正朝着 Gen 7.0(128 GT/s)演进，单端口最大吞吐双向带宽突破 512 GB/s，其在大模型训练的影响力还在持续。

算力孤岛的突围：NVLink 的技术主导

为了打破 PCIe 的瓶颈，NVIDIA 利用以太协议底层生态成熟的互联技术，推出了 NVLink 私有协议，实现接口 Serdes 速率和交换带宽的超越及持续引领。

2025 年 NVIDIA 成功推出基于 200 GT/s、吞吐带宽性能支持双向 1.8TB/s 的 GPU 产品 BlackWell 以及配套 NvSwitch 5.0 交换芯片（支持 72 个端口，总吞吐性能达到 7.2TB/s），实现整机柜 72 张 BlackWell 模组支持统一 HBM 内存共享，定义了业界 GPU 高性能互联的标杆。

开放生态的兴起：UALink、SUE-T 以及 OISA、CLink

面对 NVIDIA 的技术主导，业界巨头（AMD, Intel, Google 等）组建联盟，推出了 UALink (Ultra Accelerator Link) 及相关的开放标准。预计 2026 年 Q3，AMD 将推出业界首款基于 UALink 的超节点整机柜 Helios，双宽 ORW（Open Rack Wide）标准机柜，实现 72 颗 AMD Instinct MI455X 互联。

Meta、博通等也基于开放计算项目(Open Compute Project，OCP)组织，组建了 ESUN（Ethernet Scale-Up Network）工作组，并基于以太为底层，推出了博通主导的 SUE-T（Scale-Up Ethernet Transport）协议。

国内，以中移动牵头的 OISA（Omni-directional Intelligent Sensing Express Architecture）、新华三牵头的 GLink、华为牵头的 UB 以及在此基础上，中电标准化研究院联合产业上下游共识推出的 CLink 计算互联总线协议及相关标准体系，为大规模智算集群提供高效、智能且开放的 GPU 卡间互联标准提供了新的选择。

表5 当前 Scale-Up 互联芯片指标对比

接口总线	PCIe	NVLink / NVSwitch	SUE-T / TF1
典型交换芯片	PF320L	NVSwitch 5	Tomahawk F1
交换带宽（双向）	5.12 TB/s	7.2 TB/s	12.8 TB/s
最大端口数	80 Ports（X4）	72 Ports	512 Ports
网络拓扑	机内/机柜级	72-BlackWell 机柜级	512-XPU 单跳全互联 (集群级)
可靠性机制	ECC/奇偶校验、DPC、热插拔	硬件级重传，端到端ECC	LLR (链路重传), CBFC (信用流控)
优势	开放生态，支持异构处理器与内存直连、高效低时延	生态成熟，极低延迟。业界率先实现GPU通过200G Serdes互联产品落地	开放生态，支持超大带宽和更大规模扁平化集群组网

1.7 硬件集成方案

随着 GPU 间高速互联 Serdes 及吞吐带宽不断提升，同时单机柜功率也开始迈向 100kW 甚至 120kW 时代（如 NVL72 单柜约 120kW，未来也将朝着更大功率密度演进），传统风冷散热已接近物理极限。基础设施的工程化创新成为超节点能否稳定运行的决定性因素。

高速互联的挑战及演进

1. 传统电气互联的功耗墙与性能天花板

长期以来，基于铜线的电气互联是数据中心内部通信的核心。然而随着信号传输数据速率的不断攀升，这一传统技术正面临着根本性的“功耗-性能墙”。当 SerDes 速率攀升至 224G PAM-4 甚至更高的 448G 时，铜走线的物理损耗由于趋肤效应，呈指数级增加，信号衰减问题将变得异常严峻和难以逾越。

为了补偿这种严重的信号衰减，传统的可插拔光模块（或者相应 Retimer 驱动芯片等器件）必须集成数字信号处理器（DSP）进行信号补偿。其虽然能够维持信号完整性，但也伴随着大量供电损耗。在未来的高速网络交换中，I/O 功耗甚至可能超过处理器核心本身的功耗。这无疑形成了一个无法避开的话题：为提升带宽而增加的功耗，反过来限制了系统的整体扩展能力，以及由此带来能效瓶颈。

研究人员运用 Roofline 模型分析，为这一瓶颈提供了有力的数据证明。模型揭示，当前主流的大语言模型工作负载，如最新发布的 GPT5 和 Llama 模型，实际上是带宽受限而非计算受限。如图所示，光学互联将极大提升通信带宽（相较于以 NVLink 为代表的现有技术），从而提升大语言模型整体性能天花板。

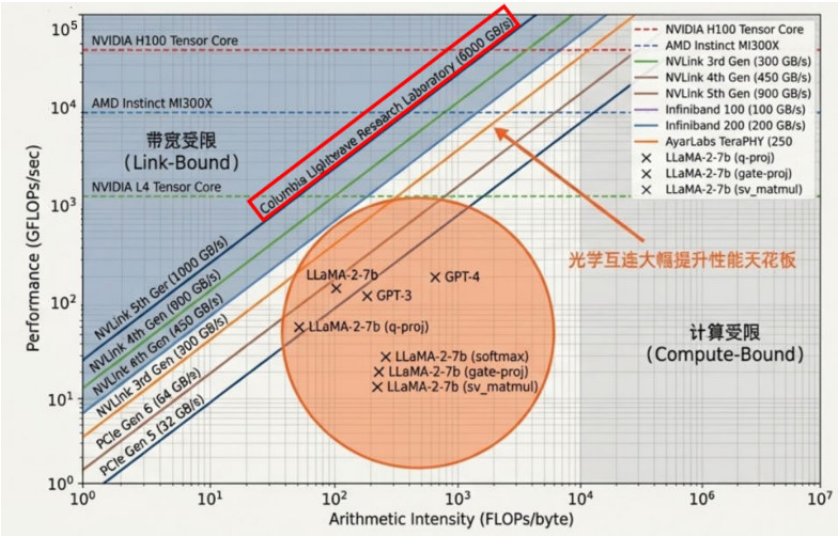


图28 算力密度与性能瓶颈的 RoofLine 模型 [33]

电气互联在物理定律面前已显现出其性能瓶颈。正是这一根本的物理极限，正推动着整个行业向一场革命性的技术转变迈进——将光学器件以前所未有的紧密度集成到芯片封装之中，从而开启一个全新的光互联时代。

不过，现阶段电互联作为芯片可靠通信的载体，依然承载着超节点互联各种工程落地实现，未来随着光互联技术的逐步成熟，电互联和光互联也必将在各自领域相辅相成，共同推进超节点高速互联技术进一步向前演进。

2. 现阶段超节点集群电互联工程实现

在高性能数据中心架构中，信号传输距离与能效，对互联技术路线之间的核心制约关系。

- 芯片级与短距互联：高速高密 SerDes 与先进封装的博弈。

如下图所示，越接近芯片内部（例如 Die-to-Die 或 Chip-to-Chip 级），对极高能效（<2 pJ/bit）和超高带宽的依赖越强。在高速传输速率及高密 layout 情况下，高频信号的介质损耗和串扰急剧增加，传统的板材和走线技术已经无法支撑这一高密的信号传输。因此，在微距尺度下，必须依赖 2.5D/3D 等先进封装技术来维系高速接口的信号完整性。

- 跨柜长距互联：光互联的必然选择。

  当传输距离拉长至跨多机柜或跨节点集群（数十米至数百米）时，高速 Serdes 下电信号无法克服急剧增加的损耗。因此，跨柜的长距离高带宽互联只能通过光互联（传统光模块，未来通过共封装光拉远）得以实现。光互联虽然在光电转换端存在一定的功耗损耗，但在长距离传输中展现出了无可替代的优势和带宽扩展性。

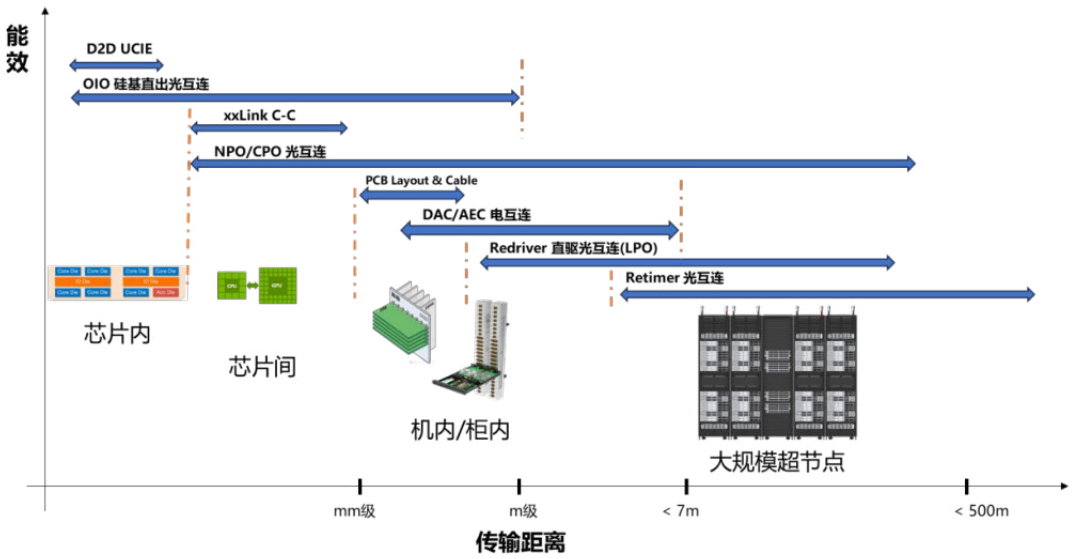


图29 各种互联技术的传输距离以及能效对比示意图

- 整机工程与板级互联：112G/224G 时代的电互联创新。

  在整机工程交付领域，跨芯片到跨主板的连接是核心应用场景。工程实现主要通过 PCB Layout、板内线缆（Flyover Cable）或板间电缆来实现系统级的互联互通。

  以当前主流的超节点互联架构为例，SerDes 信号正处于 112GT/s 的普及期，并向未来 224GT/s 的技术关口迈进。在这个频率下，即便采用超低损耗的 PCB 板材，电信号的传输距离也被极度压缩。为了突破传统背板的物理瓶颈，业界在电互联技术上也衍生出了多种创新的高阶板材背板或者无背板直连解决方案：

  - **Cable Tray 铜互联解决方案：**利用高速裸线缆替代 PCB 走线，大幅降低信号衰减，实现柜内计算与交换节点见可靠互联。

图30 Cable Tray 组网方案示意图

- **优势：**机柜液冷和电源盲插设计更容易实现；机柜深度要求低，突破背板物理尺寸的限制。
- **劣势：**长期维护成本高，线缆故障率随密度提升而上升；未来 Serdes 突破 400GT/s，演进上难以实现可靠连接。
- **典型应用：**NVIDIA NVL72 超节点。

- **正交背板互联解决方案：**优化系统风道与布线，缩短信号跨板距离的同时，利用高阶板材实现板间高速互联。

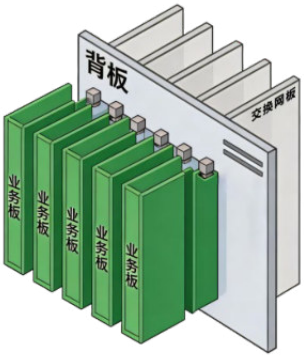


图31 整机背板组网方案示意图

- **优势：**相比 Cable Tray 成本降低；大幅减少外部线缆数量，释放空间；大幅度缩短传输路径，传输损耗低，符合 AI 大模型对带宽和延迟的极致要求；可维护性好，支持连接器级别故障定位以及节点板卡维护，Cable Tray 需要整体更换。
- **劣势：**定制化设计 PCB 背板，量产难度大。节点的深度被压缩，协调空间成为难题，研发成本高。
- **典型应用：**NVIDIA Rubin Ultra NVL144。

- **正交直连解决方案：**移除背板，让线卡与交换网板直接正交对接。

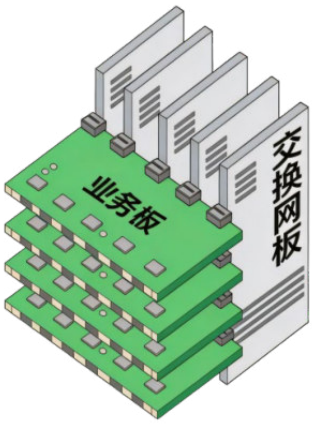


图32 正交直连组网方案示意图

- **优势：**彻底消除背板损耗，适用于追求极致性能场景。
- **劣势：**系统复杂度高，机柜架构和节点单板基于不同方案需重新设计，硬件设计可复用性差，研发成本高。
- **典型应用：**中兴超节点。

以上三种超节点高速信号电互联，在业界都有工程实现案例，但是从未来演进以及通用解耦设计角度，正交直连以及正交背板方案侧重点主要在于解决极大带宽和高速互联速率下的端到端电气信号可靠性上，在架构设计的通用性上存在一定瓶颈。而 Cable Tray 的方案，虽然设计兼容性上更优，但未来随着 Serdes 速率以及高速带宽密度的进一步提升，系统的可维护性以及可演进性还需要更多工程领域技术创新。

3. 光互联工程实现技术探索

解决互联瓶颈的核心，是在物理空间上将光学器件与 GPU 拉近。距离越短，电气链路的损耗就越低，系统的整体能效也就越高。从当前主流的可插拔模块，到近封装光学（NPO）、共封装光学（CPO），再到革命性的封装内光 I/O（OIO）[34,35]，未来将逐渐成为光学集成的技术演进路线。

- 可插拔光模块 (Pluggable Optics)

  当前数据中心最主流的方案，光模块通过连接器插入到交换机或服务器的前面板。

  核心价值与局限：标准化的接口和现场可维护。然而，随着速率提升，其功耗急剧增加，并且面板物理空间限制了端口密度，成为带宽扩展设计的瓶颈。

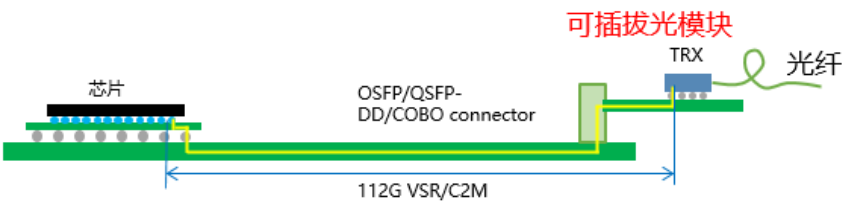


图33 可插拔光模块互联示意图

- **近封装光学 (Near-Packaged Optics, NPO)**

  NPO 作为光互联技术演进路径上的重要中间形态，将光引擎从传统的前面板移至主板上，通过 Socket 插座或直接贴装的方式，紧邻交换芯片或 xPU 等核心组件放置。

  核心价值：NPO 在物理空间上将光学器件与主芯片拉近。通过缩短物理互联距离，实现低损耗的电气互联条件下，通过光互联拉远。NPO 保持了光引擎与核心芯片的独立封装，不仅降低了制造工艺门槛和高功耗带来的散热挑战，也规避了封装制造低良率的风险，是当前兼顾性能与工程可实现性的可行方案。

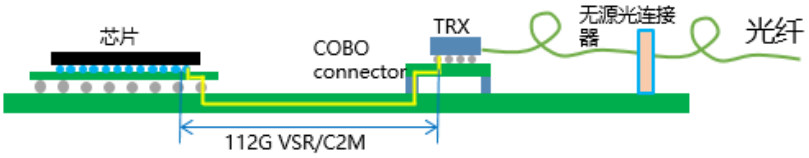


图34 NPO 互联示意图

- **共封装光学 (Co-Packaged Optics, CPO)**

  CPO 是一种先进的封装技术，它将光学引擎与芯片集成在同一个基板（Substrate）上。

  核心价值：其主要目标是通过将光学引擎与芯片并排放置，极大地缩短电信号传输路径。这一变革直接规避了可插拔光模块方案中为补偿长距离信号衰减而必需的高功耗 DSP，从而显著降低了功耗，提升高速信号传输质量。

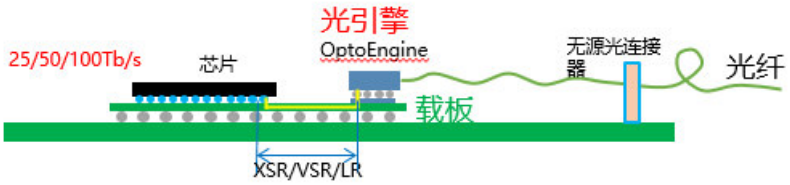


图35 CPO 互联示意图

- **封装内光 I/O (In-Package Optical I/O, OIO)**

  OIO 是一种基于芯粒（Chiplet）的光互联解决方案，它将光学 I/O 芯粒直接集成在芯片封装内部。

  核心价值：OIO 专为高密度、低延迟芯粒间互联而设计。其目标是实现与封装内电气互联（如 UCIe）相当的带宽密度、能效和延迟，同时具备光信号长距离传输的独特优势。

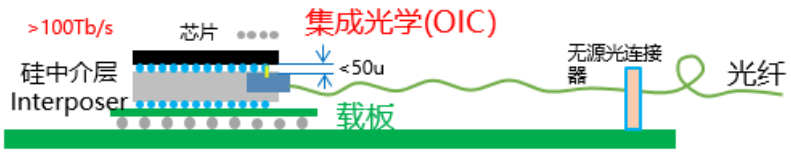


图36 OIO 互联示意图

4. 未来展望

在下一代智算中心超节点集群的演进中，基于铜缆的电气互联与近芯片光互联（CPO/OIO）必将相辅相成、互为补充。

随着 AI 大模型规模的爆炸式增长，集群的数据互联能力已成为决定整体性能的关键瓶颈。当系统网络向 224G 甚至 448G SerDes 演进时，传统电气互联受物理定律制约，严重的信号衰减以及补偿损耗所需的 DSP 引入了难以逾越的 “功耗–性能墙”。因此，在物理空间上拉近光学器件与计算核心距离的共封装技术，成为了突破大规模组网带宽密度和能效极限的必然选择。

然而，现阶段光互联的大规模工程化部署仍面临显著瓶颈：

- **热管理与封装制约：**对热高度敏感的光子集成电路与极高功耗的 GPU 共封装会产生严重的热串扰，这使得液冷成为未来超节点机柜强制性的物理基建要求。
- **制造成熟度限制：**需要加速 2.5D 和 3D 异构集成工艺的成熟，特别是高精度的光纤对准、键合和封装技术，目前成本高昂且良率有待提升。
- **生态与运维博弈：**深度集成的光学元件增加了底层硬件故障排查与维护的复杂性，且存在开放生态标准与定制化封闭系统之间的路线博弈。

未来的硬件架构将基于传输物理距离进行重塑：在芯片封装内部或极短距离的板级节点内，电气互联继续发挥其低成本与亚纳秒级极低延迟的优势；而在跨节点、集群大规模组网中，光互联主导数据交换。两者深度协同，共同支撑智算中心算力的持续演进。

供电系统的发展及演进

1. 服务器电源演进方向

AI 推动 GPU 服务器功耗快速上升，供电技术随之快速发展，业界以往的分布式供电架构已遭遇瓶颈，集中式整机柜供电迎来技术发展热潮。

图37 数据中心柜内供电部件规格演进

与此同时，随着单机柜功率越来越大，供电电压正在从传统的低压交流电向高压直流电演变。

图38 数据中心柜内供电方案规格演进

2. AI 热潮推动整机柜功耗激增，相比分布式，集中式供电优势开始体现

突破束缚与瓶颈

- 提升功率：电源模块集中在独立机架中，电源物理空间更大，更适用高功耗设备需求。
- 节省空间：相比分布式应用显著节省机柜电源空间，为高密节点留出宝贵空间。
- 优化冗余：更利于 ATS（Automatic Transfer Switch）电源设计，N+1 冗余应用，降低冗余电源数量。
- 降低通流：母线电压从 12V 提升到 54V，电流为之前的 1/4，解决通流瓶颈。
- 重构配电：输入采用工业连接器，突破传统 PDU（Power Distribution Unit）配电功率限制;支持三相交流输入，避免三相不平衡。

实现节能与降本

- 节能：更利于高效率电源设计，实际负载点更优，节省能源。
- 降本：减少分布电源的冗余数量，节省电费和维护成本。

简化散热与运维

- 简化散热：电源独立风道，与系统散热解耦，简化散热难度和串扰，适合液冷机柜发展。
- 智能运维：电源模块统一部署在 PowerShelf 中，与服务器节点解耦，远程管理，智能监控、调配和使用。

存在挑战与困难

- 初期投资高：需评估长期收益与初期成本的平衡。
- 依赖新基建：机房配电架构、散热架构需要重新规划、建设或改造。

在以 GPU 计算为主的高功耗数据中心，PowerShelf 方案通过集中化、模块化的设计，在能效、运维、功率密度上体现了显著优势。

整机柜 PowerShelf 正在快速发展，新架构、新需求、新方案不断涌现

- 单输入电源+BBU 后备电池（ORV3 规范）
- ATS 双输入电源（国内+海外 Oracle）
- 21 英寸机柜，电源 73.5mm 宽（ORV3 规范）
- 19 英寸机柜，电源 68mm 宽（英伟达）
- 1U Shelf
- 2U Shelf

图39 Powershelf 电源规格演进

800V 解决方案的成熟度持续提升并日趋完善

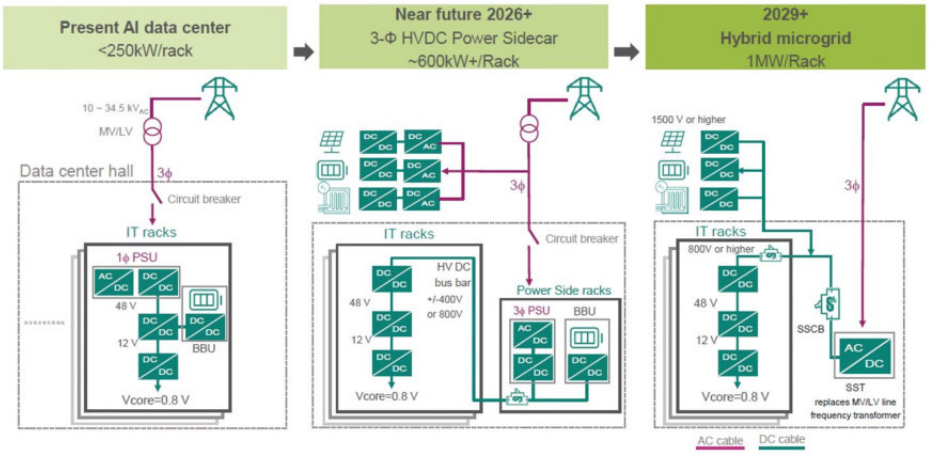


图40 800V 高压直流供电技术演进

当前通用解决方案依托 800V PowerShelf 转 54V 实现阶段性过渡，随着技术的进一步发展和演进，未来也可能出现采用电源板 PDB（Power Distribution Board）将 800V 直接引入计算节点的实现方案。

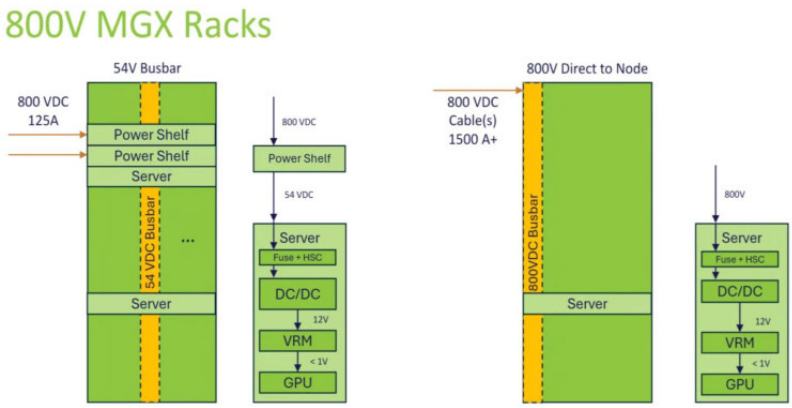


图41 800V 高压直流柜内供电方案演进

800V PowerShelf

- 19 英寸 PowerShelf：功率，90KW，单 PSU 功率，15KW
- 21 英寸 PowerShelf：功率，108KW/120KW，单 PSU 功率，18KW/20KW

分布式供电系统的 PDB 方案，在板内通过电源砖模块将 800V 转为 54V 或者 12V。

在分布式供电系统的 PDB 板上，配置了输入电压为 800V DC 的砖型 DC/DC 电源变换模块，该模块采用 16:1 的电压变换比设计;当前技术规划中，半砖封装的模块额定输出功率可达 6kW，全砖封装的模块额定输出功率可达 10kW。

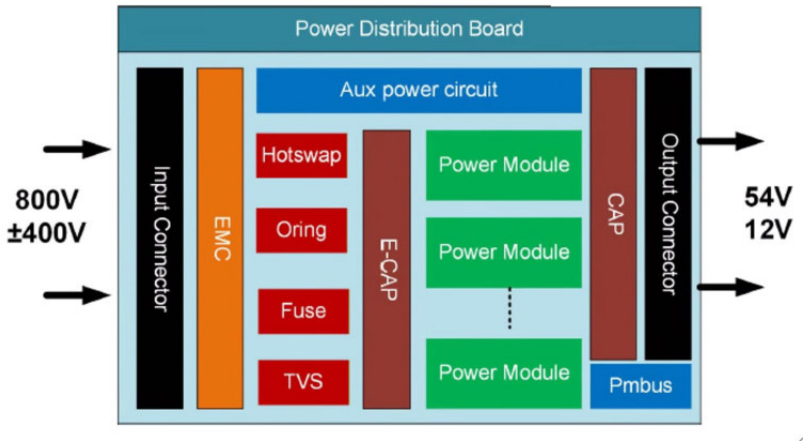


图42 分布式供电系统高压直流电的转换示意图

板级供电发展及演进

1. 当前板级供电架构

随着 AI 需求日益增长，GPU 作为 AI 时代的核心部件技术不断迭代，功耗不断增长，目前已达 1000W+；板级供电引入中间总线 48V 供电架构，大幅降低母线传输损耗，是应对千瓦级功率的核心趋势。为应对功率密度的增加以及 PCB 的空间限制，板级电源方案由板载分立方案演进为集成模块方案。

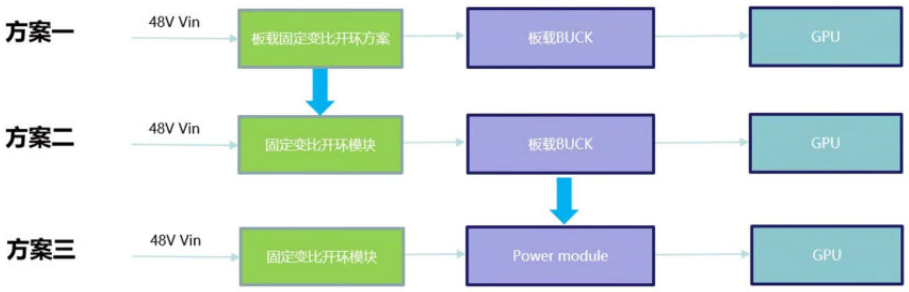
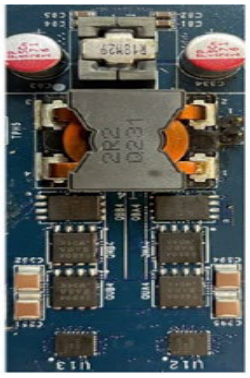


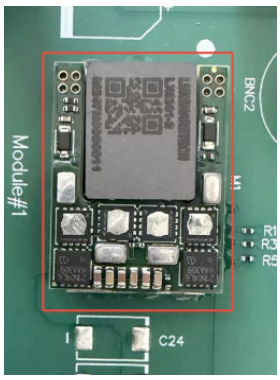
图43 GPU 供电方案技术对比示意图

2. 固定变比开环分立方案

开环分立方案是一种基于分立元件构建的电源控制方案，其核心特点是控制环路中不存在反馈机制。即系统根据预设的输入信号直接控制电源的输出，而不根据实际输出结果进行调整。



分立方案



开环模块

3. 固定变比开环模块

对数据中心、GPU 加速卡等功率密度要求高、开发周期短、批量一致性要求高的场景，相较于分立方案，固定变比开环砖模块是首选方案，其在多指标对比上更优：

表6 固定变比的两种供电方案对比

指标	固定变比模块	固定变比分立方案
功率密度	极高	高
BOM成本	高	低
开发周期	短	长
效率	高	中高
EMI	优	中
可靠性	高	中

当前 4:1 开环模块是中间总线架构(1BA)中的主流方案;面对 GPU 等高性能计算硬件持续增长的功率需求，业界正积极开发 8:1、10:1 乃至更高变比的电源转换方案，这些技术被认为是应对未来挑战的更优选择。

4. Power module 发展

当前电源模块(Power Block)的性能提升正迎来关键突破——电流承载能力已从传统的 1A/mm^2 跃升至 2A/mm^2，同时封装体积持续微型化，功率密度也随之显著攀升。在性能持续跃升的 GPU 架构中，对高效、紧凑的供电解决方案的需求日益迫切。新一代电源模块通过提升电流密度与缩小封装，恰好回应了这一需求：它能够在更小的物理空间内，以更高效率输送稳定而强大的电力，直接助力 GPU 突破性能与能效瓶颈。随着

模块化、集成化设计成为趋势，此类高密度电源方案为 GPU 的芯片布局释放出更多空间，使得运算单元、内存与其他关键组件得以更紧密排布，进一步优化整体系统架构。

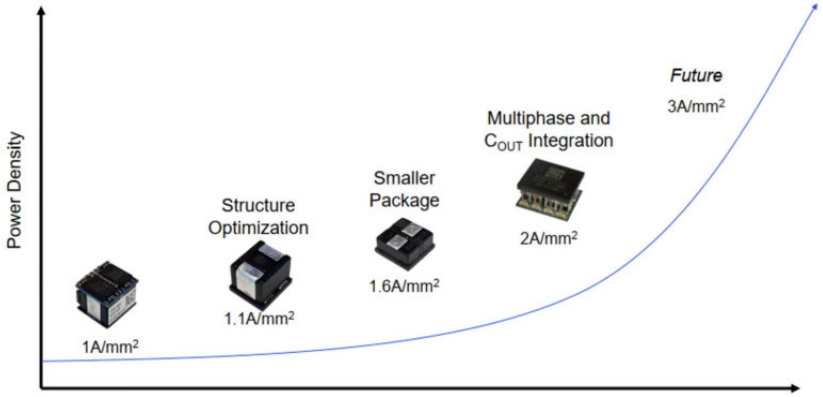


图44 电源模组的技术演进

现阶段两相 Power Block 为主流应用，而主流厂商 MPS 新一代方案已将集成度提升至 4 相(见下图)。这意味着，在单个高度优化的封装内，可整合更多相的 MOSFET、驱动器和电感。其直接优势在于：

功率密度倍增：单位面积或体积内可承载的电流能力大幅提升。

封装极致缩小：显著节省 PCB 板面积，为 GPU 核心或显存留出更多布局空间。

性能与效率优化：集成设计减少了寄生参数，提升了开关频率与响应速度，同时降低了功率损耗。

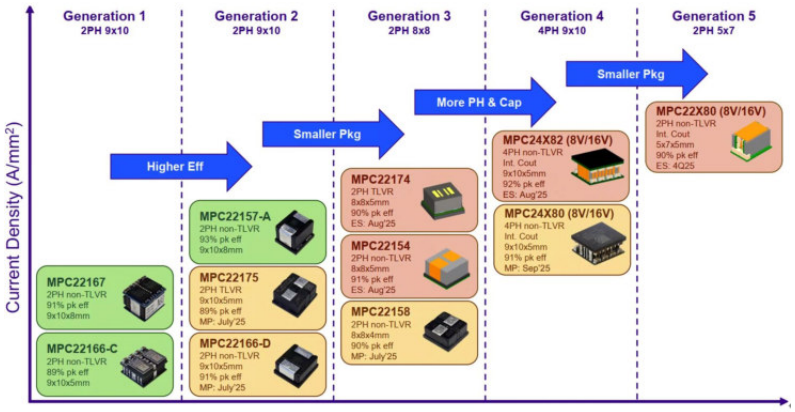


图45 常见 Power Block 电源的方案演进

5. 垂直供电

由于 GPU 电流不断上升，计算功率的散热会被 PDN (Power Delivery Network) 损耗限制，垂直供电无疑是未来供电方案的趋势。如图，现阶段市场预研以模块背面垂直供电为主，未来随这 GPU 计算能力进一步加强，供电电流会指数级上升，为进一步减小 PDN，可能会将 VR module 嵌入到 GPU 的基板中。

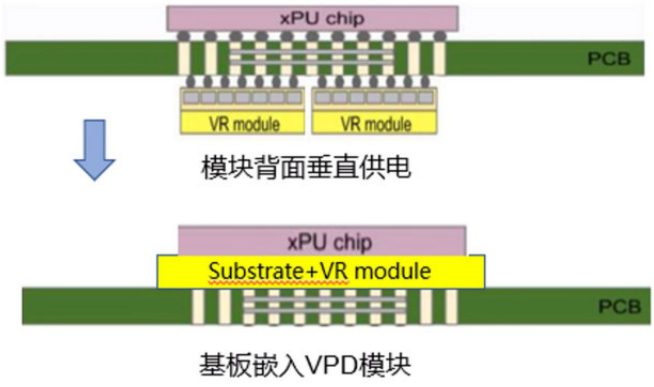


图46 垂直供电技术示意图

垂直供电优势

- 供电效率更高：垂直供电无较长的电源传输路径，极大减小了 PCB 上的铜损，这对 AI/GPU 大电流场景收益客观。
- ESL 更低，更高的环路带宽响应：更短的路径意味着更低的寄生参数，可得到更快的瞬态响应能力。
- 节省 PCB 空间：将电源部分挪到了芯片下方，节省出 PCB 空间，也降低了板级布线的难度。

垂直供电的瓶颈

- 散热困难：既要给上面主芯片散热，又要兼顾背面的垂直供电 module 散热，热设计面临双重挑战。
- 结构限制：垂直供电无疑会增加 PCB 的背面厚度，而 PCIe 形态，OAM 形态均对背面高度有限制，这无疑对结构提出了更高要求。

风冷设计以及演进瓶颈

1. 风冷散热技术的演进历程与应对策略

在过去十年间，数据中心计算经历了从通用计算向异构加速计算的深刻转变，GPU 芯片的广泛部署成为关键驱动力。伴随这一进程，GPU 的热设计功耗持续攀升：从 2016 年英伟达 P100 的 300W，到 2020 年 A100 的 400W，再到 2022 年 H100 的 700W，芯片功率密度增速显著超越传统 CPU。

为应对这一趋势，传统框式 GPU 服务器普遍采用以空间置换散热的设计思路，典型物理高度为 4U 至 8U，从而为散热系统留出关键设计裕度。在此空间内，风冷方案通过多重强化手段实现：

高性能风扇系统：采用 80mm 及以上规格的大尺寸风扇或多风扇阵列，以产生高静压与大风量，强制气流穿越高风阻散热模组。

大型化散热模组：配备 2U 至 4U 高度的强化散热器，其核心由传统热管阵列演进为覆盖全芯片的均热板，并进一步发展为 3D-VC 架构——通过三维堆叠的均热板与热管复合结构，将导热路径从二维平面扩展至三维空间，显著降低芯片结到散热鳍片的热阻。

通过上述极致工程优化，传统风冷方案在特定阶段内艰难维系了对400W至750W级别单芯片的散热可行性。然而，该方案亦伴随显著代价：高噪音水平、高风扇功耗占比，且因结构强度与振动要求导致服务器重量与成本大幅上升。这实质上是一种趋近物理极限的“强制散热”模式，其可持续性已面临严峻挑战。

2. 超节点架构：风冷散热的能力天花板与双重困境

超节点整机柜架构通过资源池化、模块化与高密度集成，追求极致的能效与算力密度。该架构对散热系统提出两类截然不同却同样严峻的挑战，共同标志着风冷方案的能力几乎已达上限。

- **困境一：高功率多卡节点的“级联散热极限”**

  部分超节点设计允许单个计算节点占据 4U 至 8U 高度，以容纳多达 8 颗高性能 GPU。即便在此相对充裕的空间内，面对下一代 GPU 功率向 1500W 以上跃升(甚至如业界预测的 Rubin 架构 GPU 可能达到 2300W，甚至 Rubin Ultra 或高达 3600W 级别)，风冷方案亦陷入理论困境。

  1500W 单芯片的风冷散热已经逼近物理极限，且叠加核心矛盾“级联散热效应”。在密闭节点内，多个超高功率热源线性排列，下游散热器必然吸入已被上游加热的空气。即便采用最优的分区风道设计，气流的温升累积仍无法消除。当单芯片功耗超过 1500W 且节点内级联排列 2 颗芯片时，即使该区域级联风量达 200CFM，下游芯片进风温度可能较环境温度升高 13° C 以上。为确保芯片结温处于安全范围，所需风量与静压将驱使风扇功耗呈指数增长，导致系统能效比急剧下降，从工程与经济性角度已几乎不具备可行性。

- **困境二：1U 超高密度节点的“空间物理极限”**

  超节点演进的另一路径是追求极致空间效率，将计算节点压缩至 1U 高度。在此极端约束下，即便应对当前普遍超过 500W 的主流高性能 GPU，风冷方案也已完全丧失物理可行性。

  1U 高度仅 44.45 毫米，扣除 PCB、结构件及必要间隙后，可用于散热器的垂直空间通常不足 30 毫米。在此条件下：

  - **有效散热面积严重不足：**散热鳍片高度与总面积被极度压缩，无法提供千瓦级散热所需的最小换热面积。
  - **空气动力性能失效：**薄型风扇在有限厚度内无法产生穿透高密度微鳍片阵列所需的静压和大风量，气流组织趋于失效。

3. 结论：风冷时代面临终结与必然的技术转移

综上所述，超节点架构的发展已从两个维度将风冷散热推至不可逾越的瓶颈：

在高功率多卡节点中，级联散热效应与风扇功耗的指数增长构成了其经济性与工程可行性的天花板。在 1U 超高密度节点中，物理空间的绝对限制构成了其基础热传导定律上的天花板。

面对 GPU 功率向千瓦级乃至更高层级迈进的确切趋势，风冷方案在服务器散热领域，特别是超节点与高性能计算场景中，已近乎完成其历史使命。散热技术的范式转移近乎势在必行，液冷技术凭借其更高的比热容与导热系数，成为突破上述双重困境、支撑未来算力持续发展的唯一工程可行路径。

液冷技术发展演进

1. 冷板式液冷:超节点架构的主流散热方案

面对风冷方案在超节点架构中遭遇的物理极限，液冷技术凭借其介质固有的高热容与高导热特性，成为必然的替代方案。其中，冷板式液冷，因其在性能、兼容性与可维护性之间的卓越平衡，已发展成为当前超节点整机柜架构中绝对主流的散热解决方案。

卓越的单芯片解热能力：冷板配合优化的微通道或歧管流道设计、更高性能的导热材料如相变材料、液态金属等，可稳定支持 1500W 以上的芯片功耗，对于采用多芯片/多 die 封装(扩大芯片面积)的 NVIDIA-Rubin 架构 GPU，可支持 2300-3600W 级别的散热需求。

与超节点架构的高度兼容：冷板式液冷本质上是对服务器内部散热模块的替换，而非对计算节点或整机柜架构的颠覆性重构。其冷却回路(CDU、管路、快接接头)能够完美适配超节点的模块化、高密度部署理念。

渐进式的全液冷演进路径：该方案天然支持从“混合冷却”(CPU/GPU 等核心高热器件液冷，其余部件风冷)向“全液冷”(所有发热元件均由液冷覆盖)的演进。通过为内存、电压调节模块(Voltage Regulator Module，VRM)、光模块等次级热源设计集成式或辅助冷板，可逐步消除系统内所有风扇，最终实现机房级的极致 PUE。

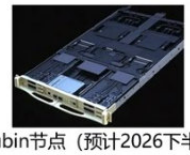
2. 产业生态的成熟与标准化推动

冷板液冷方案的普及，离不开强劲的产业需求牵引与快速成熟的生态系统建设。以英伟达为代表的芯片与系统厂商，通过其产品路线图与参考设计，为行业树立了清晰的液冷化标杆。

产品化引领：从 A100 的液冷选项，到 H100 及 GH200 Grace Hopper 超级芯片对液冷的强力推荐，再到最新发布的 GB200 NVL72 平台及规划中的 GB300、Rubin 架构，英伟达已在其高端数据中心产品线中全面拥抱冷板式液冷。这些系统级设计不仅验证了液冷的性能，更定义了冷板、连接器、流量分配单元等关键组件的规格，推动了供应链的规模化和成本下降。


GB200节点（已量产）


GB300节点（已量产）


Rubin节点（预计2026下半年）

	量产中		2026下半年~2027	
	GB200	GB300	Rubin	Rubin Ultra
整柜GPU数量 GPU功耗 GB300	72 GPU 1200W ~1	72 GPU 1400W ~120KW	~144 GPU 2300W ~400KW	~144 GPU 3600W ~600KW
液冷方案	冷板式液冷+风冷	冷板式液冷 全液冷	冷板式液冷 全液冷	冷板式液冷 全液冷
节点内线缆方案	线缆方案	线缆方案	无线缆	无线缆
NV趋势：冷板液冷、全液冷、节点模块化设计、无线缆、整机柜解决方案				

图47 NVIDIA 产品液冷技术演进示意图

标准化进程加速：开放计算项目、开放数据中心委员会(Open Data Center Committee，ODCC)等产业组织正积极推动冷板液冷组件与接口的标准化工作，涵盖冷板尺寸与安装、盲插式快接接头、冷却液特性以及监控管理协议等。标准化是降低部署复杂度、实现多厂商互操作性的关键，进一步巩固了冷板液冷的主流地位。

综上所述，在强大的技术优势、与现有架构的兼容性以及日益完善的产业生态共同作用下，冷板式液冷已成为支撑当前及下一代超节点整机柜算力密度的基石性散热技术。

3. 冷板式液冷散热能力技术演进与优化方向

面对千瓦级 CPU/GPU 芯片及其非均匀热源带来的挑战，冷板式液冷技术的优化正尝试沿着一条从宏观流场调控到微观材料集成的路径纵深发展。其演进方向可系统性地划分为五个关键方向，共同尝试推动散热能力迈向新高度。

基于热源分布的针对性流道优化

此方向的核心在于打破均一化设计，使冷却液的分配与芯片的真实热负荷精确匹配。

- **热源驱动设计：**利用芯片的详细热仿真或实测数据，通过计算流体动力学 (Computational Fluid Dynamics，CFD) 与拓扑优化算法，定制非均匀流道布局。高热区(如 CPU 核心、GPU 计算单元)下方布置高流量、高换热密度的密集流道;低热区则对应低流阻路径，从而实现全局温度均匀性与系统泵功的最优平衡。
- **分区与射流强化：**对于存在多个独立高热源(如 Chiplet 多芯粒)的场景，采用分区独立供液或嵌入式微射流阵列技术。冷却液可通过特定喷嘴或歧管直接高速冲击至每个热点区域，实现局部换热系数的大幅提升，这是应对极端局部热流的有效手段。

基础材料创新

此方向的核心在于突破传统金属的导热极限，并优化界面传热效率。

- **冷板本体材料：**研发重点集中于高导热复合材料，如掺金刚石铜、高定向热解石墨与金属的复合基板。这些材料在特定方向上的导热系数远超纯铜，能更快地将芯片热量横向扩散。
- **热界面材料：**TIM 是散热链路中的关键瓶颈。优化方向包括更高性能相变材料及液态金属的应用，目标是在低安装压力下实现极低的热阻，并保证长期可靠性。

基板均热技术强化

当芯片尺寸增大或存在多热源时，仅靠基础材料传导不足以避免局部过热。此方向为引入均热元件。

- **嵌入式均温板：**在冷板底座内部、紧贴芯片接触区域的下方，集成均温板。其内部的相变工质能实现近乎等温的二维平面超快热扩散，将集中热源迅速摊平，为上层对流换热提供均匀的热负载。

微通道结构深化与系统化

这是提升对流换热系数的核心战场，焦点从“有无"转向“最优”。

- **多尺度与异形通道设计：**在靠近芯片的高热区采用超密微通道(<100um)以最大化换热面积;在集液区则过渡为低流阻宏通道，以平衡压降。同时，通道截面形状向梯形、滴形等减阻扰流形貌演进。
- **分级分流与拓扑优化：**基于芯片实际热源，通过算法驱动设计非均匀、仿生流道拓扑，实现冷却液的智能按需分配。

微通道与芯片封装的集成

此为前沿探索方向，旨在消除所有中间热阻，将散热器作为芯片的一部分进行设计。

- **近芯片冷却/直接-on-chip 冷却：**将精密的微通道冷板作为芯片封装的上盖或中介层，通过先进工艺与芯片直接集成。冷却液在距晶体管仅毫米甚至微米级的通道内流动，实现最短传热路径和极限散热能力。
- **向 3D-IC 与异构集成的融合：**在未来 3D 堆叠芯片中，微通道冷却层可能被设计为主动式夹层，嵌入在不同芯片层之间，直接对堆叠内部产生的“体积热”进行提取。

总结而言，冷板技术的演进呈现清晰特征:从基于热源的智能流场设计，到材料的物理属性改良，再到基板内部的相变均热强化，进而到流道结构的系统化设计，以及尝试迈向与芯片封装一体化的深度融合。目前这些优化方向部分还处于研究和探索阶段，必须解决新材料物理属性不稳定、一体式均热板冷板工艺不稳定、微通道导致的高流阻和易堵塞等问题，才能让这些技术真正落地量产。

其它液冷技术路径的现状与挑战

尽管冷板式液冷占据主导，但其他液冷技术路径也在持续发展中，各自具有独特的潜力与当前面临的技术或工程挑战。

图48 各种液冷技术对比示意图

小结

本章节主要是围绕超节点核心的硬件技术架构展开：

- **宏观定义：**首先明确超节点是为应对大模型时代“内存墙”与“通信墙”挑战而诞生的新计算范式，确立了以 GPU 为中心、基于 Scale-Up（南向组网）技术实现资源深度池化的架构基调。
- **物理形态：**阐述了为突破传统互联瓶颈，物理载体从通用服务器向整机柜超节点（Rack as a Computer）和整机多框超节点（Disaggregated Chassis）演进的必然趋势，实现了计算单元在物理空间上的高密集成与逻辑资源统一。
- **互联与拓扑：**针对高带宽、低时延的极致互联需求，对比分析了业界主流的 CLOS、Torus、DragonFly 等拓扑结构，并探讨了不同组网架构在转发路径、对分带宽与 AI 业务所需要的权衡，并基于业界典型的超节点集群背后的组网考量。
- **微观层面：**深入芯片内部，揭示了先进封装技术（CoWoS/Chiplet）如何通过算力芯粒与 HBM 的异构集成突破工艺限制，同时介绍了从混合精度到微缩放技术路线的演进路径，从算法协同角度打破超大规模参数训练和推理的算力需求瓶颈。

- **工程底座：**最后回归工程实现，指出随着芯片功率密度逼近物理极限（如单芯片迈向数千瓦级），液冷技术已成为不可逆转的必然选择，它将与新一代供电技术共同构成支撑超节点集群稳定、高效运行的坚实底座。与此同时，GPU 的互联 serdes 速率的不断演进，未来光互联方案也陆续开始展露头角。

随着半导体先进制程的演进红利逐渐放缓，单芯片算力的线性增长已逐渐触及物理极限（算力墙），且算力的持续演进，也不可避免地伴随着数据吞吐的严重滞后（内存墙）以及极端物理条件下的能效危机（功耗墙与散热墙）。

面对错综复杂的瓶颈，算力增长的未来不再是 GPU 芯片的单点暴力堆叠，超节点技术未来发展必然走向软硬件一体的深度优化，并通过基于高速互联的异构算力的全局协同，实现性能与能效的进一步突破。

阶段	时代	核心技术	单 GPU 单链路带宽	单节点/单机柜 GPU 数量	最大集群/超节点规模	集群/超节点总互联带宽	核心互联方案
1	Pascal/Volta	NVLink 1.0/NVLink 2.0，第一代GPU直连技术，摆脱PCIe转发瓶颈	NVLink 1.0：80GB/s；NVLink 2.0：300GB/s	8	8（单机柜上限）	约2.4TB/s（8卡全互联峰值）	节点内NVLink，节点间仅支持普通以太网/IB
2	Ampere (A100)	NVSwitch 2.0芯片，GPU专用交换架构，实现节点内无阻塞全互联	NVLink 3.0：600GB/s	16	128（第一代SuperPod集群）	约9.6TB/s（16卡节点峰值）	节点内NVLink 3.0，节点间InfiniBand HDR/NDR
3	Hopper (H100/GH200)	NVSwitch 3.0，NVLink-Net机柜级组网，打破单机箱物理边界	NVLink 4.0：900GB/s	32（单节点）	256（单机柜统一超节点）	约28.8TB/s（单机柜256卡峰值）	NVLink 4.0全柜直连，小范围替代IB，形成单一NVLink域
4	Blackwell (GB200)	NVLink 5.0 + NVLink-C2C芯粒互联，Grace+Blackwell超级芯片，全局统一内存	NVLink 5.0：1.8TB/s；C2C芯粒互联：1.8TB/s	72 / 单机柜	576（多柜组合SuperPod）	130TB/s（单机柜72卡超节点总带宽）	全局NVLink 5.0全柜无阻塞互联，跨柜InfiniBand Quantum-2
5	Rubin（Vera Rubin，下一代）	NVLink 6.0 + NVSwitch 6.0，HBM4高带宽内存，Vera CPU+Rubin GPU芯粒，Kyber机架架构，ConnectX-9 SuperNIC	NVLink 6.0：3.6TB/s；C2C互联带宽同步升级	NVL72：72 / 单机柜；NVL144：144 / 单机柜	NVL72：576；NVL144：1152（翻倍扩容超节点）	260TB/s（NVL72单机柜）；520TB/s（NVL144单机柜）	全局NVLink 6.0，单链路带宽翻倍，跨柜Quantum-X800 IB/Spectrum-6，全光低延迟互联

2 Scale up协议核心技术解析

Scale UP 总线大部分采用物理层、链路层、事务层三层架构，部分协议（如 UB）在此基础上增加了网络层和传输层，并可以基于网络环境灵活启用关闭。总线的分层设计是应对设备互联需求升级与技术复杂度的必然选择。早期 ISA、EISA 等总线采用单一架构，信号传输、数据校验、逻辑交互功能混编，难以适配从 MB/s 到 GB/s 级的带宽跃升，在可靠性与兼容性上瓶颈凸显，可参见本书 PCIe 章节。

为破解困境，分层设计应运而生：物理层定义信号规范、传输介质等底层基础，解决数据如何传的问题；链路层通过 CRC 校验、流量控制保障数据完整性，解决数据准确传送的问题；事务层明确请求 / 响应协议、地址映射等逻辑，解决传给谁和传什么的问题。

这种架构的核心价值在于解耦：各层可独立迭代升级，故障精准定位，同时兼容异构硬件，让总线得以支撑从单机外设到超节点 Scale up 通信场景，成为现代计算系统高效运行的关键基础。

以下就 Scale UP 总线三层架构的功能和发展进行剖析。

2.1 物理层及 SerDes 技术

物理层技术

总线物理层（PHY）是高速串行总线实现数据传输的核心硬件层，也是 SerDes 技术的核心载体，主流高速总线（PCIe、Ethernet、UCIe、CXL 等）均遵循分层解耦设计（如遵循 PCI-SIG/IEEE 规范）。其核心分为三层：PCS 层、PMA 层、PMD 层，部分架构会集成独立的 PLL/CDR 时钟模块。各层功能边界清晰、协同工作，既实现高速串行传输的核心需求，又能适配不同协议、不同物理介质的差异化要求，同时大幅降低协议升级和工艺迭代的研发成本。

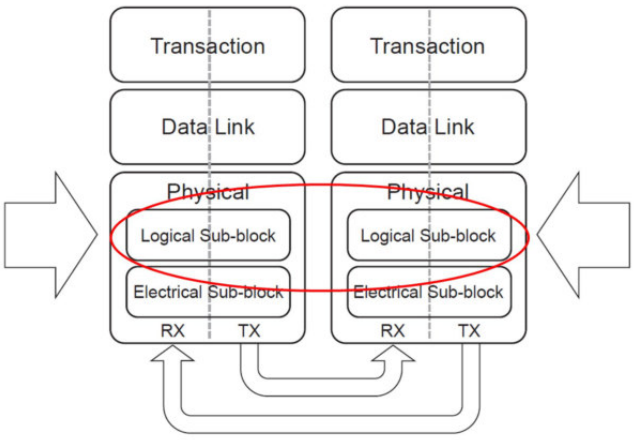


图49 物理层位置 [36]

表7 物理层各子层功能说明

层级 / 模块	核心定位	核心功能
PCS 层 物理编码子层	纯数字层，SerDes**协议数字核心**，对接上层协议（事务层 / MAC 层）	• 调制编码 / 加扰解扰、FEC • 串并 / 并串转换（SerDes 数字实现） • 帧同步 / 定界、多通道帧对齐 • 链路训练协商（LTSSM）、速率协商 • 多协议格式适配 / 切换
PMA 层 物理介质附加子层	数模混合层，**数字 – 模拟桥梁**，SerDes 信号优化核心，与协议解耦可复用	• CDR 时钟数据恢复，抵消时钟漂移 / 抖动 • DAC/ADC 数模 / 模数转换 • 自适应均衡（发送预加重 / 接收 CTLE/DFE） • 单端 – 差分信号转换，抗共模干扰
PMD 层 物理介质相关子层	纯模拟层，**物理介质直连接口**，功能与介质强绑定，不可跨介质复用	• 差分阻抗匹配，避免信号反射 • 发送端高速驱动放大、接收端低噪声放大（LNA） • 物理介质（PCB / 铜缆 / 光纤 / 同轴）适配 • EMC/EMI 优化，抑制电磁辐射
PLL/CDR 核心配套模块（时钟核心）	独立时钟模块，SerDes**高速传输节拍器**，部分架构将 CDR 归入 PMA 层	• PLL：生成低相位噪声高速参考时钟 • CDR：从接收信号中提取同步时钟，实现收发端时钟同步 • 抖动抑制 / 滤除，保证采样精度

数据发送的流程为：上层并行数据 –> PCS 层（编码 + 串化）–>高速串行数字流 –> PMA 层（数模转换 + 预加重 + 差分转换）–> 模拟差分信号 –>PMD 层（驱动放大 + 阻抗匹配）–> 物理介质（PCB / 铜缆 / 光纤）传输。接收端为反向流程。

物理层分层设计，如同其他协议的分层设计一样，松耦合的分层结构使各层都有独立进化的可能，其核心价值体现在如下方面：

1. 解耦数字与模拟设计，缩短研发周期，降低研发成本和技术风险

PCS 层（数字）与 PMA/PMD 层（数模 / 模拟）完全解耦，数字工程师可专注于协议编码和串并转换，模拟工程师可专注于信号完整性和介质适配；协议升级可仅需修改单一层级，如修改 PCS 层，PMA/PMD 层直接复用（PCIe 6.0 –> CXL 3.0），从而大幅降低研发周期和技术风险。

2. 适配多协议、多介质，提升 SerDes 复用性

同一套 PMA/PMD 层可搭配不同的 PCS 层，实现多协议兼容：如一款高速 SerDes PHY，可通过配置 PCS 层，同时支持 PCIe 6.0、CXL 3.0、UCIe 1.0 Standard PHY 模式；同一套 PCS/PMA 层，可通过更换 PMD 层，适配不同物理介质（如 PCB 背板、高速铜缆、光纤）。这也是 SerDes 技术能在主机内、数据中心、5G 等多场景应用的基础原因。

在 PCS 子层中，FEC（前向纠错）通过在发送端添加冗余校验比特（一般为 Reed-Solomon RS 编码）、在接收端利用纠错码算法自动检测并纠正一定数量的传输误码，从而在不重传的情况下提升通信可靠性。FEC 广泛应用于高速 SerDes 系统中，能显著降低误码率、提升链路可靠性并延长传输距离，但会引入额外延迟、增加功耗与带宽开销（冗余比特），并可能掩盖信道劣化问题。亦即更多的冗余位能带来更好的纠错能力，但同时也会引入更大的延迟和更大的带宽开销。因此根据应用场景需求，选择合适的 FEC 算法是各 Scale up 协议 SerDes 重点关注的问题。轻量级 FEC（如下图的 KR4）硬件实现简单，能有效消除字节级的传输错误，同时不引入过多时延，减少因错误导致的重传和等待。其纳秒级处理与 Scale up 的低延迟需求完美匹配，是唯一能在不破坏性能前提下提供的硬件可靠机制，基本为 Scale up 协议标配。

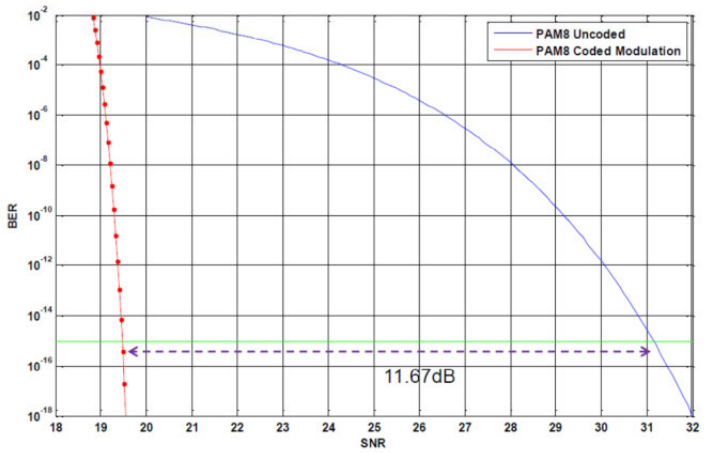


图50 轻量级 FEC（IEEE Draft）

PAM8 使用 802.3bj KR4 FEC 在误码率（BER）为 1E-15 时可提供 11.67dB 编码增益 [37]

SerDes 技术

SerDes（串行器/解串器）是物理层的核心载体，通过“并行转串行”发送、“串行转并行”接收的双向转换，解决高速传输中的引脚占用、时序干扰、距离限制等痛点，是高速数据传输领域不可或缺的数据大动脉。其技术重点围绕架构设计、调制方式、时钟同步等核心模块展开，当前技术要求聚焦于高速率、低功耗、高可靠几大方向。

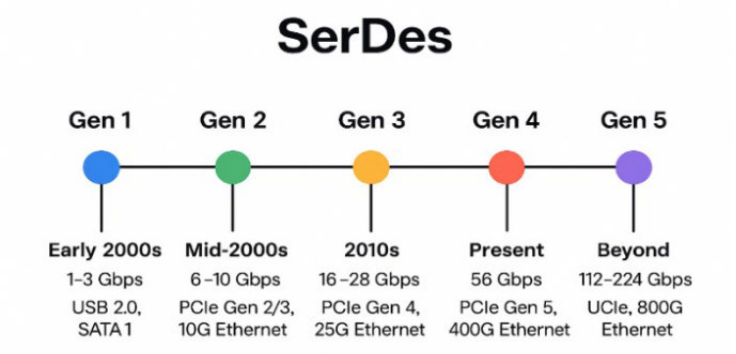


图51 SemiWiKi 的 SerDes 分代 [38]

技术产生背景与核心问题

1. 技术产生背景

SerDes 技术诞生于在 20 世纪 80 年代中期的光传输网络（SONET），用于长距离数字信号传输。将该技术应用于主机信号互联的标志性事件是 PCIe 1.0（2003）标准的推出。

早期计算系统数据量小，并行传输以架构简单、时序易控的优势成为主流，如 ISA/PCI 并行总线。但随着计算机、通信等领域向高速化、小型化演进，并行传输的短板愈发突出，需要新的解决方案。并行总线主要存在如下问题：

- 引脚与布线资源紧张，并行宽位总线导致芯片引脚、PCB 布线密度激增，推高成本且受空间限制（32 位 PCI 总线有 124 条物理连接线）。
- 高频下信号干扰严重，多路信号线串扰、电磁干扰及时钟抖动加剧信号失真。
- 时序同步难度陡增，速率提升放大时序偏移，同步设计复杂度指数级增长。
- 扩展与适配性差，传输距离短、共享带宽制约性能，单一接口无法满足多领域差异化需求，如 PCI 只能满足 32、64 位总线传输需求。

这些瓶颈催生了 SerDes 技术。SerDes 最早于 20 世纪 80 年代末至 90 年代初，美国贝尔系统（AT&T/Bell Labs）主导开发 SONET 标准，将多个 E1/T1 信号通过 SerDes 芯片串行化后送入激光器驱动电路，经光纤发送后在对端由 Deserializer（解串器） 恢复为并行数据流供交换机处理，解决了并行信号在单光纤的长距高速通信（SONET/SDH 光纤骨干网）需求。后续接口迭代中，SATA 取代 PATA（Parallel ATA，并行高级技术附件，也即 IDE 接口）、PCI-Express（PCIe）取代传统并行 PCI，其中 2003 年推出的 PCIe Gen1，作为首个在计算机外设互联领域规模化采用 SerDes 的标准，彻底突破 PCI 并行总线高频短板，推动 SerDes 向多场景延伸，在机内总线（PCIe、USB），存储（NVME），显示（HDMI）、网络等领域已广泛应用。

如今 AI 算力集群、超节点互联等场景的高速 Scale up/ Scale out 互联需求，进一步驱动 SerDes 向更高速率、更高可靠性迭代。

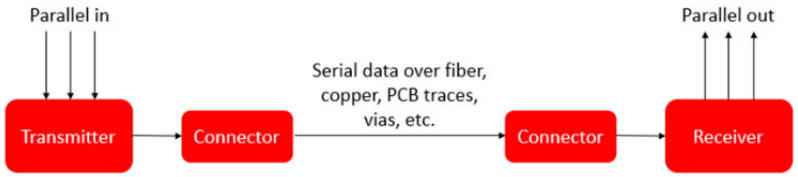


图52 SerDes 可在不同介质场景中应用

2. 核心解决问题

SerDes 技术的核心价值的是破解传统并行传输在高速场景下的多重瓶颈，同时适配多领域差异化传输需求，具体解决了如下关键问题：

- **引脚与传输资源紧张问题：**并行传输需为每路信号配置独立引脚与传输线，高速场景下宽位宽并行总线会导致芯片引脚数量、PCB 布线密度激增，不仅增加芯片封装、电路板设计成本，还会受限于设备物理空间无法扩展。传统 PCI 作为并行总线，其 32 位@33MHz 规格的最大带宽仅 133MB/s，且共享带宽架构

进一步制约性能；而 PCIe 作为首个引入 SerDes 技术的计算机外设互联标准，通过串并转换将多路并行信号压缩为 1–2 对差分串行信号传输，大幅减少引脚占用与传输线数量。如 PCIe 6.0 仅需一个差分对即可实现 8GB/s 的传输速率，远超传统 PCI 32 位总线的带宽，同时降低布线冲突与硬件成本。

- **信号干扰与完整性恶化问题：**并行传输中多路信号线间距近，高频场景下易产生串扰、电磁干扰（EMI），且电源噪声、时钟抖动会导致信号失真，传输速率越高干扰问题越突出。SerDes 采用差分信号传输技术，通过两路反向信号抵消共模干扰，同时串行传输减少干扰源数量，配合预加重、自适应均衡等算法补偿信号衰减，显著提升高速场景下的信号完整性，满足高干扰环境下的传输需求。

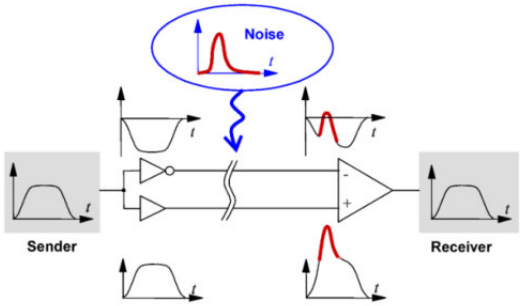


图53 差分信号抑制共模干扰 [39]

- **时序同步与扩展能力不足问题：**并行传输的多通道信号需严格保持时序对齐，速率提升会导致时序偏移量放大，增加同步设计难度，且并行总线的传输距离较短，难以适配长距互联场景。SerDes 通过内置 PLL 时钟模块与同步机制，实现串行信号的精准时钟恢复与时序校准，同时支持单链路高速传输与多链路绑定扩展，既解决长距离传输中的时序漂移问题，又可通过链路聚合灵活提升带宽，适配从 Die/芯片间短距互联到 Scale up 机内互联及 Scale out 机柜间长距传输的全场景需求。
- **多场景差异化传输需求问题：**不同领域对传输速率、功耗、可靠性的需求差异较大，传统并行接口难以兼顾多场景适配。SerDes 通过模块化设计与多协议兼容能力，可针对不同场景定制优化。如在 PCIe SerDes 技术下派生了基于 PCIe 的 NVME、USB4.0、Thunderbolt 3/4、CXL 4.0、M.2/U.2 等高速数据接口。

核心技术要点

1. 系统架构与核心模块

SerDes 系统由五大核心模块构成，各模块协同保障信号完整性与传输效率：

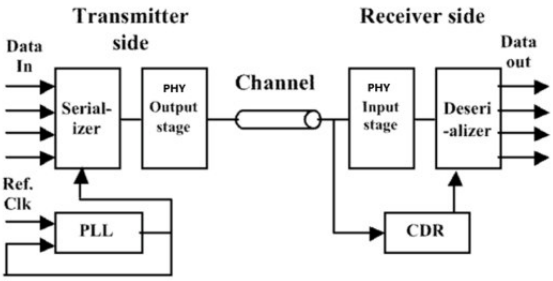


图54 SerDes 系统构成 [40]

并行数据接口：图示中的 Data in 接口，对接 CPU/GPU/DPU/FPGA/ASIC 内部总线（如 AXI、AHB 协议），支持 8/16/32bit 等位宽，负责并行数据收发，关键指标包括接口电压、时序稳定性及数据吞吐量，是芯片内部与串行链路的桥梁。

串/并转换器：串行器（Serializer）通过移位寄存器、FIFO 缓冲将并行数据压缩为高速串行流，高端方案集成预加重电路补偿高频信号衰减；解串器（Deserializer）则反向还原串行信号，内置均衡器抵消传输线码间串扰，同时解决时序对齐问题。

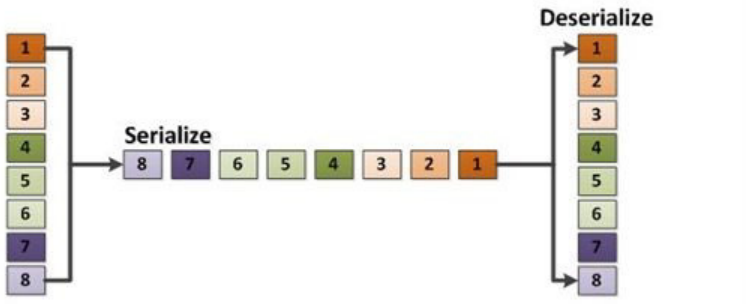


图55 并行数据串行化传输 [41]

PLL 时钟模块：生成低相位噪声、高稳定性的高速时钟，决定传输速率上限，支持动态频率调整以适配多协议需求，同时为接收端提供同步参考，是高速传输的“节拍器”。

物理传输接口（PHY）：包含发送驱动器与接收缓冲器，采用差分信号传输减少共模干扰，关键参数涵盖输出摆幅、差分阻抗及传输介质适配能力（铜缆、PCB、光纤等）。

CDR（Clock and Data Recovery，时钟数据恢复）：从不含独立时钟的高速串行数据流中提取时钟信号并同步采样数据，确保接收端能准确还原发送端的比特序列。

2. 调制技术与协议体系

调制技术直接决定传输速率与带宽利用率，协议体系则适配不同应用场景：

调制方式演进：主流采用 NRZ（Non-Return-to-Zero 不归零码，1 符号 1bit，结构简单，适用于 PCIe 5.0 及以下）和 PAM4（4-Level Pulse Amplitude Modulation 4 电平调制，1 符号 2bit，带宽翻倍，用于 PCIe 6.0、DDR5、112\224GT/s 等高速场景），在更高的速率下如 448GT/s，正在分析引入更复杂的调制方式如 PAM6 或 PAM 8 等的可行性。

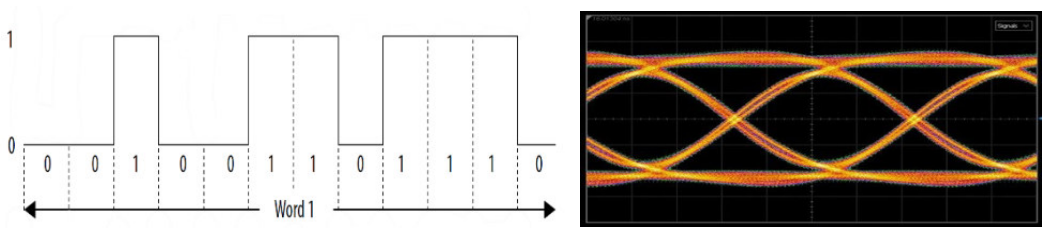


图56 NRZ（PAM2）调制及眼图

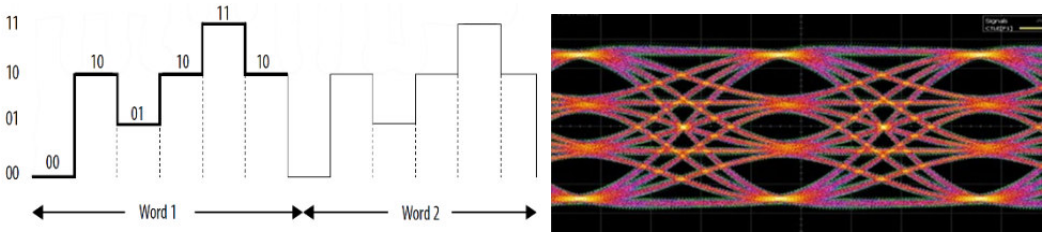


图57 PAM4 调制及眼图 [42]

主流协议分类：按 SerDes 的应用场景，可以将其进行如下**场景**简单分类，在各类场景的 SerDes 速率选择中，更关注的是满足场景及周边器件配合的应用要求。需注意的是 SerDes 本身是通用高速串行收发电路，不和特定协议绑定。链路物理层通过搭配不同的 PCS（编码/控制逻辑）和链路训练机制，同一 SerDes 可支持多种协议。

按应用类型分类：

接口标准类型	典型代表	核心技术参数	应用对象
总线类 SerDes	PCIe SerDes	版本覆盖 1.0~6.0；PCIe 6.0 采用 PAM4 调制，速率 64GT/s；支持 x1/x2/x4/x8/x16 链路宽度	CPU、GPU、网卡等外设互联
	CXL SerDes	基于 PCIe 5.0/6.0 演进，兼容 PCIe 物理层；支持缓存一致性协议；速率 32GT/s（CXL 2.0）~64GT/s（CXL 3.0）	异构计算集群、内存池化设备
	USB SerDes	USB 3.2 速率 20Gbps；USB4 速率 40Gbps；支持雷电 3/4 兼容	消费电子、外设扩展
网络类 SerDes	以太网 SerDes	覆盖 10G/25G/100G/400G/800G 以太网；200G 以上多采用 PAM4；支持 IEEE 802.3 标准	交换机、路由器、网卡、光模块
	InfiniBand SerDes	速率 200\400Gbps；支持无损传输和 RDMA	高性能计算集群、AI 超算互联
Chiplet类 SerDes	UCIe SerDes	3.0 版 64 GT/s（PAM4）/48 GT/s，2.0 版 32 GT/s（NRZ）；128b/130b 编码；支持 2D/2.5D/3D 封装	小芯片（Chiplet）间互联，如 CPU+GPU、CPU + 内存、AI 加速芯片组合
显示类 SerDes	HDMI SerDes	HDMI 2.1 速率 48Gbps；支持动态 HDR、8K 视频传输	电视、显卡、显示器
	DisplayPort SerDes	DP 2.0 速率 80Gbps；支持多流传输（MST）	电竞显示器、笔记本扩展坞

接口标准类型	典型代表	核心技术参数	应用对象
存储类 SerDes	NVMe over Fabrics SerDes	基于 PCIe 或以太网物理层；支持 TCP/RDMA 协议；速率与底层接口一致	分布式存储、全闪存阵列
	Fibre Channel (FC) SerDes	FC-NVMe 速率 32Gbps/64Gbps；专为存储区域网络（SAN）优化	企业级存储、数据中心备份
	SATA/SAS SerDes	SATA 3.0 速率 6Gbps；SAS 4.0 速率 22.5Gbps；采用 8b/10b 编码	机械硬盘、固态硬盘、磁盘阵列

按协议分类：

序号	协议	SerDes Role
1	PCIe Gen 1 - 6	使用 SerDes 实现双向高速通道
2	Ethernet (10G/25G/100G/400G)	使用基于 SerDes 的物理层（PHY）用于光 / 铜介质传输
3	SATA/SAS	用于存储设备的串行数据传输
4	USB 3.x/4	Type-C 接口和 DP模式
5	DisplayPort, HDMI	多媒体 SerDes 传输
6	JESD204B/C	ADC 与 DAC 和 FPGA/ASIC 之间的高速串行链路
7	UCIe, BoW, XSR, AIB	基于 SerDes 的芯粒互联标准

应注意“PCIe SerDes”或“Ethernet SerDes”并不是严格意义的分类方式，指的是为特定协议优化的完整 PHY 解决方案，其中 SerDes 是核心模拟前端。

Scale up 协议的发展，基本借用现有成熟连接器件和生态，如 PCIe 或 802.3 物理层：兼容现有 PCIe PHY（CXL3.0\4.0）或以太网 PHY(UALink\UB)及其光模块（如 QSFP-DD、OSFP）和铜缆，无需定制硬件；支持 802.3 标准的速率协商、链路训练机制，降低硬件适配成本。**如协议完全复用标准以太网 PHY，仅在链路层扩展相关字段，现有交换机通过固件升级，无需全新进行开发即可实现对新协议的支持。**目前除了 CXL（兼容 PCIe）和 NVLink（私有）外，其余主流 Scale up 协议（UALink、UB、ETH-X、SUE 等）均在物理层兼容 802.3 标准。

3. 核心设计挑战

高速化带来三大核心瓶颈：信号完整性（串扰、时钟抖动、电源噪声导致失真）、功耗热管理（高速处理占系统总功耗比例高，密集封装散热难度大）、多通道同步（温度、功耗波动易引发通道不一致，影响多链路协同），同时协议兼容性对材料和供应链提出更严格要求，需跨厂商协同优化。

当前技术进展

传输速率持续跃升，从成熟的 32GT/s（PCIe 5.0）向 64GT/s（PCIe 6.0）和 112GT/s 规模化过渡，头部企业已加速布局 128GT/s（PCIe 7.0）及 224GT/s 技术。英伟达 GB200 NVL72 服务器即使用了 224G SerDes 铜连接方案；400G 双向 SerDes（200G+200G，待进一步技术披露）技术也在 2026 年初的 CES 大会上作为英伟达下一代 Rubin NVL72 超节点的关键技术，计划在 2026 年中落地发布。

英伟达使用的 NVLink SerDes 为其私有标准，OIF（Optical Internetworking Forum，光互连论坛）正在联合 UALink（Ultra Accelerator Link，超加速器链路联盟）、UEC（Ultra Ethernet Consortium，超以太网联盟）、SNIA（Storage Networking Industry Association，存储网络行业协会）、OCP（Open Compute Project，开放计算项目）和 IEEE 802.3 等组织，积极制定开放的 448G SerDes 标准，有望在 2026~2027 形成标准落地。

在以太三代高速互联技术（56G、112G 和 224G）和 PCIe 6.0（64G）上，PAM4 调制方式已经成功应用。对于 448G SerDes 更高速率的技术，为降低奈奎斯特采样频率和缓解信道带宽限制，正在考虑使用更高阶的 PAM6 和 PAM8 调制方案，但由此带来的信噪比（SNR）降低及接受端时延容差窗口变窄，对电缆和连接器、PCB 提出了极高的要求。同时，使用新的调制方式也带来了和原有 PAM4 的后向兼容性问题，但为减轻 448G 高速带来的 SI（信号完整性）挑战，使用 PAM6 和 PAM8 是个很现实的选择。

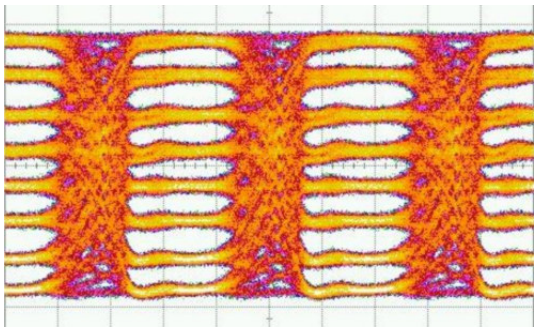


图58 PAM8 眼图，对容差带来更大挑战 [43]

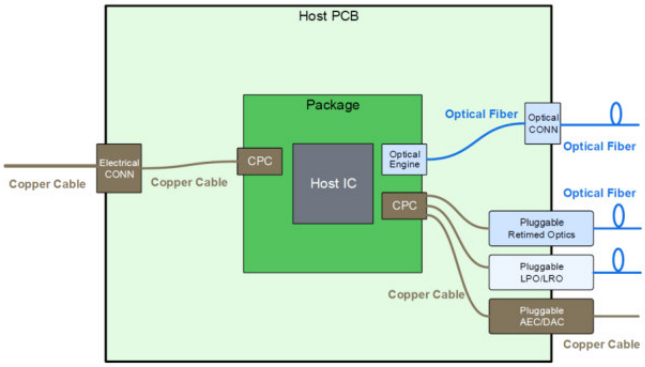
截至 2025.11，OIF 的 CEI LR（CEI：Common Electrical I/O）历史项目及 448G 项目的进展如下，后者还存在较多的待定项，预计在 2026 年能确定：

表8 OIF SerDes 项目进展 [44]

OIF CEI 项目	CEI-56G-LR	CEI-112G-LR	CEI-224G-LR	CEI-448G-LR
时间线	2014-2017	2017-2021	2021-	2026-
对应以太网速率	50/100/200G	100/200/400G	200/400/800/1600G	400/800/1600G/3200G
以太交换机容量	12.5T	25T/50T	50T/100T	100T/200T
每通道数据速率	56 Gbps	112 Gbps	224 Gbps	448 Gbps
调制方式	PAM4	PAM4	PAM4	待定
传输距离目标	3 米铜缆	2 米铜缆	1 米铜缆	待定

OIF CEI 项目	CEI-56G-LR	CEI-112G-LR	CEI-224G-LR	CEI-448G-LR
前向纠错（FEC）前误码率（BER）目标	1e-4	1e-4	1e-4	待定
串行器/解串器（SerDes）架构	模拟/数字信号处理器（DSP）	模拟/数字信号处理器（DSP）	数字信号处理器（DSP）	待定

作为下一代高速互联关键技术，448G SerDes 支持多种物理介质连接方式：



CPC: Co-packaged Copper　　CPO: Co-packaged Optics

LPO/Linear: Linear Pluggable Optics　　LRO/RTLR: Linear Receiver Optics

DAC: Direct Attach Copper Cables　　AEC: Active Electrical Cables

图59 448GT/s SerDes 可能采用的多种连接方式 [45]

CPC（Co-packaged Copper）：共封装铜缆（CPC）是将高性能铜缆直接集成到芯片封装附近的互连技术，通过缩短电气路径、利用低损铜缆，在短距实现高带宽、低功耗、低成本的数据传输，是高速计算系统中替代传统 PCB 走线和可插拔模块的重要方案。

CPO（Co-packaged Optics）：共封装光学（CPO）是面向高速系统的互连架构，其核心思想是将光学引擎（光模块）与 ASIC 芯片（如 GPU、TPU、交换机芯片）共同集成在同一封装基板或中介层（Interposer）上，从而大幅缩短电通道长度，解决“功耗墙”和“带宽墙”问题。成本显著高于 CPC 方案。

LPO（Linear-drive Pluggable Optics，线性驱动可插拔光学）,LRO(Linear Receiver Optics)：是一种省去传统光模块中 DSP 芯片、采用直驱线性电路的低成本低功耗光互连方案，适用于短距、高密度数据中心场景，在性能与成本之间取得新平衡。

DAC（Direct Attach Cable，直连铜缆）：是电接口的高速无源铜缆组件，用于短距离（通常≤7 米）设备间高速互联，具有低延迟、低功耗、低成本优势，广泛应用于数据中心机架内服务器与交换机连接。

AEC（Active Electrical Cable）：在铜缆中集成 Redriver/Retimer 的有源高速互连方案，兼具铜缆的低成本与光缆的延伸能力，是高速短距互连的重要选项，也称为 Active DAC。

机架内的电气链路因其低功耗、低延迟和低成本特性而至关重要，网络运营商严重依赖这些链路在未来速率下的持续可用性。共封装铜缆（CPC）解决方案有望显著改善损耗，但 SerDes 可实现的实际传输距离（无需或仅需少量中继器）将决定铜缆互联是否仍适用于此类应用，或者即使在机架内也需要采用光学解决方案。如

224G SerDes 最先在 NVLink 落地一样，英伟达有可能最先实现 448G SerDes 落地，其是否使用 CPC 方案可能成为重要方向参考。预计 IEEE 等公共标准也会使用 CPC 作为机架内首选互联方案。

共封装光学（CPO）是最大限度减少电气信道损耗的有效方式，因此也成为高速互联架构的热门选择。其生态系统仍在发展中。同时也存在集成复杂性、可靠性和可维护性担忧以及初始成本较高等因素的限制。随着这些挑战得到解决以及技术成熟，CPO 可能在 448G 及其后的超高速时代获得更广泛的应用。

未来趋势与现存瓶颈

SerDes 正在走向 224GT/s 规模产业化，448GT/s 标准落地的发展道路上，预计 2026~2027 年 OIF 会完成 CEI-448G（CEI：Common Electrical I/O）规范草案。同时 Broadcom、Marvell、Intel 等已开展 448G SerDes IP 原型设计，向更低功耗、更强兼容性演进。低损耗 PCB 材料、100+GHz 连接器、先进封装等周边技术在同步推进。其中高阶调制技术（如 PAM6/8）、AI 自适应均衡算法、铜光融合互联方案将成为研发重点。PCI-SIG 也已于 2025 年 8 月启动 PCIe 8.0 规范制定，目标实现 256GT/s 信令速率，进一步推动超高速 SerDes 技术迭代。

表9 海外主要 SerDes IP 供应商 [46]

公司	主要 SerDes IP 产品
Synopsys	Industry leader in PHY IP for PCIe, Ethernet, USB, UCIe
Alphawave Semi	Specializes in 56G/112G SerDes for AI/HPC/datacenter
Cadence	Offers SerDes IP and PHYs (USB4, PCIe, 112G)
Rambus	High-speed memory and SerDes PHY IP
Credo, eSilicon (Inphi)	Optical SerDes PHY IP, CPO
Siemens EDA	Offers simulation and verification of SerDes systems
AnalogX, Kandou, Marvell	Focused on ultra-low-power or chiplet-centric SerDes designs

高速 SerDes 的发展面临如下瓶颈：224GT/s 及以上速率场景下信号完整性优化难度呈指数级递增，功耗控制与量产良率待突破。同时，国内高速 SerDes IP 及先进工艺、先进材料距离国际先进技术还有差距，尚需潜心追赶突破。

2.2 链路层

Scale up 总线链路层是总线协议栈（物理层->数据链路层->事务层）的核心中转层，核心职责是衔接物理层信号传输与事务层地址映射/内存访问逻辑，保障事务包（TLP，Transaction Layer Packet）在链路内的可靠、有序、高效传输。对于 Scale Up 总线，如 UALink、PCIe（含 CXL）、Infinity Fabric 等，其聚焦节点内高密度高速互联场景。链路层需在通用功能基础上，满足极致低延迟、高带宽、高容错及缓存一致性（部分协议要求）适配需求，其中 **LLR（Link Layer Retransmission，链路层重传）、CBFC（Credit-Based Flow Control，基于信用的流量控制）、虚拟通道（VC）**是核心支撑功能。本章以成熟的 PCIe 生态为主，阐述链路层的主要功能。各 Scale up 协议的具体实现可参见具体协议，但主要关键功能类似。

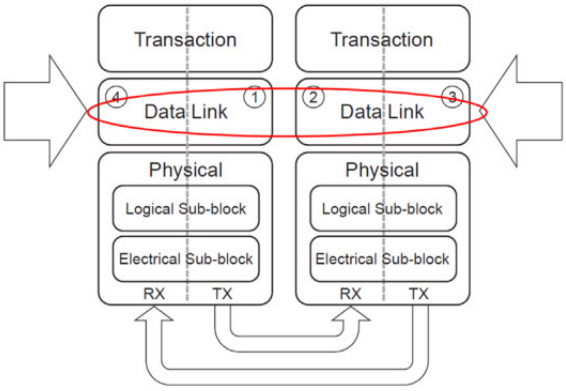


图60 链路层位置 [47]

总线链路层的主要功能

总线链路层的设计核心是解决物理层（下层）信号不稳定性与事务层（上层）高效交互的矛盾，功能围绕“可靠传输、流量管控、链路运维”三大维度展开，主要包括以下五个功能，其中重点细化 LLR、CBFC、VC 的技术实现。流控和 VC 是事务层和链路层共同参与完成，在 PCIe 规范中将其功能放在事务层。为结合链路层实现，本文将流控和 VC 功能在链路章节进行展开描述。同时，部分协议（如 UB）也将流控和 VC 管理放在链路层实现。

图61 数据链路层主要功能（UB） [48]

1. 链路管理：全生命周期状态管控

负责总线链路从初始化到故障恢复的全生命周期管理，确保链路长期稳定运行，核心流程包括初始化协商、链路训练、状态监控及低功耗管理。

- **初始化与协商：**链路两端设备上电后，通过握手信号协商链路速率（如 112GT/s or 224GT/s）、链路宽度（x4/x8/x16 通道聚合）、编码方式（PAM4/NRZ）及 LLR、CBFC 参数（重传阈值、信用池大小）。
- **链路训练：**发送训练序列校准信号时序、补偿信号衰减（调整 CTLE-连续时间线性均衡器参数），确保物理层信号质量达标；训练过程中同步验证 LLR 重传链路与 CBFC 信用交互的有效性。
- **状态监控与故障恢复：**实时监测链路错误率、信号完整性及 CBFC 信用消耗速度，若错误率超过阈值，先触发 LLR 重传，无效则启动链路重训练；若链路中断，快速通知事务层并释放信用资源，避免无效数据传输。

2. 事务包封装与解封装：标准化链路传输格式

链路层负责将事务层下发的 TLP 封装为链路层帧，接收端解封装后传递至事务层，帧结构设计直接影响传输效率与兼容性。

一般封装流程为：为 TLP（事务层报文）添加链路层帧头、帧尾，形成完整链路帧。帧头包含链路标识、事务包类型、长度信息及 LLR 重传序列号，用于接收端识别帧边界、事务属性及重传定位；帧尾包含 CRC 校验码，用于差错检测。与网络链路层（如以太网）不同，总线链路层帧一般依赖事务层的 ID（GPU-ID，设备 ID）或 BDF 地址（PCIe：总线-设备-功能）实现寻址，无需节点间路由，精简帧头字段、简化帧结构并降低解析延迟。

在 Scale Up 总线低延迟设计中，普遍采用 FLIT（Flow Control Unit，流控制单元）技术优化封装与传输效率，其核心是将可变长度的 TLP 拆分为固定长度的 FLIT 单元进行传输，而非直接传输完整可变长帧。FLIT 单元通常包含数据 FLIT、控制 FLIT 及尾 FLIT 三类：数据 FLIT 用于承载核心数据，控制 FLIT 用于传输链路控制信息（如 LLR 重传指令、CBFC 信用交互信号），尾 FLIT 用于标识一个完整 TLP 的 FLIT 流结束，确保接收端可精准重组原始 TLP。

FLIT 技术的核心优势的是适配低延迟与高效流控需求：

- 固定长度设计可简化接收端解析逻辑，无需动态判断帧边界与长度，解析延迟降低，同时便于硬件并行处理，提升链路吞吐；
- 可实现精细化流控，CBFC 信用额度可按 FLIT 单元计量，而非传统的整包计量，接收端可更精准地管控缓冲区占用，避免因可变长帧导致的缓冲区碎片化，导致可用缓冲区空间不足，发送端因信用不足导致的暂停频次；
- 适配 LLR 重传优化，当 TLP 传输出错时，可仅重传包含错误数据的 FLIT 单元，无需重传完整 TLP，配合 LLR 的段重传机制进一步压缩重传开销，尤其适配 AI 场景中长数据包的传输需求。此外，FLIT 流可灵活承载事务合并后的小粒度 TLP，将多个小 TLP 的 FLIT 流交织传输，提升链路带宽利用率的同时，不影响传输有序性。

部分从以太演进来的 Scale up 协议，如 ETH-X、SUE、ESUN 等，为保持和以太报文的兼容性，没有使用 FLIT 技术，而使用变长的报文结构。

3. 差错检测与 LLR 重传机制：保障数据完整性

该功能通过“差错检测-重传纠正”二级机制，解决物理层信号衰减、串扰、噪声导致的数据位翻转问题，LLR 是链路层重传的核心实现方式，区别于事务层重传，具备更精细的链路级控制能力。

- **CRC 校验：**发送端在 TLP 末尾添加 CRC 校验码（通常为 16 位/32 位），校验范围覆盖帧头、数据段及控制字段；接收端接收后重新计算 CRC，若与发送端校验码一致则确认数据有效，不一致则标记错误并触发重传流程。
- **LLR 核心机制：**作为链路层专属重传策略，LLR 采用“即时重传+局部重传”设计，无需等待事务层指令，大幅降低重传延迟。具体流程为：接收端检测到错误后，立即向发送端返回 NACK（否定确认），并携带错误帧标识；发送端收到 NACK 后，仅重传错误帧（Selective Repeat，SR 方式），而非错误数据及其后的所有已发送数据（Go-Back-N，GBN 方式），若连续多次重传失败（通常 3-5 次），则触发链路重训练或上报事务层。显然，SR 方式在响应速度和带宽占用方面优于 GBN 方式，但实现复杂度也高于 GBN

方式。此外，LLR 支持重传优先级调度，高优先级事务（如 GPU 间梯度交换）的重传请求可抢占带宽，避免低优先级事务阻塞关键数据传输。

4. 虚拟通道（VC）管理：多事务并行隔离

虚拟通道是链路层实现单一物理链路承载多逻辑事务流的核心技术，通过在物理链路内划分独立逻辑通道（VC，Virtual Channel），实现不同类型事务的并行传输与隔离调度，避免低优先级事务阻塞高优先级事务，是 Scale Up 总线支撑多并发事务的关键优化手段，需与 LLR、CBFC、FLIT 技术深度协同。

- **核心原理：**链路层将物理链路逻辑划分为多个独立 VC，每个 VC 分配唯一标识（VC ID），并具备独立的事务队列、缓冲区、重传缓存及流量控制上下文。发送端按事务类型或优先级分配至对应 VC，通过 FLIT 帧头携带 VC ID，接收端根据 VC ID 将事务分流至对应逻辑通道处理；各 VC 共享物理链路带宽，但逻辑上相互隔离，可独立进行 LLR 重传与 CBFC 流量控制，互不干扰。
- **通道划分与隔离：**通常按事务属性划分 VC，典型配置包括数据 VC（承载 GPU 间梯度交换、内存读写等核心数据）、控制 VC（承载配置命令、Credit 刷新信息、状态上报等控制信号）、一致性 VC（承载缓存无效请求、写回确认等一致性事务）。每个 VC 配备独立 CBFC 信用池，避免某类 VC 信用耗尽导致其他 VC 阻塞；同时独立维护 LLR 重传队列，确保高优先级 VC 的重传请求不被低优先级 VC 抢占，也确保控制类信息（Credit 刷新等）得到最优先传送。在虚拟化场景中，可以为不同 VM 或容器绑定不同的 VC，从而提供硬件级隔离，提升安全性与确定性。各 VC 的发送顺序，按链路层可定义的发送策略进行，一般有如下方式：
    - 严格优先级（Strict Priority）：高优先级 VC 始终先发
    - 加权轮询（WRR，Weighted Round Robin）：按权重分配带宽
- **与 FLIT 机制协同：**FLIT 技术（见后）为 VC 调度提供了更多便利，固定长度的 FLIT 单元可按 VC ID 交织传输，实现多 VC 并行吞吐；LLR 针对每个 VC 独立统计错误率、执行重传策略，避免单 VC 重传影响全链路事务；CBFC 通过 VC 级信用管控，精准匹配不同 VC 的事务处理能力，例如为一致性 VC 高优先分配信用，保障缓存一致性交互的低延迟。

  VC 的控制由事务层和链路层共同配合完成，如下以 PCIe 的虚拟通道控制进行说明：

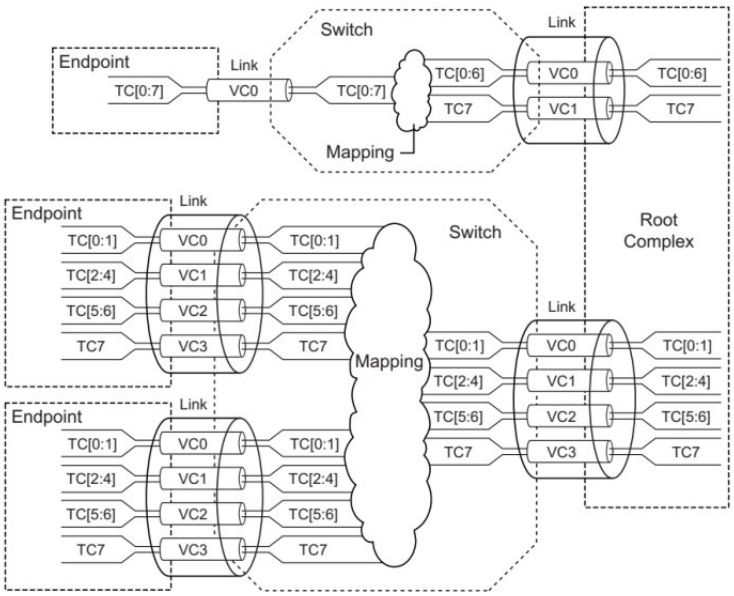


图62 PCIe / CXL 使用转发优先级（TC）配合虚拟通道（VC）进行转发 [49]

事务层、链路层、物理层在 VC 转发中的作用：

表10 PCIe 虚拟通道控制各层功能

层级	VC 相关操作
事务层（Transaction Layer）	• 决定 TLP 使用哪个 VC • 在 TLP 头部设置 TC（Traffic Class）字段（3 位） • TC 通过 TC/VC 映射表 转换为具体 VC ID
数据链路层（Data Link Layer）	• 为每个 VC 维护独立信用计数器和缓冲区 • 调度器按策略选择 VC 发送 TLP
物理层（Physical Layer）	• 透明传输：不感知 VC，仅串行化比特流

基本过程为：

- PCIe 初始化时，BIOS/OS 等需要对配置空间里的 TC-VC 映射表进行初始化。
- 事务层将数据或请求打为 TLP 报文，报文头包含 3bit 的 TC（Traffic Class）信息，发给数据链路层。
- 数量链路层将根据配置空间的 TC-VC 映射表选择对应的 VC 发送 TLP，同时维护信用计数器和缓冲区。
- 事务层根据信用计算器的计数，决定是否继续或暂停发送报文，以此实现流控（CBFC），见后。

5. CBFC 流量控制：避免接收端缓冲区溢出

CBFC 是总线链路层的核心流量管控机制，相较于网络链路层的 PFC（优先级流控），其适配总线“短距、高速、低延迟”特性，通过信用额度动态分配，实现无阻塞传输，是 Scale Up 总线高带宽利用率的关键。

- **核心原理：**链路初始化阶段，接收端向发送端通告自身各类型事务包（读请求、写数据、完成包）的缓冲区容量（一般为虚拟通道缓冲区），即信用额度（Credit），单位为定长事务包（FLIT）个数或字节数，在一些协议中也将一个 Credit 对应的数据长度定义为 Cell，收发双方需在初始化时协商确认。发送端每发送一个对应类型的 TLP，消耗 1 个信用额度或对应的字节数 Credit。当某类事务包信用耗尽时，发送端无 Credit 可用，即暂停该类型事务传输，直至接收端释放缓冲区并归还信用，再恢复发送。以下图 PCIe Credit 流控为实例：

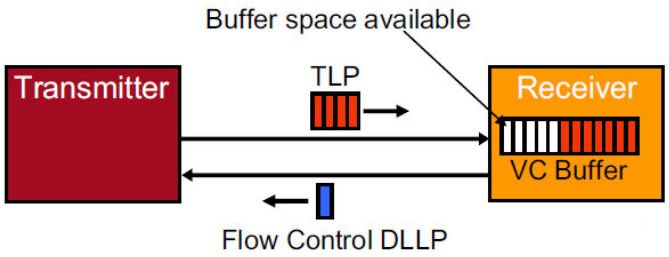


图63 链路层流控过程

- **适配 VC 机制：**CBFC 支持按 VC 进行信用管理，将接收端缓冲区按 VC 即事务类型、优先级拆分独立信用池，避免单一类型事务耗尽全部信用，导致其他事务阻塞；同时支持信用动态调整，接收端可根据自身处理能力实时调整信用分配，适配发送端突发流量。同时，为保证控制事务的最高优先级，配合 VC 的划分，CBFC 机制将信用池划分为数据事务池与控制事务池，确保控制信息、一致性信息、信用等核心数据传输不被常规数据类事务干扰。

  为更合理、充分地使用接收缓冲区空间，PCIe 6.0 版本正式引入新的 VC 流控机制，即将缓冲区进一步拆分为 VC 专用缓冲区和 VC 共用缓冲区（池），同理将 Credit 分 VC 通道专用（Virtual Channel Credit）和池共用（Pool Credit）两种类型，多个虚拟信道可共用 Pool Credit。使用共享流控（Shared Flow Control）有如下优势：[50]

  - 共享流控机制可用于降低虚拟通道（VC）的资源需求。启用共享流控时，每个 VC 关联两组资源：一组专用资源池（通常较小），独立分配给每个 VC，用于避免死锁（确保发送端至少能使用专用信用发送一个 TLP）；另一部分为共享资源池（通常较大），由多个 VC 共同使用。
  - 共享流控通过允许多个虚拟通道共用同一组公共资源，从而实现低成本的多 VC 实现方案。然而，与仅使用专用信用的方式相比，由于需要引入共享结构，其成本和复杂度有所增加。

  UALink 1.0 协议也借鉴了该机制，在数据流每一次转发过程中均依靠 Credit 机制进行流控，并将 Credit 按是否公用分为专用（Virtual Channel Credit）和共用（Pool Credit）。在单颗 Switch 芯片中，在不同的转发阶段也需要进行 Credit 流控，如下图，芯片内部的转发流程中存在多个 Credit Loop。

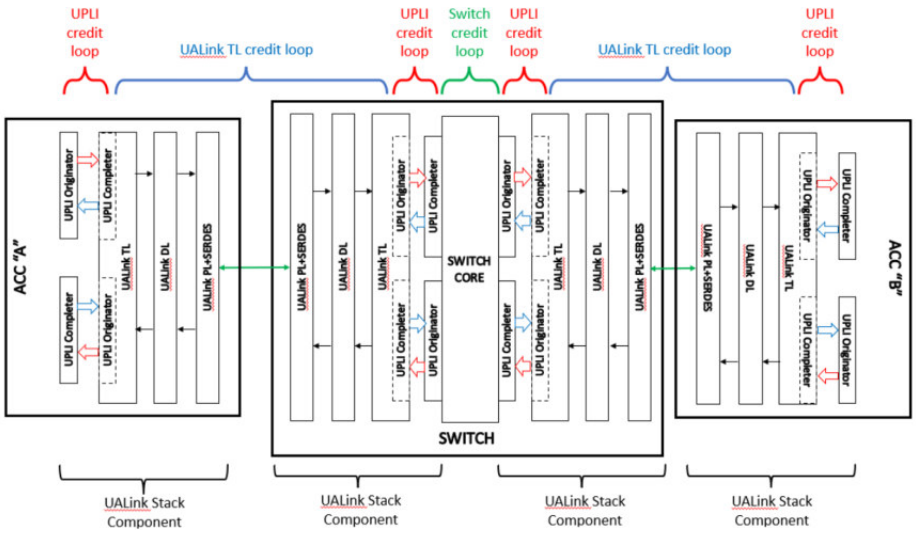


图64 UALink 转发过程中的流控环（Credit Loop）[51]

UB 协议的 Credit 流控机制也采用类似机制，UB 使用 VL（Virtual Lane）替代了 VC 概念。其流控特点：

- 数据链路层每个链路在完成初始化自协商后会启用部分或全部 VL（Virtual Lane， 类似于 VC），其中 VL0 默认启用，其它 VL 根据初始化自协商结果决定是否启用。
- Credit 独占模式：本端根据 Receive Buffer 大小和 Cell 粒度计算支持的信用证数量，根据管理面配置的策略为启用的每个 VL 分配独占的信用证，即各个 VL 使用各自的信用证，互不干扰。随后链路两端互相将本端初始化的信用证发送给对端完成初始化操作。
- Credit 共享模式：将所有接收到的信用证记录到共享信用证计数器中。为避免某 VL 长期饥饿得不到信用分配，信用共享模式下也需要给 VL 分配独占信用，即根据管理策略从共享信用计数器中分配部分信用作为独占信用给启用的 VL。独占信用不会被其它 VL 占用，独占数量根据实际场景定义。信用初始化完成后，接收到对端返还的信用时，将返还的信用汇总到共享信用计数器中，并检查各个 VL 独占信用是否充足，即是否与初始化时分配的信用数量一致，不足则进行补充，补充顺序可根据实际场景定义，如优先补充高优先级 VL 信用。

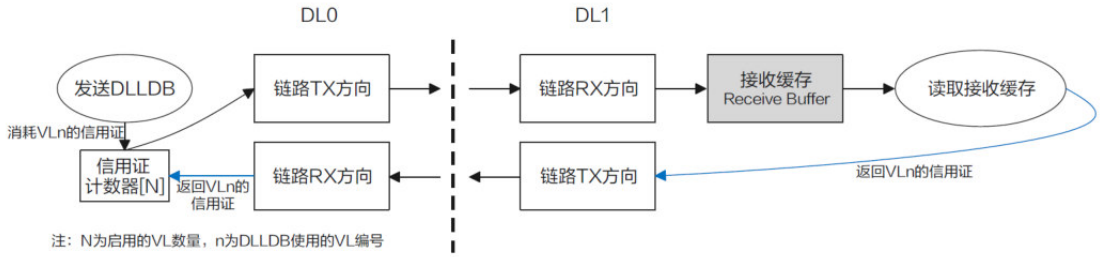


图65 UB 信用流控过程 [52]

UB 数据链路层的信用流控过程示例（Credit 独占模式）：

- DL1（接收端）在 初始化状态时通过链路 TX 方向发送信用数据包，为 DL0（发送端）的各个 VL 分配初始信用，信用总数量对应 DL1 Receive Buffer 的空间大小；

- DL0 基于每个 VL 维护信用计数器，信用计数器用于记录对端 Receive Buffer 中目前可用的 Buffer 空间大小。RX 接收到返还信用后，信用计数器按对应数量增加信用；当 TX 发送 DLLDP 后，信用证计数器则减去 DLLDP 占用的信用数量。只有当 DL0 的 VL 持有足够的信用证后，才能将对应数量的 DLLDP 发送到 DL1 对应 VL 的 Receive Buffer 中；
- DL1 Receive Buffer 中的 DLLDP 被读走后释放对应的信用证，释放的信用证数量嵌入到发往 DL0 的 DLLDP 中返还给 DL0 对应的 VL；
- DL0 收到释放的信用证后，将其加到对应 VL 的信用证计数器中，回到第二步。

对于变长的报文发送，发送方要攒够足够发送一个报文的 Credit 后，比如 5 个 Credit，才能发送这个报文。取决于 Credit Cell（一个 Credit 对应的定长数据量）大小和报文长度。最后一个 Cell 发送时，通常只有部分是有效数据，需对无效数据进行填充（一般为补零）处理。

基于 Credit 的流控（CBFC）是 Scale up 协议的主要流控方案，将缓冲区分为虚拟通道独享及通道共享两种类型并分别进行 Credit 流控，可以更有效地使用有限的缓冲区资源，该方式越来越多地被高速互联协议用于点对点流控。同时也有部分协议扩展 Credit 流控方式用于端到端的流控（如 ETH-X），核心原理均为发送端基于接收端提供的 Credit 决定发送的 FLIT 数量，实现数据流量按需发送。

总结

总线链路层的通用功能以“可靠传输、流量管控、链路运维”为核心，LLR、CBFC 与虚拟通道（VC）构成三大支柱，分别承担链路级重传纠错、无阻塞流量控制、多事务并行隔离的关键作用；Scale Up 总线链路层大都建立在此基础上，通过 LLR 低延迟优化、CBFC 动态调度、VC 级隔离与带宽分配、容错强化及缓存一致性适配（部分协议支持），实现“极致低延迟、高带宽、高可靠”的场景化需求。其本质是为节点内高密度高速互联量身定制的轻量高效传输层，区别于 Scale Out 总线的通用性设计，最终支撑 GPU 间直接内存事务传输，成为超节点、高性能计算等场景的算力互联基石。

2.3 事务层 [53]

事务层的定义

Scale up 总线的事务层（Transaction Layer）位于传输层之上、功能层之下，是协议栈的顶层核心层，负责将上层应用的内存访问（Load/Store、URMA 异步访问）、消息传递（Send/Recv）等操作封装为标准化的网络事务（如内存读写事务、消息收发事务）传递给传输层，并处理事务的排序（避免乱序导致的逻辑错误）、流量控制（防止缓冲区溢出）等关键问题，确保事务的一致性和可靠性。

硬件总线领域的“事务层”概念，其来源是英特尔公司 2001 年提出的 PCIe 总线协议，核心是通过高速串行点对点双通道传输技术，解决了旧有并行总线在信号干扰、频率提升瓶颈和带宽限制等方面的问题。

PCIe 总线采用分层协议架构（物理层、数据链路层、事务层），其中事务层是连接设备核心与总线的关键层，其核心功能包括：

- 事务封装与解析：将设备核心的请求（如存储器读写、I/O 操作）封装为事务层数据包（TLP，Transaction Layer Packet），发送至数据链路层；同时解析接收的 TLP，传递给设备核心。

- 流量控制与 QoS：通过“基于信用的流量控制”机制，确保发送端与接收端的数据速率匹配，避免缓冲区溢出；支持服务质量（QoS），为不同优先级的事务分配带宽。
- 事务排序与路由：定义事务的路由方式（如地址路由、ID 路由），确保数据包准确送达目标设备；支持“宽松排序”，提升总线利用率。

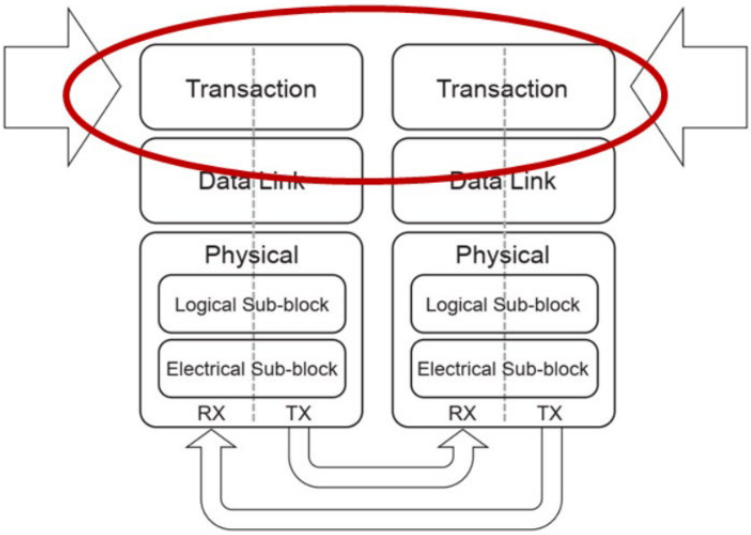


图66 PCIe 分层

Scale up 总线事务层的主要功能如下：

- 语义转换与事务抽象（核心基础功能）：

  接收上层发起的 Load/Store、RDMA 读写等 IO 操作，将其转换为标准化、硬件可识别的事务语义，主流实现均支持 Memory Read、Memory Write、Atomic 原子操作、Message 控制消息四大类基础事务，屏蔽底层传输细节，为上层提供统一的内存访问抽象，实现跨节点显存/内存的直接、透明访问。

- 事务的封装、解析：

  发送侧：为每个事务分配唯一标识，完成地址转译、事务头封装、数据块打包，生成符合协议规范的事务层微片（TL Flit），交付给下层数据链路层；

  接收侧：完成事务报文的解析、头信息校验、地址解码，还原为上层可识别的操作指令，同时完成请求与响应报文的配对，形成完整的事务闭环。

- 事务调度与全生命周期管理：

  负责事务的发起、并发调度、超时管理与状态机维护，跟踪每个事务的执行状态，处理异常事务的重传、终止与错误上报；同时支持多虚拟通道、多优先级事务的调度，优化链路带宽利用率，保障时延敏感类事务的优先传输。

- 传输保障与流控：

  与数据链路层紧密配合，实现端到端的基于信用的流控（CBFC）、拥塞管理与 QoS 调度，区分不同业务流量的优先级，同时保障同路径事务的保序传输，避免乱序带来的上层开销，实现高可靠传输。

部分协议还在事务层扩展了 RDMA 语义兼容、集合通信加速、报文聚合、端到端序保证等能力，以适配大模型训练/推理的多样化通信需求。

内存语义请求处理是事务层的核心功能，本节对内存语义、消息语义相关概念进行说明，集合通信是分布式训练的高层通信模式，依赖事务层提供的功能实现，在本节一并介绍。流控和 VC 是事务层和链路层共同协同完成，相关概念已在链路层章节进行介绍。

内存语义

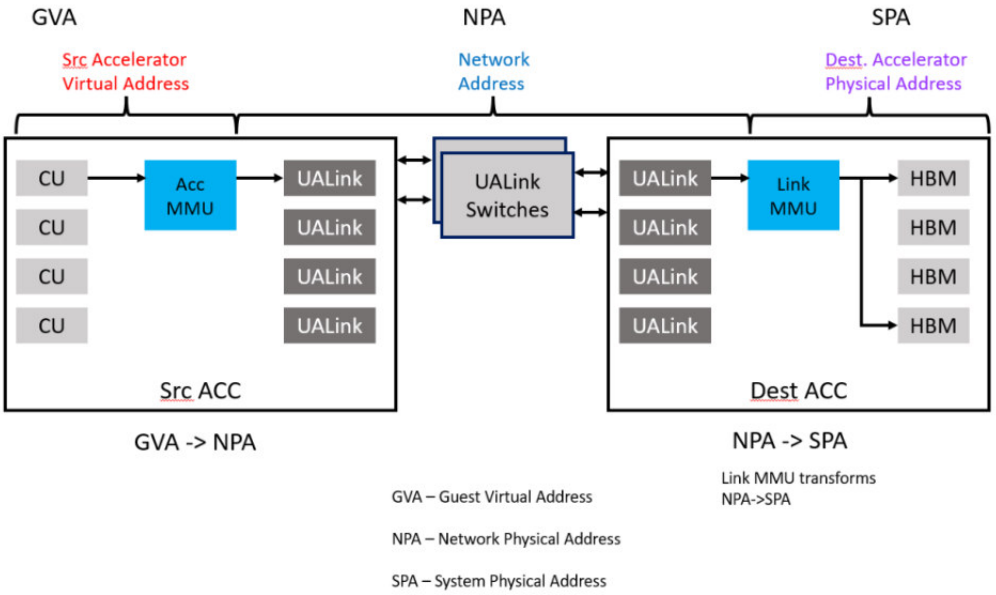
在计算机组成原理中，内存语义是指用于控制对共享内存区域或变量访问的一系列操作逻辑。内存语义就是定义了一套规则，告诉计算机系统中的各个部分，在什么时候、以什么方式可以访问或修改内存中的数据，确保数据的一致性和准确性。英伟达定义的超节点特征为节点内部处理器（GPU/CPU）通过 Scale up 网络互联，处于同一个 HBD（High Bandwidth Domain，超带宽域）。HBD 域包含了软硬件两个层面，在硬件互联层面，超节点 HBD 域内采用 NVLink、UALink 等专用 Scale up 互连协议，在加速卡（GPU/NPU）之间构建了高带宽、低延迟的通信域。解决了传统架构中 GPU 间通信必须经由 CPU/PCIe 总线的通信瓶颈，为海量数据并行处理奠定了物理基础。在软件与系统层面，HBD 内的计算节点资源管理也有所改变，计算节点可以基于内存语义直接通过内存地址空间访问 HBD 内的内存地址，计算节点间的高速通信通常直接绕过 OS 内核协议栈，使用专用硬件完成数据读写，显著降低通信开销。

1. 为什么需要内存语义

传统服务器间的通信互联依赖网络技术（如以太网或 InfiniBand），相对于系统内部总线，网络复杂的协议栈、额外的转发开销、微秒甚至毫秒级的延迟，传输效率存在数量级差异。内存语义允许 CPU、GPU、加速器之间像访问本地内存一样直接操作对方内存，处理器直接使用最基础的 Load（读取）和 Store（写入）指令访问远端数据，大大简化了编程模型。省去消息机制（如 RDMA）使用的“数据拷贝－封装－传输－解封装－拷贝”的中间环节，大幅降低跨设备数据交互延迟及提升了访问效率。

2. 内存语义的核心机制：

- **统一地址空间：**远端设备 CPU 内存/GPU 显存被映射到本地的物理地址空间中，系统可以直接用 LOAD, STORE 指令访问。统一地址空间也简化了软件开发难度，开发者无需关注数据分发、缓存同步、地址映射等底层细节，可按单节点模型开发并行软件，降低多设备协同的代码复杂度。



- 跨交换机访问对端内存，使用 GVA（Guest Virtual Address）地址发起访问。
- GVA 地址经本地 MMU 转为为网络物理地址（NPA，NetWork PhysicalAddress）
- NPA 地址在对端 MMU 转换为本地系统物理地址（SPA，System Physical Address）

图67 UALink 通过 MMU 进行地址转换，实现全局访问

- **硬件级缓存一致性**：在硬件层面实现缓存一致性协议，保证 CPU、GPU 和加速器共享内存数据时不会互相冲突，数据一致。当一个节点修改了数据，硬件会自动通知其他节点更新或失效其缓存。应用程序不需要像使用 RDMA 那样，显式地去管理数据的同步逻辑。可大幅降低跨设备数据交互延迟、提升算力协同效率，简化多设备并行软件的开发复杂度。CXL 3.0/4.0 可实现全一致性多主机间内存池共享；NVLink 5.0 支持 GPU 与 Grace CPU 的 C2C 全一致性互联，UALink 通过以下方式及软件方式来保证一致性：
  - 读对端内存时如数据存在 Cache（对端 ACC 处理器）中，要求 Cache 先行回写，以保证读取的内存数据是最新。
  - 写对端内存时，如该段内存数据存在于 Cache 中，会将所涉及 Cache 行作废。
  - 如写入数据被某核 Cache，且只占 Cache 行部分时，需先将对应 Cache 行数据回写，再进行写入，同时将 Cache 核的该行 Cache 作废。

  因硬件级 Cache 一致性协议（如 MESI）在读写数据时，须确认和修改 Cache Line 状态，此过程存在多次和 Cache 端的交互。当 HBD 规模较大时，维护一致性的通讯与同步开销可能抵消并行效率，尤其在高并行密集访问场景下易降低数据吞吐量，导致链路的有效利用率降低。

  同时，因需要复杂硬件逻辑维护一致性，会额外增加芯片面积与功耗，Scale up 协议可能有选择地部分支持一致性。即通过纯软件方式来保证读写的顺序（Fence 操作），保证数据操作的完整性（如 ETH-X 实现）；也可以减小一致性的作用范围，仅限于 CPU 及和其直联的 GPU 间保证 Cache 一致性（如英伟达 NVLink C2C），在此范围外的一致性由软件保证，如在软件开发时避免出现可能导致 Cache 不一致的读写操作，可以使用 Fence 等机制来保证读写顺序。

UB 使用对特定地址块的 Ownership 机制，提供对共享内存的读写控制访问权限，来保证跨节点数据一致性，也是一种值得借鉴的方式，但需考虑权限控制带来的管理复杂性。

协议名称	误码率要求	关键技术手段
NVLink	1e-12	CRC 校验 + 自适应 FEC
PCIe	1e-12	LCRC、ECRC 校验 + FEC (Gen6)

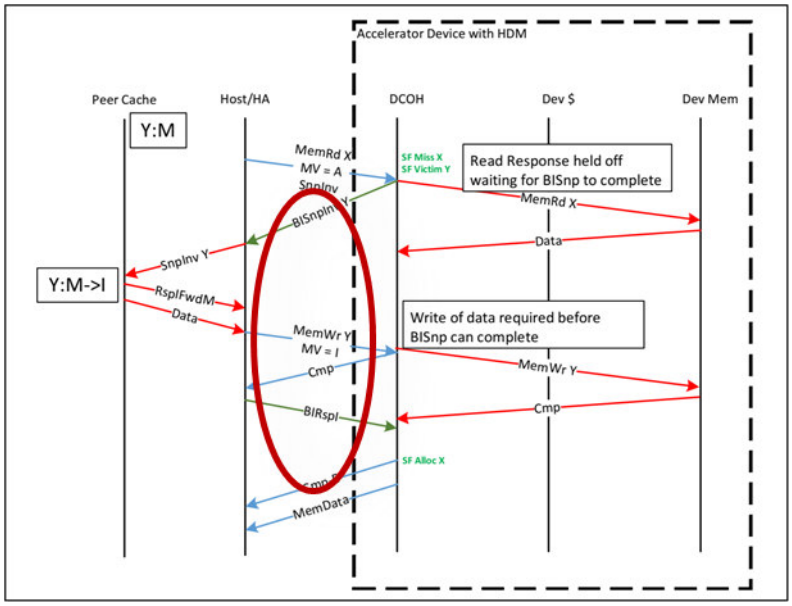


图68 CXL 的数据读取过程，需进行多次 Snoop 报文（红圈）交互来确认数据一致性（过程解读详见 CXL 章节）

- **低延迟：**超节点中 ScaleUP 网络的时延直接影响计算的同步效率，这就要求系统实现端到端微秒级时延和纳秒级抖动。
    - 端到端微秒级时延：

      超节点 Scale up 网络的亚微秒级延迟要求是支撑 AI 大模型训练和推理的技术底线，其实现依赖于协议栈精简、硬件架构革新和系统协同优化的综合方案。通过内存语义支持、专用交换架构和低延迟 FEC 等关键技术，实现了从传统消息机制的数据包传输到内存直接访问的转变，为超节点数据低延时读写提供了物理基础。
    - 纳秒级抖动：

      超节点中 GPU 等加速器常进行张量并行、专家并行等操作，编译器与运行时会将设备间通信和计算过程重叠优化，此优化依赖稳定可预测的通信延迟。若抖动过大，会打破这种优化平衡，导致计算资源相互等待，训练或推理效率大幅下降，合理要求是将抖动控制在 0.1μs 内。

      控制延迟抖动的实现路径一般围绕硬件架构、协议设计和流量管理等多维度进行优化。如广泛采用的 Flit（Flow Control Unit，流控单元，即定长数据片）机制，配合 Credit 流控机制管控缓冲区，可有效降低避免了拥塞导致的抖动。同时，无损传输是低抖动的底线，频繁的重传虽可兜底数据完

整性，但会引入更多的延迟抖动。因此低抖动要求本质上还需配合无损传输需求，将抖动控制在避免触发频繁重传的范围内，防止重传与抖动形成恶性循环，避免 GPU 或加速器上下文中断等严重问题。

- **高可靠性**：无损传输

Scale up 网络的高可靠性是保障 AI 大模型长时间稳定训练、多 GPU 协同计算数据一致的关键，其可靠性要求聚焦数据一致性、传输稳定性和任务连续性，实现路径则贯穿物理层、链路层等多层架构优化，超节点高密度互联场景下，单丢包可能导致训练任务中断和阻塞，这要求 Scale Up 网络具备无损传输的特性。

表11 主流互联协议要求的 BER

UALink	1e-12（推测）	CRC + 可选重试 + FEC
ETH-X	1e-12（推测）	以太网基础 + 增强可靠性机制
UB	1e-12（推测）	CRC + 多FEC动态切换

为保证 GPU 间低延迟、高带宽数据传输可靠性，Scale up 网络协议的误码率（Bit Error Ratio，BER）要求为 1e-12 或更低。这一标准已成为行业共识，原因如下：

- 维持并行计算节奏稳定：在张量并行、专家并行等计算模式中，反向传播的梯度同步需在微秒级窗口内完成。若网络出现拥塞、丢包或重传等异常，会打乱流水线并行的节奏，原本计算与通信的协同平衡被打破，整体计算效率会大幅下降。
- 确保长时任务持续运行：AI 大模型训练常需数千块 GPU 连续运行数周。一旦网络发生中断，不仅会浪费此前数天的计算资源，还可能导致训练任务彻底无法推进，因此网络需具备极强的稳定性，避免出现任何可能导致任务中断的故障。

3. 内存语义典型实践

英伟达的 NVLink 是内存语义典型代表，通过软硬件协同实现内存语义的核心机制。

硬件层：在每个 GPU 的 MMU（内存管理单元）集成 NVLink 专用的地址翻译模块，识别远端内存地址。使用这种方式将所有连接的 GPU 的本地内存、显存、甚至 CPU 内存映射到同一个虚拟地址空间，实现统一虚拟地址（UVA）。每个 GPU 的 L2 缓存集成 NVLink 一致性代理（Coherence Agent），当一个 GPU 修改某块远端内存时，硬件会通过 NVLink 链路发送“无效化 / 更新”信号，保证其他 GPU 看到最新数据，在硬件层面保证缓存一致性。

软件层：通过 CUDA 将底层语义抽象为简单 API，CUDA 的 nvcc 编译器会根据 NVLink 的内存语义优化代码：比如将“本地写 + 远端读”优化为 NVLink 直接传输，避免冗余同步；自动选择弱一致性模型提升性能。

CXL（Compute Express Link）是由 CXL 联盟制定的开放标准的高速互连协议，是在 PCIe 物理层和电气接口之上，定义了一套全新的、支持缓存一致性和内存语义的高层协议栈。

超节点架构中，内存语义和消息语义协同工作， 如大模型训练中频繁的参数同步操作，无需经过传统的“序列化-网络传输-反序列化”流程，直接通过内存语义通信完成，可满足大模型训练/推理中的小包通信需求，

提升专家网络小包数据传输及离散随机访存通信效率。消息语义负责超节点内设备间的控制指令传递、任务调度与状态同步，保障系统整体协调运行，支撑大数据块传输与跨超节点通信。

消息语义

消息语义指的是传统的、基于显式发送（Sand）/接收（Recv）操作实现数据交换的通信模式，数据在计算节点（如 GPU）之间传输时，被视为一个个独立的“消息”或“包”。消息传递语义的核心是“数据移动”：发送方需将数据序列化（转换为可传输格式）、通过网络协议（如 TCP/IP、RDMA）发送，接收方需反序列化（还原为本地格式）并存储到本地内存。这种模式通信过程是主动且显式的：

- 数据包化：数据需要被封装成特定的消息格式。
- 握手协议：发送方和接收方需要建立明确的通信连接或通道。
- 操作方式：程序员或系统必须明确调用“发送”和“接收”指令。这类似于两个人通过邮局寄信，必须写好信封（封装）、投递（发送）、等待邮差运输、对方签收（接收）。

消息语义常见类型：

- 以太网：技术最成熟，成本低、兼容性好，但是延迟相对较高。
- IB（InfiniBand）：专用高速网络，支持全互联拓扑（任意节点直接通信）、高带宽、低延迟（<1 微秒）、硬件级 RDMA。
- RoCE（RDMA over Converged Ethernet）：基于以太网的 RDMA 技术，兼顾了 InfiniBand 的性能与以太网的低成本，但是配置和调优复杂性高，基于以太网协议存在一定的协议开销。

集合通信

集合通信（Collective Communication）是并行计算 / 分布式系统中多个进程（或计算节点）共同参与的一组通信操作（参考下节通信原语），用于实现进程间的数据交换、同步或聚合。集合通信是为了解决点到点通信在多节点场景下 “编程复杂、性能低下、同步困难” 的问题。现代高性能网络硬件开始原生支持某些集合通信操作。例如，InfiniBand 网络和 NVSwitch 能够在交换机层面直接执行归约操作，大幅降低了 CPU 开销和通信延迟。

集合通信的概念起源于高性能计算（HPC）领域，其标准接口规范由 MPI（消息传递接口，Message Passing Interface）定义。随着深度学习的发展，它已成为分布式训练的核心技术。

通信原语

集合通信最为基础的操作有发送、接收、复制、节点间进程同步等，这些基本的操作经过组合构成了一组通信模板，也称为通信原语。

原语名称	核心功能	典型场景
广播（Broadcast）	1个主进程将数据发送给组内所有其他进程	模型参数分发、配置同步
散射（Scatter）	根进程将一份数据分片，分发给组内所有进程	数据并行的任务分片分发
聚集（Gather）	所有进程将本地数据发送给根进程，根进程汇总	任务结果收集、分片数据合并

原语名称	核心功能	典型场景
归约（Reduce）	所有进程将本地数据按指定规则（求和 / 最大值）计算，结果返回根进程	梯度求和、统计指标计算
全归约（All-Reduce）	归约的基础上，将最终结果分发给所有进程	分布式训练的全局梯度同步
全聚集（All-Gather）	汇总所有进程将本地数据到全部进程	模型并行参数同步
归约散射（Reduce-Scatter）	先归约再将数据分片，分发给组内所有进程	张量并行梯度分摊
全交换（All-to-All）	进程间全对全数据交换	MoE（专家混合）模型中的数据分发
屏障同步（Barrier）	所有进程到达该点后等待，直到全部到达再继续	多节点阶段执行同步

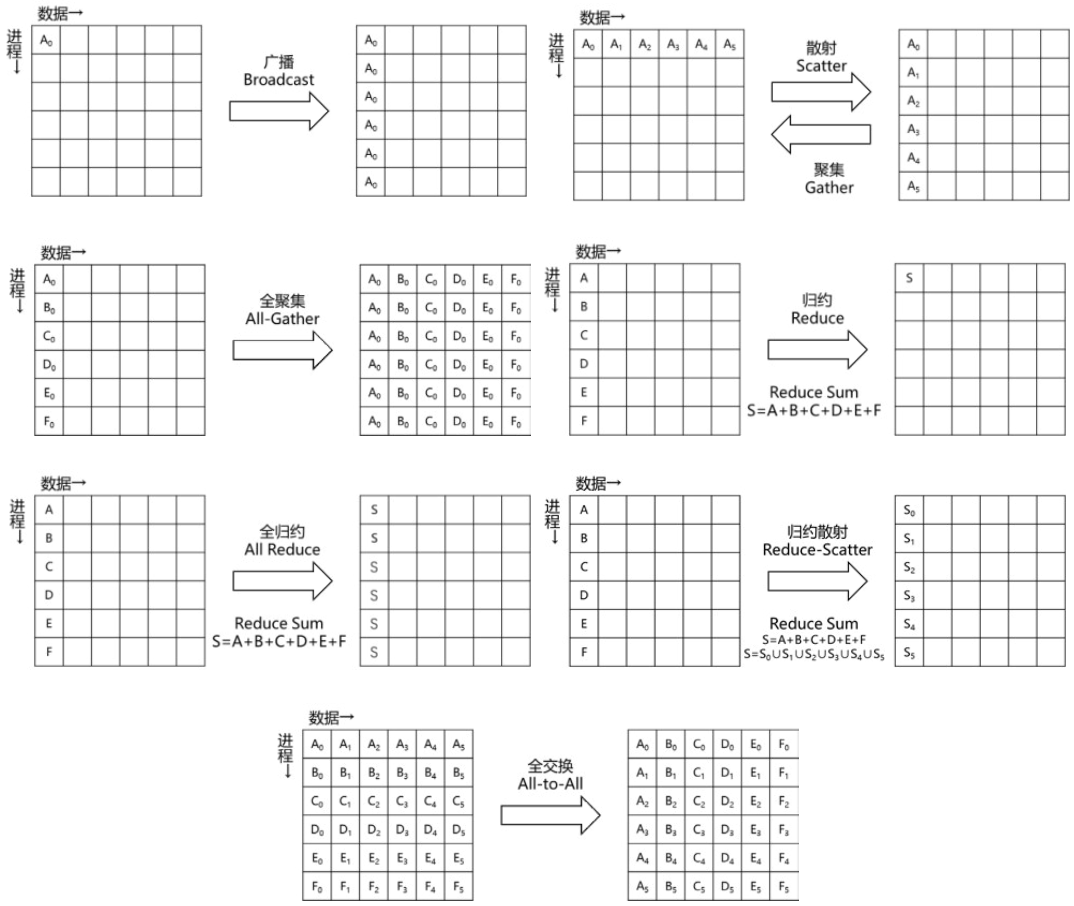


图69 通信原语示意图

2.4 总结

事务层的为协议栈顶层核心，承担语义转换与事务抽象、报文封装解析、全生命周期调度、端到端传输保障（部分协议由传输层实现此功能）等核心功能，为上层应用屏蔽底层传输细节，提供标准化的跨节点内存 / 显存访

问抽象。同时，事务层为集合通信提供底层支撑，满足大模型分布式训练的数据同步与聚合需求。是超节点实现高性能高可靠运行的核心基础。

3 主要Scale up协议介绍

3.1 PCIe 与 CXL 协议介绍

总线及 PCIe 的发展历程和技术演进

在计算机系统的发展长河中，内部组件之间的互联技术始终是推动性能跃升的核心动力。在 PCI Express（PCIe）成为主流之前，计算机经历了从早期并行总线到高速串行互连的深刻变革。这段历史记录了技术和工程设计在带宽、延迟、可扩展性与信号完整性之间不断寻求平衡的智慧。

计算机内部互联的雏形可追溯至 20 世纪 80 年代。1981 年，IBM 在其 PC XT 机型中引入了 ISA（Industry Standard Architecture）总线，这是最早的标准化扩展接口。最初的 8 位 ISA 提供约 4.77 MB/s 的带宽，随后 16 位版本将带宽提升至 8 MB/s。尽管在当时已属先进，但随着图形处理和数据存储需求的增长，ISA 的带宽迅速成为瓶颈。

为应对这一问题，IBM 于 1987 年推出了 MCA（Micro Channel Architecture），试图通过 32 位宽度和更高时钟频率提升性能。MCA 在技术上优于 ISA，但由于其封闭性和高昂成本，未能广泛普及。与此同时，行业开始寻求开放、兼容且高性能的替代方案。

为打破 IBM 在 MCA 上的事实主导，以康柏（Compaq）为首的九家 IBM 竞争对手迅速联合，于 1988 年推出 EISA 总线（扩展工业标准架构），在保持 32 位性能的同时，完全兼容 ISA 设备，赢得了广泛支持。但其架构陈旧，性能提升有限，很快被后起的 PCI 总线替代。

同期，在运行图形用户界面和简单三维游戏时，显卡性能被严重束缚。为解决这一瓶颈，视频电子标准协会（VESA）于 1992 年推出了 VLB 总线标准。它并非取代 ISA，而是作为其高速扩展，直接连接到 80486 处理器的本地总线，从而实现与 CPU 近距离数据交换。因后续奔腾（Pentium）处理器，其总线架构与 80486 截然不同，VLB 无法直接适配，该技术也迅速没落。

1991 年，Intel 提出 PCI（Peripheral Component Interconnect）总线标准，并联合多家厂商成立 PCI-SIG（PCI 特殊兴趣组），推动其标准化与普及。PCI 采用 32 位并行架构，工作频率为 33 MHz，理论带宽达 133 MB/s，显著优于 ISA 和 MCA。它支持即插即用、多设备共享总线，并迅速成为台式机、服务器和工作站的主流扩展接口。

随着应用需求进一步提升，特别是服务器领域对高吞吐量的需求，PCI 的带宽再次面临挑战。为此，业界在 1998 年推出了 PCI-X（PCI Extended），将总线宽度扩展至 64 位，时钟频率提升至 133 MHz，最大带宽可达 1 GB/s。PCI-X 在服务器和高端工作站中得到广泛应用，成为 PCI 的高性能延伸。

其间，20 世纪 90 年代中期，随着 3D 游戏、三维建模和多媒体应用的兴起，图形处理对带宽的需求呈指数级增长。传统的 PCI 总线采用共享架构，所有外设争用带宽，且其 32 位/33MHz 的规格仅提供约 133MB/s 的带宽，已无法满足显卡对大量纹理、帧缓冲数据传输的迫切需求。为突破这一瓶颈，英特尔公司于 1996 年 7 月正式推出 AGP 总线标准。AGP 并非替代 PCI，而是作为其补充，专为图形加速卡设计。它基于 PCI 2.1 规范进行扩展，采用点对点连接方式，直接连接显卡与北桥芯片，从而独享通道，避免了 PCI 的带宽争用问题，但该技术也随着 PCIe 的成熟而被替代。

尽管 PCI 和 PCI-X 在当时取得了巨大成功，但它们基于并行总线架构的本质缺陷逐渐显现：

- **信号完整性问题：**随着频率提升，并行线路间的串扰、时钟偏移（skew）和电磁干扰日益严重，导致数据误码率上升。
- **共享总线瓶颈：**多个设备共享同一总线，容易产生资源竞争，无法实现真正的并行通信。
- **扩展性差：**增加设备数量会降低整体性能，难以支持未来高速设备（如 GPU、SSD）的需求。
- **功耗与布线复杂：**大量并行信号线增加了 PCB 设计难度和系统功耗。

为解决这些问题，Intel 于 2001 年提出新一代 I/O 架构——3GIO（Third Generation I/O），并于 2003 年正式命名为 PCI Express。PCIe 摒弃了传统的并行总线，转而采用高速串行、点对点的连接方式，每个链路由独立的差分信号对组成，彻底摆脱了并行总线的时钟同步和信号干扰的困扰。

PCI Express(PCIe)自诞生以来，已历经二十余年发展，从最初的串行总线标准演进成为现代计算平台的核心互连技术。这一演进路径体现了从传统并行总线向高速串行架构的革命性转变，同时通过持续提升带宽、降低延迟、增强可靠性和可扩展性，为现代计算架构提供了高效、灵活的互连解决方案。以下为总线历史的大致发展路径。

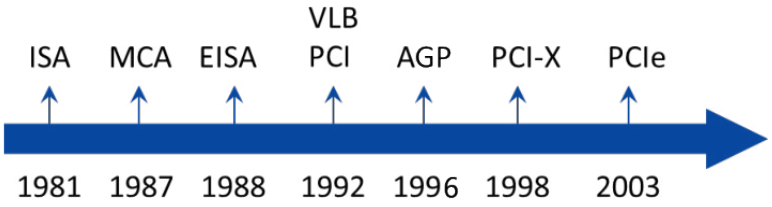


图70 PCIe 前计算机总线发展

作为当前应用最广泛使用的高速互联标准，PCIe 凭借成熟、标准化、高性能的架构设计，其若干特性被其他高速总线和 Scale up 协议借鉴。主要影响在于如下方面：

- **分层架构思想：**事务层/链路层/物理层解耦设计。
- **链路管理机制：**链路训练与状态机、热插拔、电源管理等。
- **可靠性基础：**CRC 校验、ACK/NAK 重传（LLR-Link Layer Retransmission，链路层重传）、流量控制（CBFC－Credit-Based Flow Control，基于信用的流量控制）。

1. PCIe 的版本演进与带宽提升

下表展示了 PCIe 各版本的核心参数与技术演进。每一代 PCIe 的带宽基本实现翻倍，同时通过改进编码方式和信号处理技术，显著提高了带宽利用率。这种持续演进不仅满足了当前应用需求，也为计算架构创新预留了发展空间。

表12 PCIe 各版本核心参数与技术演进

PCIe 版本	规范发布时间	单通道速率	x16 双向带宽	调制及编码方式	核心技术改进
1.0	2003年	2.5 GT/s	8 GB/s	NRZ+8b/10b	串行架构替代并行总线，点对点连接设计
2.0	2007年	5.0 GT/s	16 GB/s	NRZ +8b/10b	速率翻倍，链路训练优化，兼容性提升
3.0	2010年	8.0 GT/s	32 GB/s	NRZ +128b/130b	编码效率提升，链路训练增强，低功耗状态管理
4.0	2017年	16.0 GT/s	64 GB/s	NRZ +128b/130b	速率翻倍，电气规范强化，
5.0	2019年	32.0 GT/s	128 GB/s	NRZ +128b/130b	速率翻倍，电气规范升级，
6.0	2022年	64.0 GT/s	256 GB/s	PAM4+1b1b	PAM4调制技术，固定大小Flit编码，低延迟FEC
7.0	2025年	128.0 GT/s	512 GB/s	PAM4+1b1b	光互连支持，增强FEC，能效比优化
8.0	2028年(预计)	256.0 GT/s	1024 GB/s	待定	可能探索PAM6/PAM8 或新型调制方案

PCIe 1.0 作为首个串行 PCI 标准，解决了传统 PCI 并行总线的信号衰减和时钟同步问题，通过点对点串行连接为后续发展奠定了基础。PCIe 6.0 是 PCIe 标准诞生以来变化最大的版本，引入了 PAM4 调制和固定大小 Flit 编码，使单通道速率从 32 GT/s 提升至 64 GT/s，同时通过轻量级前向纠错(FEC)技术显著降低了误码率和延时。PCIe 7.0 和 8.0 则进一步将速率提升至 128 GT/s 和 256 GT/s，并引入光互连支持和新型连接器技术，为超大规模数据中心和边缘计算提供了更强大的互连能力。

2. PCIe 架构的核心设计理念

PCIe 的架构设计遵循分层原则，通过事务层、数据链路层和物理层的三层结构实现了高效、可靠且易于扩展的互连系统。这种分层设计使 PCIe 能够无缝集成到现有计算架构中，同时为未来创新提供灵活性。

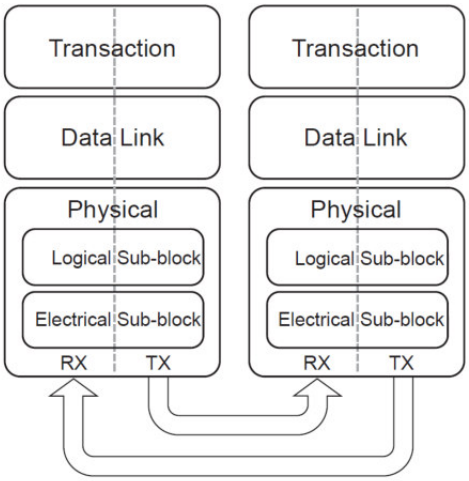


图71 PCIe 分层结构 [54]

事务层（Transaction layer）： PCIe 协议栈的最高层，负责定义事务类型和流控制机制，为上层软件提供统一的编程接口。

数据链路层（Data Link Layer）： 位于事务层和物理层之间，主要负责确保数据包在物理链路上的可靠传输，通过序列号和 LCRC 校验码实现选择性重传。

物理层（Physical Layer）： PCIe 的最底层，直接与物理媒介打交道，负责比特流的发送和接收，通常被细分为 PCS（物理编码子层）、PMA（物理媒介接入子层）和 PMD（物理媒介相关子层）三个子层。

这种分层架构使 PCIe 能够同时满足高性能与高可靠性的需求，为显卡、存储设备、网卡等外设提供了远超传统 PCI 的传输带宽与更低延迟，也更进一步扩展用途，实现了除外设互联外的计算节点 Peer to Peer 互联。目前一些超节点方案即使用 PCIe P2P（PCIe fabric）技术来实现。Scale up 分层架构的详细讲解可参见前 Scale up 协议核心技术解析章节。

PCIe 的分层架构分析

事务层

事务层是 PCIe 协议栈的最高层，负责定义事务类型和流控制机制，为上层软件提供统一的编程接口。其核心功能包括：

TLP 构造与解析： 将读写请求、完成包等封装成事务层数据包(TLP)，如 Memory Write、Memory Read Request 等。TLP 包含头信息和有效载荷，头信息定义了事务类型、地址、源/目标标识等关键信息。这种封装方式使事务层能够抽象化底层物理链路的复杂性，为操作系统和驱动程序提供与传统 PCI 相似的编程模型。

流量控制机制： 采用信用(Credit)机制确保发送方不会因接收方缓冲区溢出而导致数据丢失。每个接收端维护发送端的信用计数器（配合链路层实现），发送方只有在获得足够信用时才能发送数据，从而实现无损传输。这种机制在 PCIe 6.0 中进一步优化，引入了固定大小的流控制单元(FLIT)，提高了带宽利用率。

事务排序规则：定义了不同类型的内存、I/O 和配置事务之间的排序规则，保证系统的一致性和正确性。例如，Memory Read 完成包必须按请求顺序返回，而 I/O 写操作可能允许乱序完成。这种排序机制确保了多设备并发访问共享资源时的数据一致性。

地址路由方式：支持基于地址、ID 和隐式等多种路由方式，使 PCIe 能够灵活适应不同拓扑结构。在树状拓扑中，路由通常由根复合体控制；而在点对点（Peer to Peer）拓扑中，路由可能由交换机或终端设备自主完成。这种灵活性使 PCIe 能够支持从简单的台式机到复杂的服务器集群的各种应用场景。

数据链路层

数据链路层位于事务层和物理层之间，主要负责确保数据包在物理链路上的可靠传输。其核心技术包括：

数据包可靠性：为每个 TLP 添加序列号和 LCRC 校验码，形成数据链路层数据包(DLLP)。序列号用于追踪数据包顺序，LCRC 用于检测数据包传输过程中的错误。这种机制确保了即使在物理链路上可能出现比特错误或数据包丢失的情况下，数据链路层仍能保证数据的完整性和正确性。

ACK/NAK 协议：接收方通过发送 ACK(DLLP)确认正确接收的数据包，或通过 NAK(DLLP)请求重传错误或丢失的数据包。这是一种选择性重传机制，仅重传出错或丢失的数据包，提高了传输效率。

流量控制扩展：除事务层的信用机制外，数据链路层还提供额外的流量控制机制，如流量控制信用计数器和流控制状态机，确保链路不会过载。数据链路层的流量控制与事务层的信用机制形成互补，共同保障了数据传输的可靠性。

错误恢复机制：支持多种错误恢复策略，如自动重试、链路重训练等，确保在物理链路出现不稳定时，系统仍能保持高可靠性。这些机制在 PCIe 6.0 中进一步增强，通过低延迟 FEC 技术实现了更高效的错误检测与纠正。

数据链路层的核心目标是为事务层提供一个可靠、有序、无差错的通信通道。通过强大的错误检测和恢复机制，数据链路层确保了 PCIe 在各种应用场景下的高可靠性。

物理层

物理层是 PCIe 的最底层，直接与物理媒介打交道，负责比特流的发送和接收。它通常被细分为三个子层：PCS（物理编码子层）、PMA（物理媒介接入子层）和 PMD（物理媒介相关子层）。这三个子层协同工作，实现了 PCIe 的高速串行通信。

1. PCS 层（物理编码子层）

PCS 层负责将事务层和数据链路层的并行数据转换为适合物理传输的格式。其核心技术包括：

- 编码与解码：PCIe 1.0-3.0 采用 8b/10b 编码，PCIe 4.0-5.0 采用 128b/130b 编码，而 PCIe 6.0 则采用 1b/1b 编码配合 PAM-4 调制。这些编码方式在提高带宽利用率的同时，也保证了足够的跳变用于时钟恢复。例如，PCIe 6.0 的 1b1b 编码使带宽利用率从 PCIe 5.0 的约 85%提升至约 96%。
- 链路训练功能：管理通道对齐(Lane Deskew)和极性反转(Polarity Inversion)等链路训练功能，确保物理链路的稳定性和可靠性。在链路训练过程中，PCS 层发送 TS1/TS2 训练序列，与对端设备协商链路速率和宽度。

- 速率协商：支持不同速率的设备间自动协商，确定最佳工作速率，提高了系统兼容性和灵活性。例如，PCIe 6.0 设备可以与 PCIe 5.0 设备连接，但链路速率会协商至 PCIe 5.0 支持的 32 GT/s。

PCS 层的特点是实现了逻辑与物理的映射，通过编码技术将高可靠性要求的数据包转换为适合高速传输的格式，同时保留了足够的冗余信息用于错误检测和恢复。

2. PMA 层（物理媒介接入子层）

PMA 层包含串行器/解串器(SerDes)，负责将并行数据转换为高速串行比特流。其核心技术包括：

- 串并转换：在发送端，将来自 PCS 的并行数据流转换为高速串行比特流；在接收端，将接收到的高速串行比特流转换回并行数据流供上层处理。PMA 层的 SerDes 技术是实现 PCIe 高速传输的核心。
- 时钟数据恢复(CDR)：从接收到的串行数据流中提取嵌入的时钟信号，实现时钟和数据的同步恢复，这是高速串行通信的关键技术。CDR 技术使 PCIe 能够在不传输独立时钟信号的情况下保持同步，显著降低了信号传输的通道需求。
- 链路初始化与训练：负责物理链路的初始化、速率协商和均衡设置，确保链路能够以最佳状态运行。在 PCIe 6.0 中，PMA 层进一步优化了均衡技术，支持更长距离的信号传输。

PMA 层的核心功能是实现高速串行通信的基础，通过 SerDes 技术将数据速率提升至传统并行总线难以企及的水平，同时通过 CDR 技术解决了时钟信号传输的难题。

3. PMD 层（物理媒介相关子层）

PMD 层直接适配物理介质，是物理层中最接近实际传输媒介的部分。其核心技术包括：

- 信号驱动与接收：包含驱动器(Driver)和接收器(Receiver)电路，负责将数字信号转换为适合在特定物理介质上传输的模拟信号。例如，在铜缆传输中，PMD 层需要调整驱动器的输出电压以补偿信号衰减。
- 介质适配：根据物理介质（如铜缆、光纤）的不同特性，调整信号参数和传输方式。例如，铜缆需要预加重(Pre-emphasis)和去加重(De-emphasis)来补偿高频损耗；而光纤则需要光电转换器来实现光信号传输。
- 信号完整性保障：实现预加重、去加重和接收均衡平等技术，补偿高速信号在传输过程中的损耗和失真，确保信号完整性。在 PCIe 6.0 中，PMD 层进一步优化了这些技术，支持更长距离的传输。
- 电气特性控制：管理电压摆幅、阻抗匹配等电气特性，确保信号在物理介质上的稳定传输。例如，PCIe 7.0 的 PMD 层支持更高的电压摆幅，以补偿光互连中的信号损耗。

PMD 层的设计高度依赖于所使用的物理介质，通过优化模拟信号特性，使 PCIe 能够在不同传输距离和环境条件下保持高可靠性和高带宽。这种设计使 PCIe 能够灵活适应铜缆、光纤等多种物理传输介质。

4. 物理层子层的协同工作

PCS、PMA 和 PMD 三个子层通过紧密的协作共同实现了 PCIe 的高速串行通信：

- 发送端流程：来自数据链路层的并行数据首先由 PCS 层进行编码（如 8b/10b 或 128b/130b 编码），生成编码后的并行数据流；然后 PMA 层将这些并行数据转换为高速串行比特流，并通过 CDR 技术实现时

钟恢复；最后，PMD 层根据物理介质的特性（如铜缆或光纤）调整信号驱动参数，将串行信号发送到物理介质上。

- 接收端流程：物理介质上的信号首先由 PMD 层接收并转换为数字信号，根据介质类型可能需要光电转换（光纤）或简单的电平转换（铜缆）；然后 PMA 层将串行信号转换为并行数据流，并通过 CDR 技术恢复时钟信号；最后，PCS 层对接收的并行数据进行解码，恢复原始数据并传递给上层协议处理。
- 链路训练过程：在链路训练阶段，PCS 层发送训练序列（如 TS1/TS2），协商链路速率和通道数量；PMA 层根据协商结果调整 SerDes 参数（如电压摆幅、CDR 带宽）；PMD 层根据物理介质特性（如铜缆长度、光纤衰减特性）设置信号驱动强度（如预加重水平），并通过回环测试验证信号完整性，将结果反馈给 PMA 和 PCS 层进行进一步调整。

这种分层设计使 PCIe 能够灵活适应不同的物理介质和传输环境，同时保持协议栈的统一性和兼容性。PCS、PMA 和 PMD 层的协同工作确保了 PCIe 能够在从几厘米到几十米的多种距离范围内实现高速、可靠的串行通信。

PCIe 工作过程

1. PCIe 初始化：设备枚举与配置

PCIe 系统启动后，首先由 BIOS/OS 完成初始化，核心是设备枚举与配置空间配置：

枚举阶段：BIOS/OS 通过 PCIe 根复合体（Root Complex）发起扫描，识别总线上所有 PCIe 设备（如显卡、网卡），为每个设备分配唯一的总线号、设备号、功能号（BDF），建立设备拓扑结构。

配置阶段：每个 PCIe 设备都有独立的配置空间（用于存储设备信息），初始化时 BIOS/OS 读取配置空间的设备 ID、厂商 ID 等信息，确定设备类型与功能，同时配置设备的传输参数（如链路宽度、速率），完成设备与根复合体的链路协商，确保设备正常通信。

2. PCIe 寻址：地址映射与解析

初始化完成后，PCIe 通过地址映射实现设备间寻址，分为两步：

地址分配：BIOS/OS 为每个 PCIe 设备分配独立的物理地址空间（避免地址冲突），该地址空间被映射到系统的物理内存地址域。

地址解析与访问：当 CPU / 其他设备需要访问 PCIe 设备时，先将目标设备的虚拟地址转换为系统物理地址，根复合体根据物理地址中的 BDF 信息，定位到目标 PCIe 设备，再通过 PCIe 链路发起访问请求，实现数据交互。

3. PCIe Peer to Peer 工作模式：

在 PCIe 应用于 Scale up 领域中时，常使用 Peer to Peer 工作模式，也称为 P2P 对等模式，该模式的核心是 PCIe 设备（GPU）间无需 CPU / 根复合体中转，直接通过 PCIe Switch 传输数据，简化流程、降低时延，工作过程简要如下：

- 初始化与地址映射：系统启动时，BIOS/OS 完成初始化，为参与 P2P 传输的设备（如 GPU）分配独立 PCIe 地址空间，并配置 PCIe Switch 完成地址映射，确保各设备可直接识别对方的 PCIe 地址，无需 CPU 介

入地址转换。该过程确定的地址和路由后，一般维持静态不再变动。后续的寻址和路由都由此配置进行。此过程和以太交换有较大的区别。

- 传输发起：发起端设备（如 GPU1）直接向目标端设备（如 GPU2）发起访问请求，携带目标设备的 PCIe 地址及传输指令（读 / 写）。
- 直接传输：请求通过 PCIe Switch 直接送达目标端，地址转换为目标地址，目标端验证地址合法性后，直接响应请求，双方通过 PCIe 链路完成数据直接交互。
- 传输完成：数据传输结束后，两端设备可向 CPU 反馈传输状态（用于系统监控），全程无 CPU 中转，大幅降低传输时延、节省 CPU 资源。

PCIe P2P 作为较为成熟的 Scale up 技术，使用该技术构建的超节点在当前超节点架构中具备一定代表性，可实现统一编址、内存语义访问和低时延等超节点关键特性，但互联带宽受到 PCIe 自身带宽的限制，以及不支持 Cache 一致性等，和 NVLink 相比还有较大差距。

CXL 技术简介

CXL（Compute Express Link）是一种建立在 PCIe 物理和电气规范之上的开放互连技术，旨在解决异构计算中的内存一致性、资源共享和池化问题，所以和 PCIe 章节放在一起进行说明。与 PCIe 相比，CXL 不仅关注带宽和延迟，更强调内存一致性、资源共享和安全性，为高性能计算和 AI 应用提供了更强大的互连解决方案。

CXL（Compute Express Link）是由 CXL 联盟（Intel 主导，主要大厂均参与）制定的开放高速互连标准，核心目标为保持 PCIe 物理层兼容的前提下，实现计算、内存、存储资源的高效协同与池化，解决 AI/HPC 时代“内存墙”与资源孤岛问题。主要解决 PCIe 仅作为数据通道，不能提供从内存资源协同的问题，即解决了远端内存的硬件缓存一致性、直接 Load\Store 访问、池化共享问题。

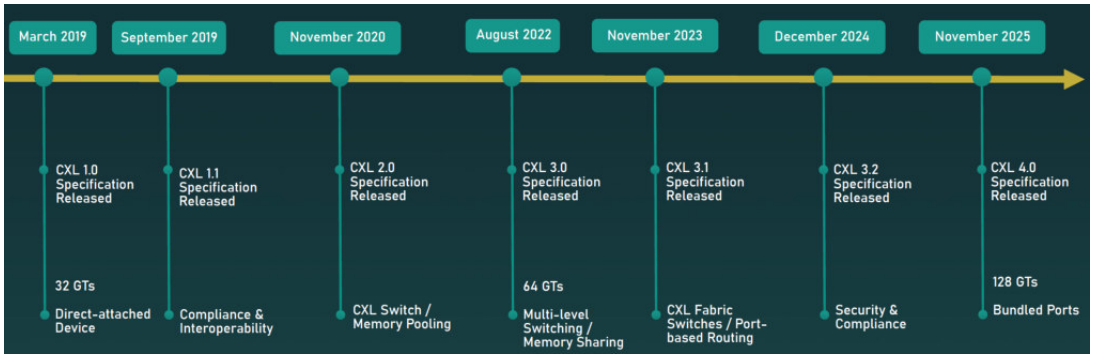


图72 CXL 版本发布时间线 [55]

如下图，CXL 技术的出现，驱动计算架构从“单机集成”迈向“资源服务化”转变，实现三大突破：

资源解耦池化：内存（Type 3 设备）、计算（Type 2 加速器）可跨服务器构建共享池，单机内存瓶颈被打破，资源利用率提升；

异构协同简化：CPU 与 GPU/FPGA 通过硬件一致性无缝协作，消除数据拷贝与软件同步开销，训练通信延迟显著降低；

分解式架构落地：计算、内存、存储独立扩展与动态调度，优化数据中心 TCO，支撑云原生与大模型弹性需求。

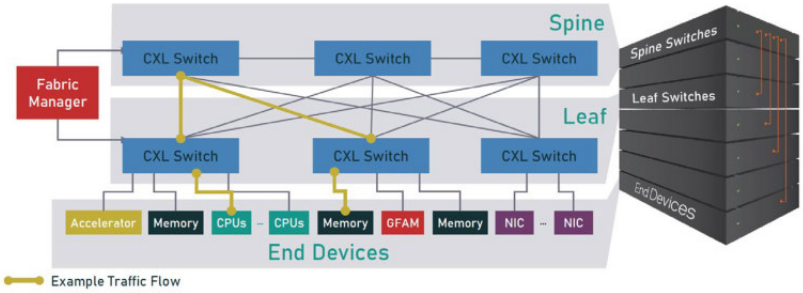


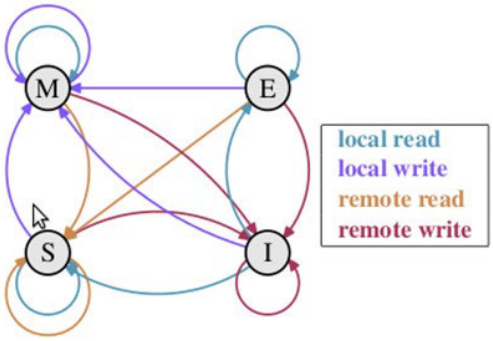
图73 使用 CXL 技术实现的计算资源池化 [56]

CXL 协议

Cache 一致性

缓存一致性是多核与异构计算架构的效率基石，硬件层面通过协议（如 MESI）实时同步各处理单元私有缓存中共享数据的状态：在多核系统中，当任一核心修改数据，其他核心的 Cache 旧副本自动失效或更新，确保所有处理器看到统一、最新的内存视图。一致性的核心价值在于：

- 保障正确性——杜绝脏读、数据竞争等逻辑错误；
- 提升性能——减少冗余主存访问，加速数据共享；
- 简化编程——硬件自动维护一致性，开发者无需软件编码刻意维护一致性保证。



M（已修改，Modified）：本地处理器已经修改缓存行，即是脏行，它的内容与内存中的内容不一样，并且此 cache 只有本地一个拷贝(专有)；
E（专有，Exclusive）：缓存行内容和内存中的一样，而且其它处理器都没有这行数据；
S（共享，Shared）：缓存行内容和内存中的一样，有可能其它处理器也存在此缓存行的拷贝；
I（无效，Invalid）：缓存行失效，不能使用。

图74 MESI 协议

MESI 状态迁移与操作处理

1）本地操作（Local Operations）

本地操作指当前 CPU 核心发起的读、写请求，是状态迁移的主要触发源。

- 本地读（PrRd）
  - I 状态：缓存缺失，发送 BusRd 总线请求，广播查询其他缓存。
    - 其他缓存有有效副本（E/S）：数据返回，当前状态→S。
    - 无其他副本：从主存读取，状态→E。
  - E/S/M 状态：缓存命中，直接读取，状态不变。
- 本地写（PrWr）
  - I 状态：缓存缺失，发送 BusRdX（读所有权），获取数据并无效其他副本，修改后状态→M。
  - S 状态：发送 BusUpgr（升级），无效其他核心副本，修改后状态→M。
  - E 状态：直接修改，无需总线，状态→M。
  - M 状态：直接修改，状态不变。

2）远程操作（Remote Operations）

远程操作指其他 CPU 核心发起的总线请求，当前核心通过嗅探响应并更新状态。

- 远程读（BusRd）
  - M 状态：先将脏数据写回主存（BusWB），再将数据发给请求方，状态→S。
  - E 状态：发送数据，状态→S。
  - S 状态：发送数据，状态不变。
  - I 状态：无响应。
- 远程写（BusRdX/BusUpgr）
  - M 状态：先写回主存，然后本地缓存失效，状态→I。
  - E/S 状态：本地缓存失效，状态→I。
  - I 状态：无响应。

MESI 协议中，Snoop（窥探）是保障多核心/多设备缓存一致性的核心机制，核心作用是监控总线或互联链路中的缓存操作，同步各核心/设备的缓存行状态。当某核心对缓存行执行读写、修改操作时，Snoop 会广播该操作至所有相关核心及设备，各核心原理是根据自身缓存行的 MESI 状态（修改态、独占态、共享态、无效态）响应，确保同一缓存行在不同节点中状态一致，避免脏数据读取、数据不一致等问题。

在 CXL 所涉及的多 CPU、多 TYPE 1\2\3 设备协同场景中，Snoop 是实现全局缓存一致性的基础，但其无差别的 Snoop 广播极易产生大量的无效窥探、占用链路带宽、增加时延，所以需要对传统 Snoop 广播进行改进。

在 Intel 的早期 CPU 设计中，在多核场景，同样需通过 Snoop 机制保障缓存一致性（遵循 MESI 协议），但大量无效窥探（如访问不同缓存行、无数据依赖的窥探请求）会占用 CPU 内部互联链路带宽、增加核心处理

开销，影响低时延传输性能。同样在多路（多 CPU）服务器中，为了保持各个 CPU 插槽间缓存（Cache）的一致性，系统需要通过互联链路（如 Intel 的 UPI 或 AMD 的 Infinity Fabric）发送监听请求（Snoop Requests）。

随着核心/CPU 数量增加，广播式的监听会产生巨大的通信流量，导致互联带宽被大量垃圾流量占据，增加访问延迟。

Intel 在 Nehalem 架构时引入了 Snoop Filter（SF）机制，在 CPU 互连子系统中增加硬件目录（Directory），即 Snoop Filter。其充当一个目录，记录了哪些缓存行存储在哪个 CPU 中，当发生内存访问时，它能精确判断是否需要跨 CPU 监听，从而避免无效的广播。此架构后续不断演进，在最新至强 6（Intel Xeon 6）CPU 一致性保证机制中仍占据关键地位。

SF 的关键作用和价值在于：

- 记录哪些缓存行被哪些核心/CPU 缓存及其 MESI 状态
- 消除全系统广播 Snoop（Broadcast Snoop）
- 将 Snoop 请求精准定向至目标单元（如仅通知持有副本的 CPU）

Intel 宣称优化的 Snoop Filter 可以过滤掉 90% 以上 的无效监听。这意味着 CPU 不再需要等待每个远端 CPU 返回确认信号，直接将本地访问延迟降低，对性能的提升有极大帮助。

CXL（Compute Express Link）技术的诞生，核心是为解决异构计算兴起带来的设备互联瓶颈。随着 AI、HPC 等场景发展，CPU 需与 GPU、FPGA 等加速器高效协同，但传统 PCIe 协议存在带宽不足、时延较高、缓存一致性难以保障的短板，且不同厂商加速器接口不统一，导致设备互联兼容性差、资源利用率低。解决的思路也很清晰：在 PCIe 的基础上，引入多核 CPU 的 Cache 硬件一致性机制，使用 PCIe 数据报文来承载 Snoop 功能，使远端内存（TYPE 1、2、3）能和 Host 保持缓存一致性。主要体现在：

- 将总线 Snoop 机制扩展到 PCIe/CXL 互联设备（TYPE 1、2、3 设备）
- 修改、扩展 PCIe 数据帧结构为新的 CXL 帧结构，兼容原帧结构
- 将 Snoop 及应答 Ack 消息封装在 CXL 帧结构中
- 使用 Snoop Filter 过滤无效的 Snoop

CXL 总线在 PCIe 总线上针对一致性访问需求进行了如上改进，并保持了对 PCIe 总线的兼容，修改后的总线命名为 Flex 总线。

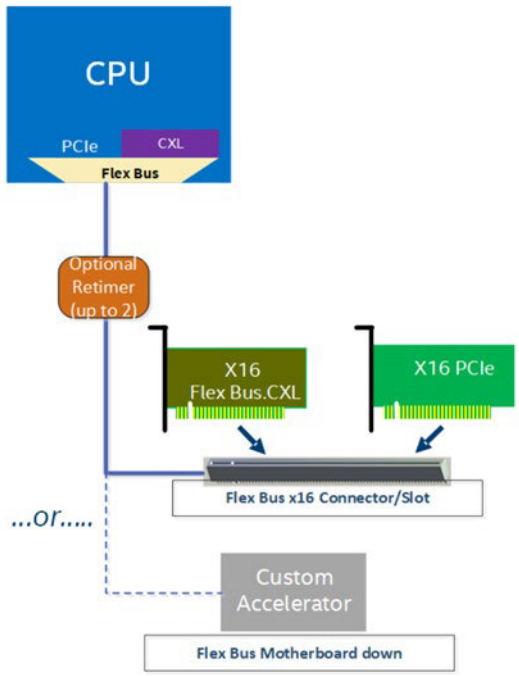


图75 Flex Bus[57]

总线关键特性：

- Flex 总线电气特性等同于 PCIe
- 在 PAM4 调制下，每 lane 能达到 64GT/s 或 128GT/s 传输速率（CXL 3.0/PCIe 6.0 和 CXL4.0/PCIe 7.0）
- CXL 模式下总线宽度支持 x16，x8，x4，x2，x1 模式
- 提供 PCIe 点对点传输或 CXL I/O，Caching，memory3 个子协议，子协议功能主要为：
  - CXL.io：PCIe 兼容 I/O 协议（配置/初始化）
  - CXL.cache：硬件级缓存一致性（CPU<-> 加速器）
  - CXL.mem：低延迟字节级内存访问（主机 <-> 设备内存）

CXL 协议同样和 PCIe 一样进行了分层结构，保留了 PCIe 兼容设计：

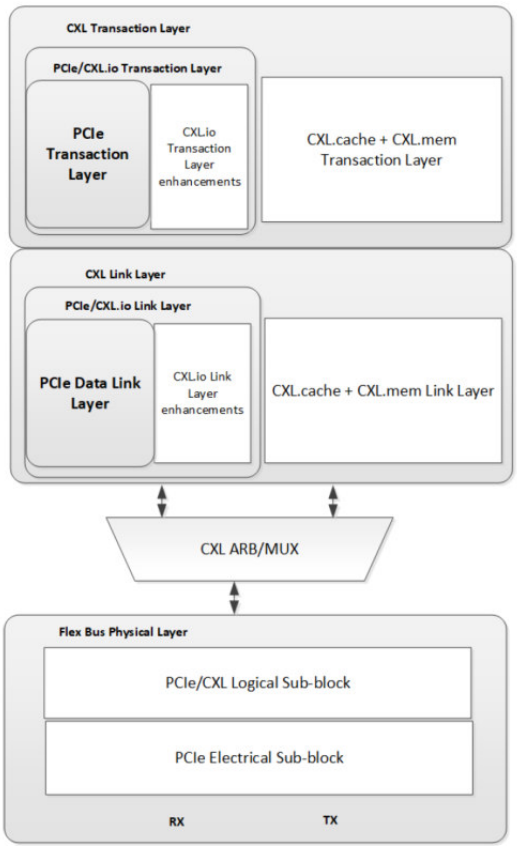


图76 CXL 使用和 PCIe 一样的分层设计 [58]

表13 CXL 协议对 PCIe 协议的扩展和复用情况

PCIe 层级	CXL 层级	CXL 关键技术复用细节
物理层(PHY)	全复用	• 电气特性、连接器、速率（PCIe 5.0/6.0 完全复用 • PAM4 信令、FEC、SerDes 等完全继承
数据链路层	大部分复用	• 链路训练流程扩展协商字段（序列中新增 CXL Capability 标识） • ACK/NAK、CRC、流量控制和 PCIe 机制保持一致
事务层	协议多路复用	• CXL.io：直接复用 PCIe TLP 格式（配置/中断/DMA），OS 驱动零修改 • CXL.cache/mem：定义新型 TLP（含 Snoop 请求、内存 Load/Store 等） • 通过 TLP 头中 Format 字段区分流量类型(PCIe TLP 及 CXL TLP)

按 CXL 接入设备的主要功能，设备按 TYPE 1、2、3 分为 3 类，定义如下：

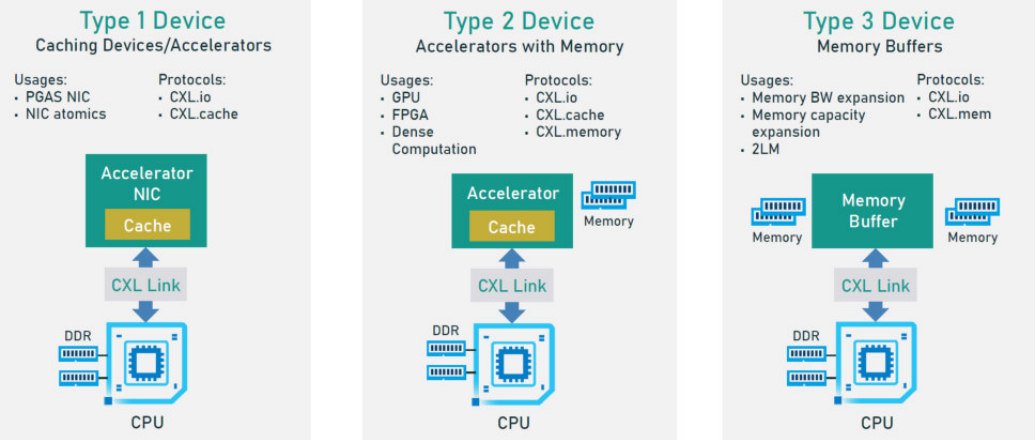


图77 CXL 设备分类 [59]

- TYPE 1：仅有处理器，无内存，或只有私有内存的设备。设备处理器可以访问主机内存，如智能 NIC。
- TYPE 2：有处理器，可以访问主机内存，自有内存可以被主机访问，如 GPU/DPU/FPGA。
- TYPE 3：无处理器的内存扩展设备。扩展内存可被一个或多个主机访问，主要形态为池化内存。

2025 年 11 月，CXL 联盟发布了当前最新版本 CXL 4.0 规范（基于 PCIe 7.0），CXL 各版本主要规格如下：

Features	CXL 1.0 / 1.1	CXL 2.0	CXL 3.x	CXL 4.0
Release date	2019	2020	2022-2024	2025
Max link rate	32GTs	32GTs	64GTs	128GTs
Flit 68 byte (up to 32 GTs)	✓	✓	✓	✓
Flit 256 byte (up to 64 GTs)			✓	✓
Type 1, Type 2 and Type 3 Devices	✓	✓	✓	✓
Memory Pooling w/ MLDs		✓	✓	✓
Global Persistent Flush		✓	✓	✓
CXL IDE		✓	✓	✓
Switching (Single-level)		✓	✓	✓
Switching (Multi-level)			✓	✓
Direct memory access for peer-to-peer			✓	✓
Enhanced coherency (256-byte flit)			✓	✓
Memory sharing (256-byte flit)			✓	✓
Multiple Type 1/Type 2 devices per root port			✓	✓
Fabric capabilities (256-byte flit)			✓	✓
Back invalidate capabilities on Type 3 devices (HDM-DB)			✓	✓
Fabric Manager API definition for PBR Switch			✓	✓
Host-to-Host communication with Global Integrated Memory (GIM) concept			✓	✓
Trusted-Execution-Environment (TEE) Security Protocol			✓	✓
Memory expander enhancements (up to 32-bit of meta data, RAS capability enhancements)			✓	✓
Security, compliance, and CXL Memory Device enhancements			✓	✓
CXL Bundled Port				✓
Memory RAS enhancements (granular event reporting, Post Package Repair (PPR), and flexible memory sparing operations)				✓

图78 CXL 各版本发布时间及关键规格 [60]

CXL Fabric Manager（FM）

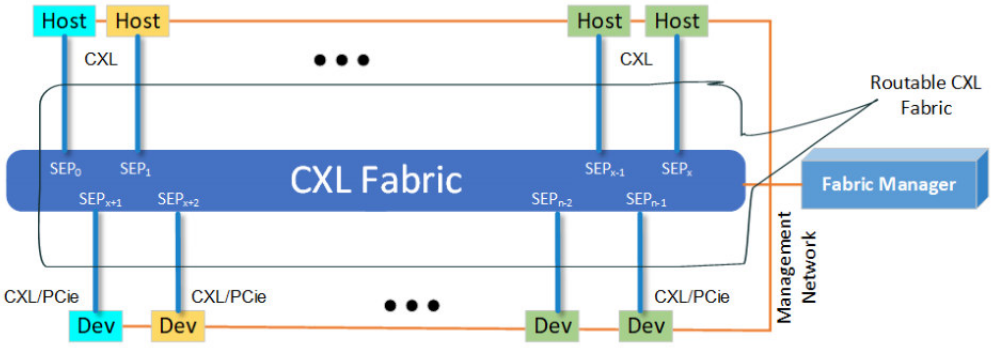


图79 CXL Fabric[61]

当多个 Host 和 Device 通过 CXL Switch 组成 Fabric 时，整个 Fabric 的管理需要一个专属的管理组件 FM（Fabric Manager）进行管理。如上图，Fabric 由一个或多个互联的 CXL Switch 组成。Switch 端口可连接至 CXL 主机（Host）端口或 CXL 设备（Dev）。

Fabric 管理器（FM）在 CXL 网络（Fabric）中不可或缺，但角色可以被其他具备类似管理功能的组件替代。FM 可通过带外对 Switch 网络进行管理。FM 组件也被其他 Scale up 协议借鉴，如 UB 的 UBFM。

FM 的主要功能：

- 负责 fabric 初始化、组件发现，构建拓扑视图，完成链路与参数协商。
- 管控端口绑定、路由配置，协调带宽与内存资源，支撑全局地址映射。
- 监控 fabric 内组件与链路状态，处理异常事件（包括热插拔），保障传输可靠性。
- 提供安全管控与标准化 API，实现多组件协同。

CXL 3.0 与 4.0 技术特点对比

2025 年 11 月发布的基于 PCIe 7.0 的 CXL 4.0 规范，除速率提升一倍以外，在原 3.0 规范基础上继续继续了特性增强，以下对 CXL 3.0 及 4.0 规格进行一些解析：

1. CXL 3.0 技术特点

直至 CXL 3.0 协议发布，CXL 才进入了成熟期。CXL 3.0 于 2022 年 8 月发布，基于 PCIe 6.0 物理层(64 GT/s)，其核心创新包括：

- 多对多连接(MLD)：打破了 CXL 1.x/2.x 的树形拓扑限制，允许多个主机(Host)和设备刚好之间建立点对点或 Fabric 连接，形成网状(Mesh)网络结构。CXL 3.0 支持 Type-2 设备连接到多个主机，实现了跨主机的内存共享。
- 增强型内存一致性：引入 Snoop Filter，减轻主机 CPU 的侦听负担，提升大规模部署下的性能。CXL 3.0 支持更复杂的内存拓扑结构，包括多级内存池化和跨节点的内存共享。这种一致性机制确保了多个主机对共享内存的访问一致性。

- 内存共享与池化：原生支持多个主机共享同一块设备内存(如 CXL 内存扩展器)，并能进行内存池化管理，显著提高内存利用率和灵活性。CXL 3.0 通过 MLD 实现了机架级内存共享，每个 MLD 设备最多可连接 16 个主机。
- 安全增强：支持 IDE 完整性和数据加密框架，为 CXL.io、CXL.cache 和 CXL.mem 流量提供端到端的完整性、重放保护和加密。

CXL 3.0 协议 Snoop Filter（SF）的引入，将为保证多节点 Cache 一致性要进行的 Snoop 探测降低到最低必要限度，极大减少了因 snoop 动作进行的探测报文。以下结合 SF 的机制简要说明一次主机对设备侧的内存访问并保证 Cache 一致性的过程。

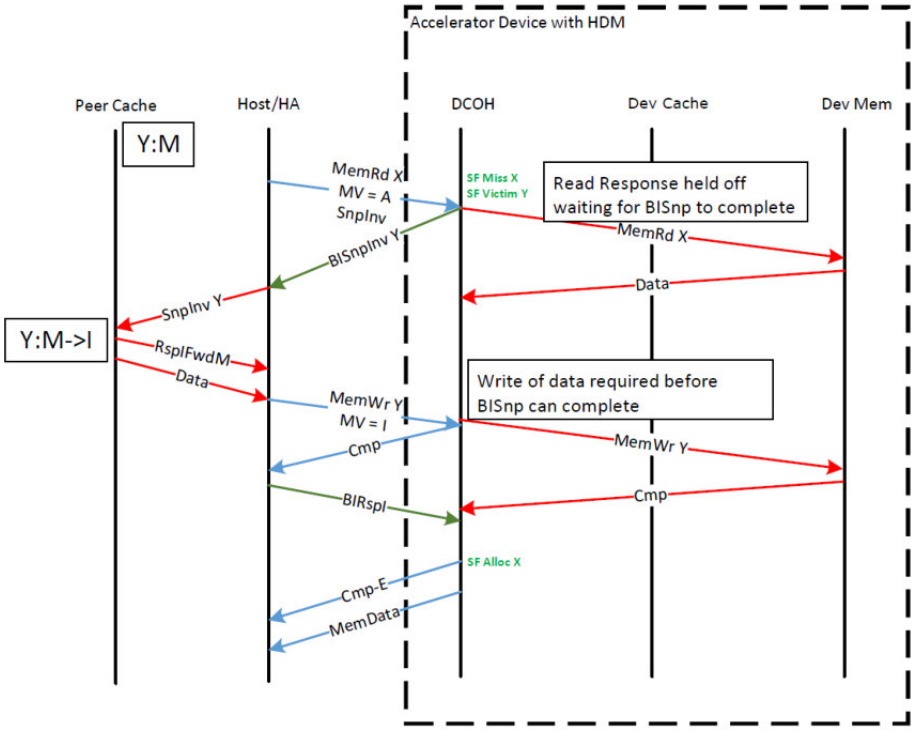


图80 Host 访问 Accelerator（GPU）过程 [62]

DCOH： Device Coherency Operation Handler，设备端一致性操作处理器，执行 Snoop Filter 管理（分配/释放/Victim 流程），处理 BISnp/BIRsp 交互，生成 Cmp/MemData。

Snoop Filter (SF)： 侦听过滤器，位于设备端的轻量级元数据结构，记录哪些主机缓存持有设备内存副本。是 CXL 协议的关键性设计。SF 因容量限制，记录满了以后如果需要记录新的信息，就要清理旧的记录（类似于老化）。要被清理的记录对应的，在 Peer Cache 中被修改的数据将写回至内存，同时标记 Peer Cache 中的数据为 Invalid，再在 SF 中消除该条记录。

HA： Home Agent，主机归属代理，CPU 中负责特定地址范围一致性的硬件模块。

Peer Cache： 对等设备缓存，Fabric 拓扑中其他设备的缓存，此次可为 Host 的 Cache。

HDM： Host-managed Device Memory，主机管理的设备内存，由 OS 统一编址映射至系统物理地址空间

上图流程解读如下：

- 主机发起发起对设备侧地址 X 的 MemRd（memory read），MV = A 表明主机可能持有该缓存行的共享（Shared）、独占（Exclusive）或修改（Modified）副本。DCOH 在一致性确认前，不应将该缓存行副本授予设备。需强制使其他处理单元缓存中 X 的副本失效（Invalid）（SnpInv）。DCOH（设备一致性操作处理器）的 SF（Snoop Filter）Miss（SF 中无 X 地址的记录），且 SF 已满（因 SF 容量有限，无新空间分配新条目）。选择地址 SF 中存在的 Y（地址为 A）为 Victim（SF Victim Y），计划将 SF 中的 Y 清理以腾出空间放置 X。
- 并行启动：
  因 Y 是一致性未确认，对 X 的读取（MemRd for X）在数据（X）返回 DCOH 后暂时挂起。
  DCOH 向主机发起 Snoop，失效 Y 在 Peer Cache 中的副本（BISnpInv Y）。
- 主机响应 BISnpInv Y，向 Peer Cache 发起 SnpInv Y，将 Y 数据标识为脏数据（Invalid），同时 Peer Cache 返回 RspIFwdM，表示被侦听的 Cache line 状态（含数据 Y）已从 Modified 转变为修改态（Invalid），即主机可确认数据 Y 已无如何缓存副本。该 Cache line 的数据也从 Peer Cache 传输至主机缓冲区。
- 主机（Host/HA）将数据 Y 写回至设备侧的内存，设备侧返回完成信息（Cmp）。主机（Host/HA）向设备侧代理（DCOH）发送 BIRspI，表明主机已完成一次 Back-Invalidate Snoop 过程，并且主机内该 Cache Line（含 Y）为 Invalid，主机没有该缓存行的可缓存副本（MV=I）。
- 至此，SF 中已可删除 Y 的信息，空出的空间可以为数据 X 使用，即 X 的信息被写入 SF，表示 X 被其他处理器（Host）Cache。DCOH 向主机（Host/HA）发送 Cmp-E，并发送含 X 数据行。该行将被 Host Cache，且状态为 E（Exclusive），表明该行数据被 Peer Cache 独享。

如前述，Snoop（侦听）是缓存一致性协议中的基本机制，多核 CPU 的 Snoop 广播机制，随着核心数量增加会产生严重的带宽瓶颈和延迟问题，严重影响性能，因此诞生了 SF 机制。

CXL 要解决的是跨系统、多节点的缓存一致性问题，Snoop 广播带来的带宽消耗和延时更加不能容忍，CXL 引入 SF 机制，消除了一致性判断所要进行 Snoop 广播，仅针对包含了设备内存数据的 Peer Cache 进行定向通信，极大优化了 Snoop 机制。但因芯片成本原因导致 SF 容量有限（SF 容量大占用芯片的面积也大），工作过程中 SF 内容需根据访问目的进行不断刷新，如图解，刷新过程也增加了额外的通信流量和延时。CXL 系统将在 SF 容量和性能间选取一个合理的平衡。

2. CXL 4.0 技术特点

CXL 4.0 于 2025 年 11 月发布，基于 PCIe 7.0，其核心创新包括：

- **速率提升：** CXL 4.0 基于 PCIe 7.0，单 Lane 速率可达 128GT/s，以 x2 作为原生宽度，支持最多 4 个重定时器(Retimer)接入，有效扩展了覆盖范围，支持更远距离传输和更多干扰环境。同时也继承了 PCIe 7.0 的光互联能力。
- **通道捆绑技术(Bundled_ports)：** 将多个物理 CXL 设备端口聚合为单一逻辑实体，设备可连接 1 个或多个主机根端口，且不改变现有设备枚举方式，保持向后兼容。即一个设备可以使用多个 x16 链路和同一 Host 互通，不像常规 PCIe 设备只能有一个 x16（or x8、x4、x2）链路连接 Host，设计向目前多链路 Scale up 协议对齐。

3. CXL 技术的应用场景

- **内存池化与资源共享**

内存池化是 CXL 技术的核心应用场景，通过共享和动态分配内存资源，显著提高了数据中心的内存利用率：

内存池化：CXL 支持服务器级内存池化，通过 MLD 和 Type-3 设备，多个主机可以共享同一块设备内存。超算中心可以将内存资源集中管理，根据工作负载需求动态分配，提高了资源利用率和系统弹性。

内存共享：全局 FAM（G-FAM）特性支持跨节点内存共享，多个 GPU 可以使用共同的内存池。这为推理所需的大规模 KV-Cache 内存需求带来了可扩展的解决方案，根据推理过程中 KV-Cache 的需求，可以动态扩展其所需的池化内存，完成以存代算的性能提升。

- **异构计算、AI 加速与大模型训练**

在 AI 领域，CXL 有望成为类似 NVLink C2C 的用于连接 CPU 与 GPU、加速器的关键技术，显著提升访问带宽和降低数据传输延迟，并能提供 Cache 一致性能力。

相对于当前已使用的 NVLink 5.0\ 224GT/s SerDes 传输速率，CXL 3.0\ 64GT/s 的速率存在代差。即便到了 CXL 4.0\128GT/s 的速率仍然和 NVLink 存在差距。得益于 PCIe 广泛的生态，未来基于 PCIe 的 CXL 用于 CPU-GPU，或 GPU-GPU Scale up 互联仍然存在较大的可行性。

- **云原生架构**

CXL 是构建分解式数据中心和云原生架构的核心技术，使计算、内存和存储资源可以独立扩展和按需组合。CXL 支持服务器分解和池化，使原服务器独享的资源（如 CPU、GPU、内存、网卡、存储等）可以通过池化，根据应用需求灵活地进行组合，很好地契合了云原生架构。

PCIe 和 CXL 技术展望

PCIe 和 CXL 代表了互连技术的两个不同维度：PCIe 关注物理传输的极限性能，而 CXL 关注逻辑层面的资源协同与一致性。两者的结合将为未来计算架构提供更强大的互连能力，推动计算性能的不断提升和能效比的优化。

从 PCIe 1.0 到 8.0，带宽持续翻倍，从最初的每 Lane 2.5GT/s 到预计的 256GT/s 以上，同时通过不断改进的编码方式（8b/10b -> 128b/130b，PAM4 -> PAM6?）和信号处理技术（均衡、预加重、FEC 等），显著提高了带宽利用率和信号完整性。其三层架构（事务层、数据链路层、物理层）设计合理，各层职责明确，相互解耦，为现代计算平台提供了高效、可靠、可扩展的互连基础。

CXL 3.0 通过多对多连接和增强型一致性协议，使内存共享和池化成为可能；而 CXL 4.0 则通过通道捆绑、CXL Fabric 和光互连支持，进一步扩展了内存共享的范围和能力，为超大规模计算架构提供了坚实基础。两者均基于 PCIe 物理层构建，但通过协议扩展实现了内存一致性与资源共享功能。

面向未来，PCIe 和 CXL 将继续协同演进，PCIe 将不断突破物理传输的极限，如更高的速率（PCIe 8.0 的 256 GT/s）、更远的传输距离（百米级光互联）和更低的延迟；而 CXL 将不断扩展资源协同的边界，如更复杂的内存拓扑（CXL Fabric）、更强大的一致性协议（G-FAM）和更广泛的安全保护机制。

对于 AI、HPC 等高性能计算领域，CXL 或将成为连接 CPU、GPU 和内存扩展器的关键技术，支持超大规模模型的训练和推理；而对于传统数据中心和企业应用，CXL 将提供足够的内存共享和池化能力，优化资源利用率。两者的协同应用将使计算架构更加灵活、高效和可扩展，为数字经济时代的创新提供坚实基础。

3.2 GLink 协议介绍

GLink 技术要点

GLink 为新华三公司为满足超节点互联要求，自行研制的 Scale up 互联方案。本章节旨在介绍 GLink 技术方案，该技术方案专为高性能计算、人工智能训练和推理、大数据处理等场景，加速卡(如 GPU)超节点架构中跨芯片高带宽、低时延、高可靠的通信需求而设计，也适用于对数据传输效率和系统扩展性有严格要求的其他场景。

技术目标

GLink 旨在提供智能、灵活、高效、可靠的加速卡卡间互联技术，协同多个加速卡的算力以及域内互联的资源池，使集群像一个超级加速卡一样工作。通过互联实现加速卡智算集群性能的纵向扩展，为大模型训练、推理以及高性能计算等 AI 应用场景提供互联技术支撑。基于此，GLink 互联技术的核心目标：

1. 支撑多种互联拓扑

支持 Scale Up 域内多种加速卡互联方式：支持直连拓扑，为加速卡间搭建直接的、低延迟的通信路径，不需要交换芯片参与，能降低系统整体成本与功耗，简化物理布线与系统管理流程；支持基于交换芯片互联的单级拓扑，在交换拓扑架构下，支持源加速卡与 Scale Up 域内任意目的端加速卡的全向通信，无需经过主机 CPU 或者其他加速卡绕转；支持机柜间的背靠背互联拓扑，通过此背靠背互联，系统可将 2 个机柜聚合为一个统一的 Scale Up 高带宽域；支持两级交换的互联拓扑，通过两级交换的互联拓扑突破单机柜或双机柜直连架构的规模瓶颈，构建能支持更多加速卡计算节点的全互联高带宽域。有关互联拓扑的具体示例，请参见互联拓扑章节。

2. 大规模互联扩展

基于 Scale Up 互联规模必然会继续扩大的需求，GLink 互联技术支持通过二级 Switch 互联架构实现网络规模扩展，最大可支持 2K 规模的加速卡 Scale Up 互联网络，满足超大规模计算集群的互联需求。同时基于二级 Switch 互联设计，GLink 互联技术可以支持从 8 卡互联规模平滑扩展至千卡级互联，适配不同阶段的算力需求增长。

GLink 互联技术在二层网络架构上实现技术创新，通过数据细分确保数据包在 HASH 计算后的负载均衡，避免因 HASH 极化导致的链路利用率不均问题；采用**端到端 Credit Base 流控和点到点 Credit Base 流控**，双维度 Credit 流控机制确保数据传输无丢包；以及通过链路重传、端到端重传、链路故障自动隔离等技术提升网络健壮性。

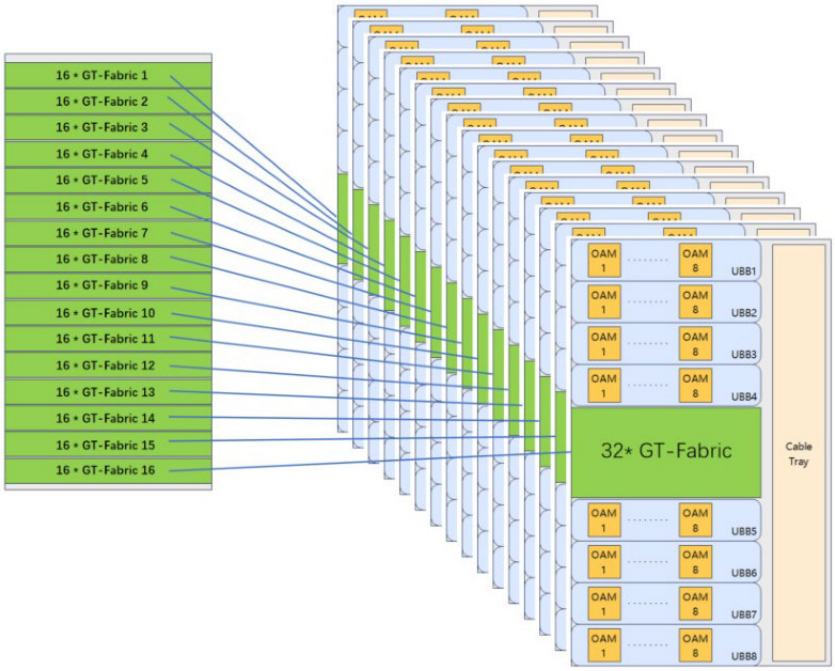


图81 1024 卡互联示意图

3. 多语义支持

为实现加速卡间的协同计算和互联通信的高效，在通信语义方面，GLink 互联技术支持**内存语义和通道语义**，以应对加速卡在不同应用场景下多样化的内存交互需求，其目标是在统一的互联架构上，为不同粒度的内存操作提供最优化的执行路径。

内存语义场景：GLink 互联技术与系统加速卡通过全局虚拟地址空间与低延时传输协议的协同，实现了细粒度数据访问的高效、可靠，提升并行算法的执行效率，降低开发者的编程复杂度。

通道语义场景：GLink 互联技术为实现批量数据的高效移动提供一条硬件路径，将一块大规模、连续的内存区域作为一个整体，从一个源高效地拷贝至一个目的，通过“传输-计算”重叠，系统可最大化利用硬件资源，提升整体效率。

4. 无阻塞调度数据传输

为支持加速卡间高效互联与极致通信带宽利用率需求，GLink 互联技术方案支持细粒度的逐链路 Hash 负载均衡，以及端到端基于 Credit-Based 的流控机制，通过“负载均衡+动态流量控制”的协同作用，GLink 互联技术实现了对网络资源的主动式、无拥塞管理，提升了互联网络的传输效率，大模型训练构筑了一个高效、可靠的数据传输平面。

逐链路 Hash 负载均衡：GLink 互联技术通过逐链路 Hash 算法，将数据流动态解构为细粒度数据单元，均匀地喷洒到所有可用物理链路上，确保了每一条链路带宽都能被充分利用，消除了因流量不均而产生的网络热点，同时 GT-Link 和 GT-Fabric 进行深度协同，严格保障数据接收顺序与发送顺序一致，确保加速卡接收保序。

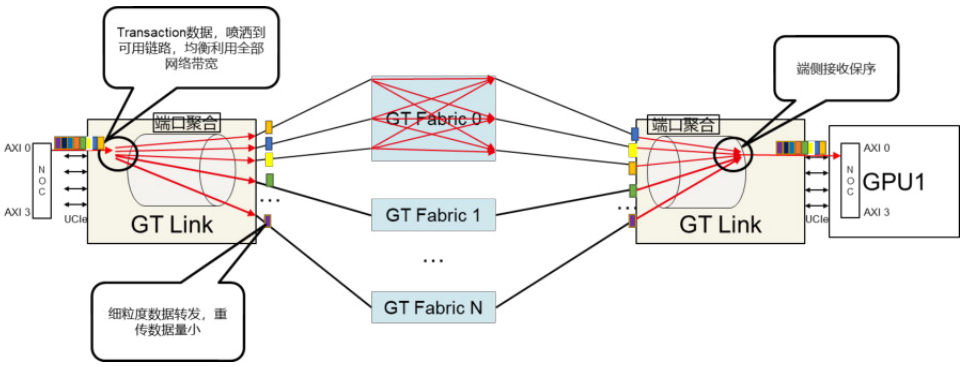


图82 逐链路 Hash 负载均衡

端到端 Credit-Based 流控：针对传统网络中因拥塞导致的丢包与延迟抖动问题，GLink 互联技术通过将网络资源管理从“事后补救”变为“事前规划”，通过源端发送数据请求和目的端根据当前处理能力，向源端授予 Credit 方式，进行带宽分配，从数据源头精确管控数据注入速率，确保网络传输不会出现因缓存溢出而导致的延时增大或者丢包问题。

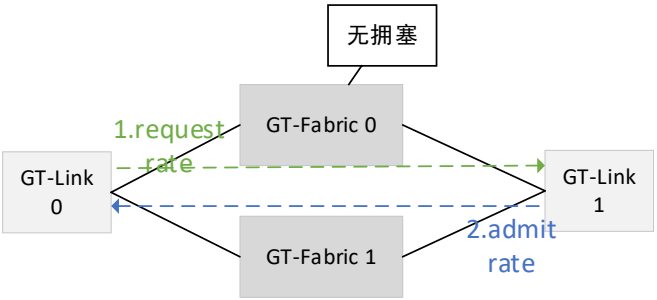


图83 端到端 Credit-Based 流控

5. 可靠数据传输

要充分发挥超节点算力，可靠性、稳定性是决定系统计算效率及成本的重要指标之一，为应对这一挑战，GLink 互联技术设计了分层可靠协同机制。

GLink 互联技术物理层支持 FEC(RS272/RS544)，作为保障数据完整性的第一道防线，在接收端自动检测并纠正传输过程中发生的特定数量的比特错误。当前出现无法纠错的数据时，会在接收端产生错误报文，GLink 互联技术支持链路层点到点重传，保证链路误码等错误报文，通过链路重传快速恢复。

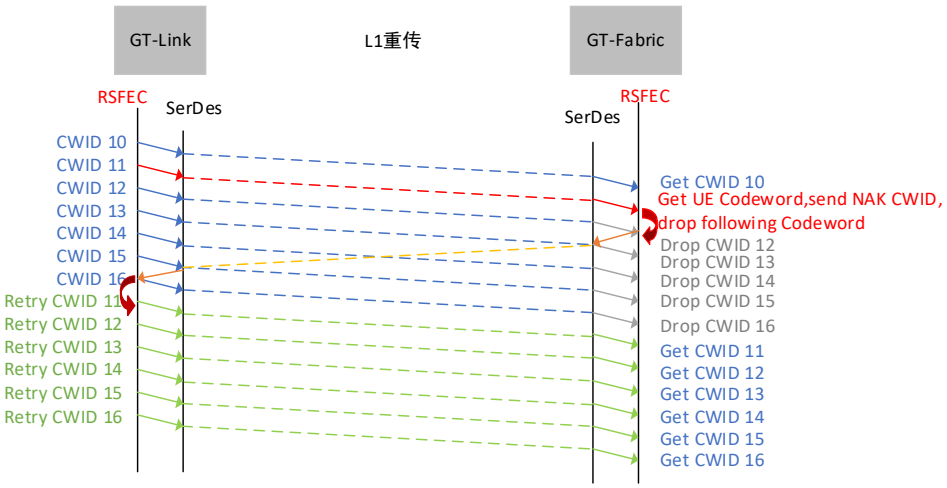


图84 链路重传

GLink 互联技术，针对硬件链路故障场景，设计了完整的处理机制，确保在单条或多条物理链路发生故障时，系统能够自动、快速地实现路径切换。

链路层发现链路异常后，会立即进行自动隔离，同时网络层会自动整网同步故障信息，后续的数据转发会被硬件自动、无缝地引导至其他可用链路上。

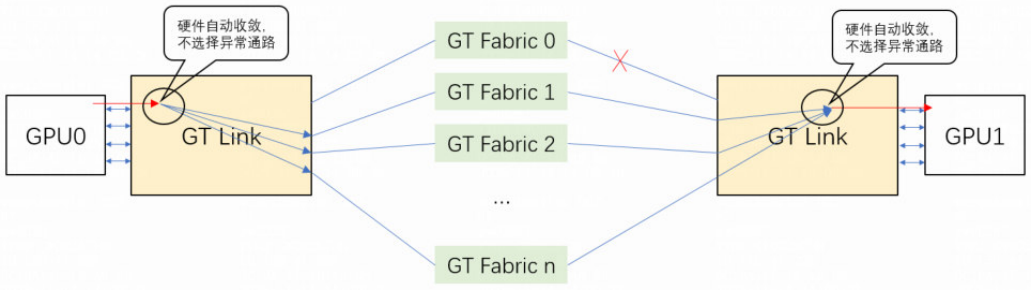


图85 链路故障隔离

在传输层，GLink 互联技术支持**端到端重传机制**，作为其分层可靠性架构中的最终保障层。该机制与下层的链路级故障隔离、链路重传等功能构成了多层级可靠性协同体系：链路层重传高效解决绝大多数链路瞬态错误，而端到端重传则专注处理极罕见的异常场景（如硬件链路故障）。当发生此类故障时，系统首先通过链路自动隔离机制实现快速切换，再通过端到端重传完成数据无损恢复，从而确保即使在最极端异常情况下，网络仍能保障 100%的数据可靠交付，为关键业务提供终极可靠性保障。

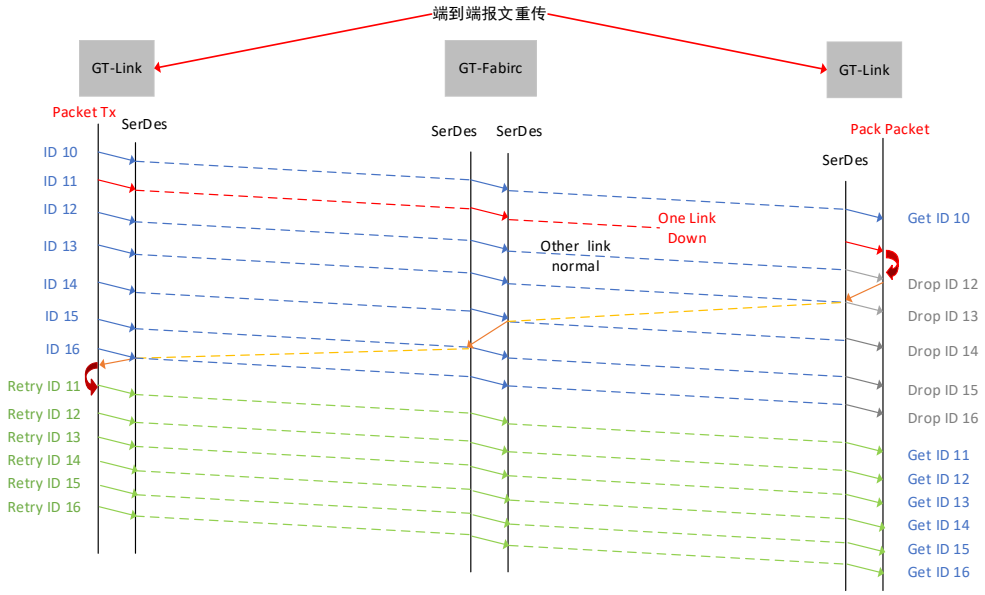


图86 端到端重传

通过从物理层到事务层的多个层级上部署系统性的容错与故障处理策略，GLink 构建了一个具备高度韧性的通信架构，将可靠性从单一的链路属性提升为整个系统的架构级能力。它确保了即使在组件不可避免发生故障的情况下，超节点系统依然能够维持稳定运行。

图87 多层级可靠性保障

6. 集合通信在网计算加速

为应对大规模分布式 AI 应用中的通信瓶颈，GLink 引入了创新的集合通信加速能力。该技术通过将部分集合通信计算任务（如 Reduce、All-Reduce、Broadcast）从加速卡节点**卸载至网络交换芯片**，实现通信范式的根本性变革。

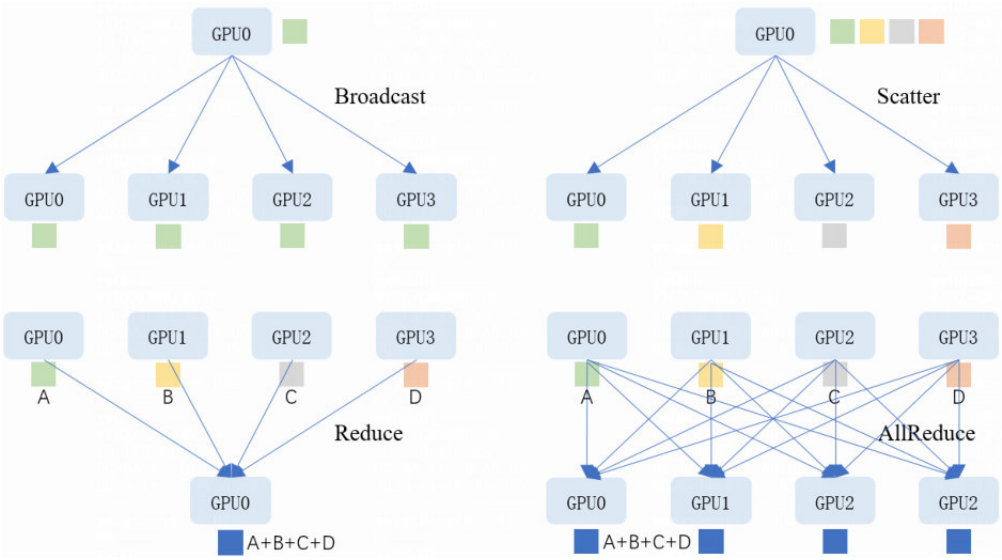


图88 集合通信

集合通信在网计算核心价值：

- 减少网络数据总量,通过在网络内部直接对数据执行聚合运算，显著降低需要在链路中传输的数据量，从而提升整体带宽利用效率；
- 降低端点开销,减轻加速卡在集合通信中的计算与内存带宽负担，使其更专注于模型任务。

这一机制从根本上重塑了网络的角色：交换芯片从一个被动的数据管道，转变为一个主动参与计算的协同处理器。这不仅有效降低了通信延迟、提升了有效带宽，更将网络从基础设施的“成本中心”转变为赋能系统算力提升的“价值中心”，为千卡级集群的极致扩展提供了关键支撑。

互联拓扑

1. 直连互联拓扑

为满足小规模 AI 集群对成本与效率的极致追求，GLink 互联技术提供灵活的直接互联拓扑。这类拓扑的核心目标是为加速卡间搭建直接的、低延迟的通信路径，其核心优势：

- 无需昂贵的交换芯片，显著降低系统硬件成本与整体功耗；
- 点对点直连架构消除了交换跳数，实现节点间最短通信延迟；
- 物理布线更为简洁，系统管理复杂度显著降低，便于管理与维护。

受限于加速卡芯片封装面积与 I/O 引脚数量等物理条件，直接互联所能支持的加速卡数量存在明确上限（例如典型场景下最多支持 **8 颗加速卡**互联）。

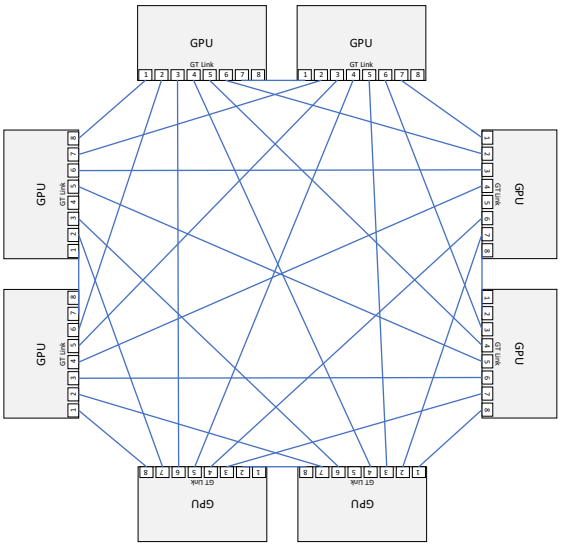


图89 加速卡直连拓扑

2. 单级互联拓扑

GLink 互联技术支持通过**单级互联拓扑**，构建大规模且高效的 Scale Up 集群拓扑网络。该拓扑采用多交换芯片协同架构，在逻辑上构建统一的交换层，实现加速卡节点间高效、灵活的数据转发与互联。

其核心设计通过模块化交换单元的组合，实现系统规模的线性扩展，具备优异的扩展能力。在通信过程中，任意两个加速卡节点间建立连接后，数据可在预设的高带宽路径中高效传输，从而最大限度利用加速卡端口的完整 I/O 带宽。由于数据包**仅需经过单次交换**转发即可到达目的节点，端到端通信延迟可控制在微秒级。结合硬件级的流量调度机制，系统能够为每个加速卡通信分配专属带宽资源，确保高效的通信效率，满足高并发计算场景的需求。

单级互联拓扑通过交换芯片直连架构，实现了带宽最大化、延迟优化和线性扩展性的三重突破，为构建下一代高性能计算网络提供了坚实的技术基石，适用于需要超大规模加速卡互联的 AI 训练、科学计算和实时数据分析等前沿领域。

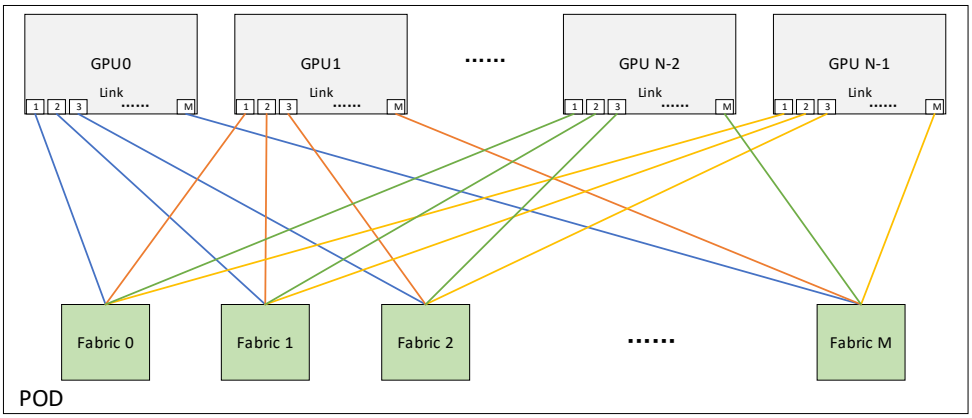


图90 多加速卡单级互联拓扑

3. 背靠背互联拓扑

为突破单机柜在供电、承重与空间上的物理限制，同时适配现有数据中心环境，GLink 互联技术提供支持背靠背互联拓扑。该架构旨在将 Scale Up 的高性能域无缝扩展至多个机柜，构建逻辑统一的跨机柜紧耦合高带宽计算域。

背靠背互联拓扑架构关键优势：

- 该架构充分考虑了数据中心的现实约束，无需对单个机柜提出超额的供电与承重设计。通过将负载分散到两个标准机柜中，显著提升了在常规机房部署大规模集群的可行性，降低了基础设施改造成本。
- 两个物理上独立的机柜，通过这一架构被融合为一个逻辑上统一的 Scale Up 系统。所有加速卡，无论位于哪个机柜，都处于同一个高性能互联网络域内，实现了 Scale Up 规模的翻倍。

GLink 的背靠背互联拓扑，在物理限制与算力需求之间取得了平衡。将 Scale Up 的优势成功延伸至多机柜环境，为在标准数据中心内部署高效能、紧耦合的 AI 计算集群提供了关键且可行的技术路径。

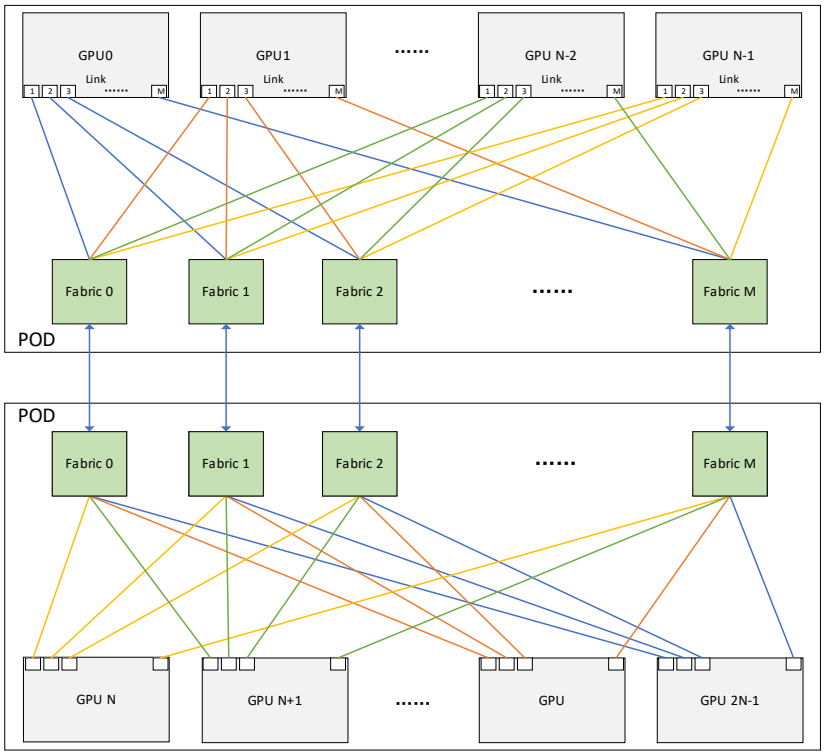


图91 多加速卡背靠背互联拓扑

4. 两级互联拓扑

为突破单颗交换芯片在端口密度与转发带宽上的物理极限，满足千卡级加速卡计算集群对全互联高带宽域的迫切需求，GLink 引入支持两级互联拓扑。此架构标志着从机柜级 Scale Up 向数据中心级 Scale Up 的关键演进，是满足未来更大规模 AI 模型及复杂并行计算任务需求的核心网络基石。

在此架构下，通信路径根据加速卡的相对位置而确定：Pod 内通信，数据流仅在单个接入交换机内部完成转发，实现一跳的超低延迟通信；Pod 间通信，数据流需要通过三个交换芯片转发完成。尽管 Pod 间通信引入

了额外的跳数，但 GLink 通过其高带宽互联技术与低延迟交换芯片，确保其在规模化扩展中仍能保持极高的效率。

GLink 支持的细粒度逐链路负载均衡机制，以及端到端基于 Credit-Based 的流控机制，实现了两级互联网络链路带宽均衡利用，消除了流量热点，是保障超大规模网络无阻塞运行的关键。

GLink 的两级互联拓扑，通过其清晰的分层架构与智能的流量管理机制，将 Scale Up 的边界从单个机柜扩展到数据中心级别。它代表了高性能网络技术在 AI 驱动下的前沿演进，为构建超大规模 AI 集群提供了坚实且面向未来的互联解决方案。

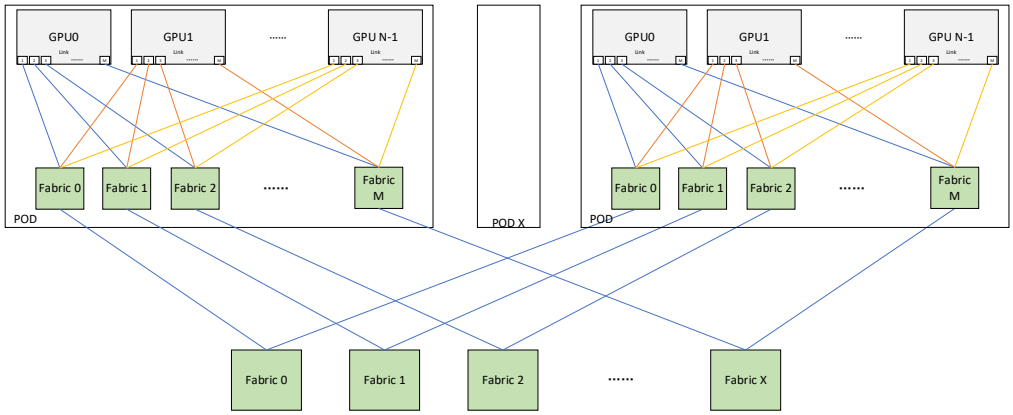


图92 多加速卡两级互联拓扑

GLink 互联技术

互联技术方案

GLink 互联技术通过创新的“端侧+网侧”协同设计，为下一代 AI 基础设施构建高带宽、低延迟、无阻塞且具备大规模扩展能力的互联底座。

GTLink 为 GLink 互联技术加速卡端侧高速互联接口，负责芯片级的高速数据收发与协议处理。

GTFabric 为 GLink 互联技术网侧高性能交换芯片，支持拥塞控制、自适应路由、负载均衡、在网计算等功能。

GLink 通过端网协同为加速卡提供智能、灵活、高效、可靠的卡间互联技术解决方案，高效协同多个加速卡的算力以及域内互联的资源池，赋能合作伙伴快速打造具有竞争力的智算产品。

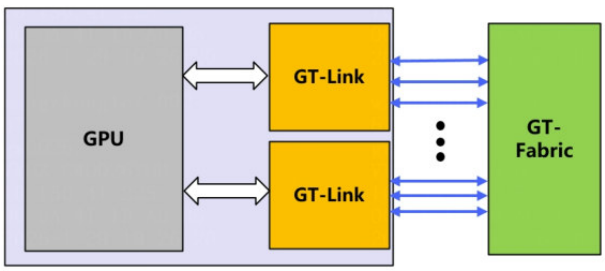


图93 GLink 互联技术

GLink 技术栈

GLink 技术栈遵循分层化设计准则，构建了一个由物理层、链路层、网络层、传输层和事务层组成的完整通信体系。这种清晰的分层架构不仅定义了严格的功能边界，更通过各层级的高效协同，为整个互联系统提供了坚实的结构性基础。

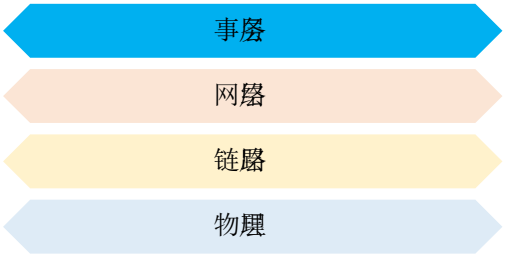


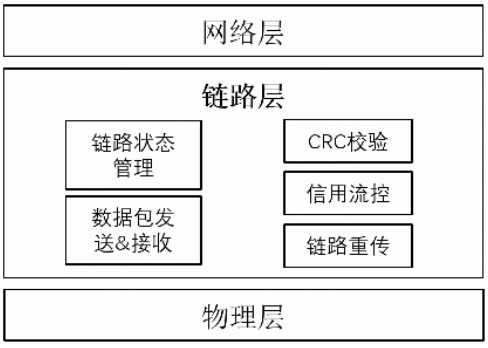
图94 GLink 技术栈

1. 物理层

物理层是 GLink 互联技术栈的最底层，承担着数字逻辑与模拟信号之间转换的职能。它通过在物理介质上，高效、可靠地传输与恢复原始比特流，向上为链路层提供数据收发服务，向下通过高速 SerDes 串行通道与对端设备建立物理连接。支持 56G、112G 及 224G 等高速 SerDes 串行通道，为持续提升单端口及系统聚合带宽提供底层支撑。集成 FEC（前向纠错）编解码，支持 RS(544, 514) 与 RS(272, 257) 等，能够自动检测并纠正传输过程中产生的比特错误，提升链路稳定性。

2. 链路层

链路层位于物理层与网络层之间，为网络层提供点到点的可靠通信。链路层主要功能如下：



接口链路状态管理：监测与维护其所属端口的连接状态，将链路状态变化上报，供上层设备驱动软件或网络管理软件处理。迅速感知链路状态异常事件（如 Down），为路由收敛、故障切换提供决策依据。

数据包发送与接收：数据发送方向，为数据包添加链路层头和 CRC 后，交付物理层发送；数据接收方向，从物理层接收的数据，在完成 CRC 校验后，剥离链路层头和 CRC，将数据包传递下一级处理。

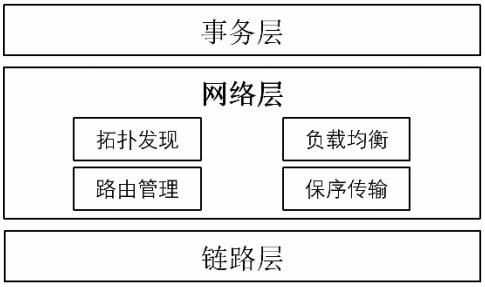
CRC 校验：发送端计算添加 CRC 校验码，接收端重新计算校验码并比对。CRC 校验失败后触发链路重传机制，确保错误数据不会被向上传递。

信用流控：根据接收端缓存能力生成“信用”，发送端必须持有信用，才能发送数据，从而避免接收端缓存过载而导致的丢包。

链路重传：发送端在本地缓存已发送但未被确认的数据副本，当数据出现 CRC 错误或者丢失，链路层会自动启动重传。

3. 网络层

网络层位于链路层与传输层之间，它建立在链路层提供的可靠单跳传输能力之上，是互联架构的“指挥中心”，负责将数据从源节点，按照规划路径准确、高效地送达目的节点。



拓扑发现：网络启动或设备状态发送变化（如新增节点、链路故障）时，系统会自动探测并学习设备间互联的物理结构(即哪个端口连接了哪个设备)，实时绘制出一张“网络地图”。

路由管理：不同设备间同步互联拓扑关系后，会为探测到的目的节点生成转发路由表，并自动下发到芯片硬件中。

基于硬件自动的拓扑发现和路由生成功能，可以支持新加速卡节点的自动识别和无缝加入，极大简化了运维。当发生故障时，能毫秒级感知并重新计算路径，实现业务的快速自愈，保障任务不中断。

负载均衡：通过将数据流动态拆分为更小的数据单元，并对每个单元进行独立哈希，将其动态地“喷洒”到所有可用的物理链路上。确保每条链路带宽都能被充分利用，消除因流量不均而产生的网络热点。

保序传输：GTLink 和 GT Fabric 深度协同，基于端口+通道，严格按照原始顺序向加速卡传输数据。

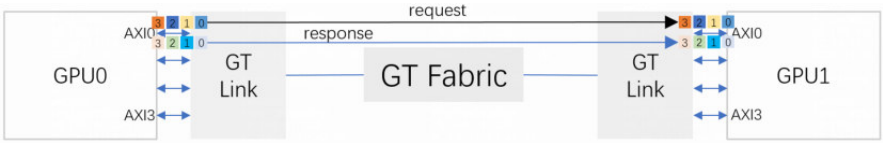


图95 保序传输示例

4. 事务层

事务层是 GLink 互联技术栈中直接与加速卡计算生态交互的最高层次，它承担着 NOC 总线协议与网络事务报文之间双向转换的核心职责。并管理网络事务报文的端到端可靠交付。

为适应不同加速卡内部互联架构的差异化需求，端侧 GTLink 可以支持不同 NOC 总线的事务转换，内存语义场景，以 AXI 总线为例：

加速卡发送方向：AXI 总线输入事务(如内存访问、原子操作等)->打包成网络报文->由 Tx Path 数据通路发送出去。

加速卡接收方向：Rx Path 输入数据->从 Header 中解出通道信息->恢复成事务输出到 AXI 总线。

即事务层在发送端将加速卡发起的内存访问、原子操作等转换为网络事务报文；在接收端则执行逆向转换，将网络报文还原为加速卡可执行的指令格式，从而实现跨节点内存访问操作。

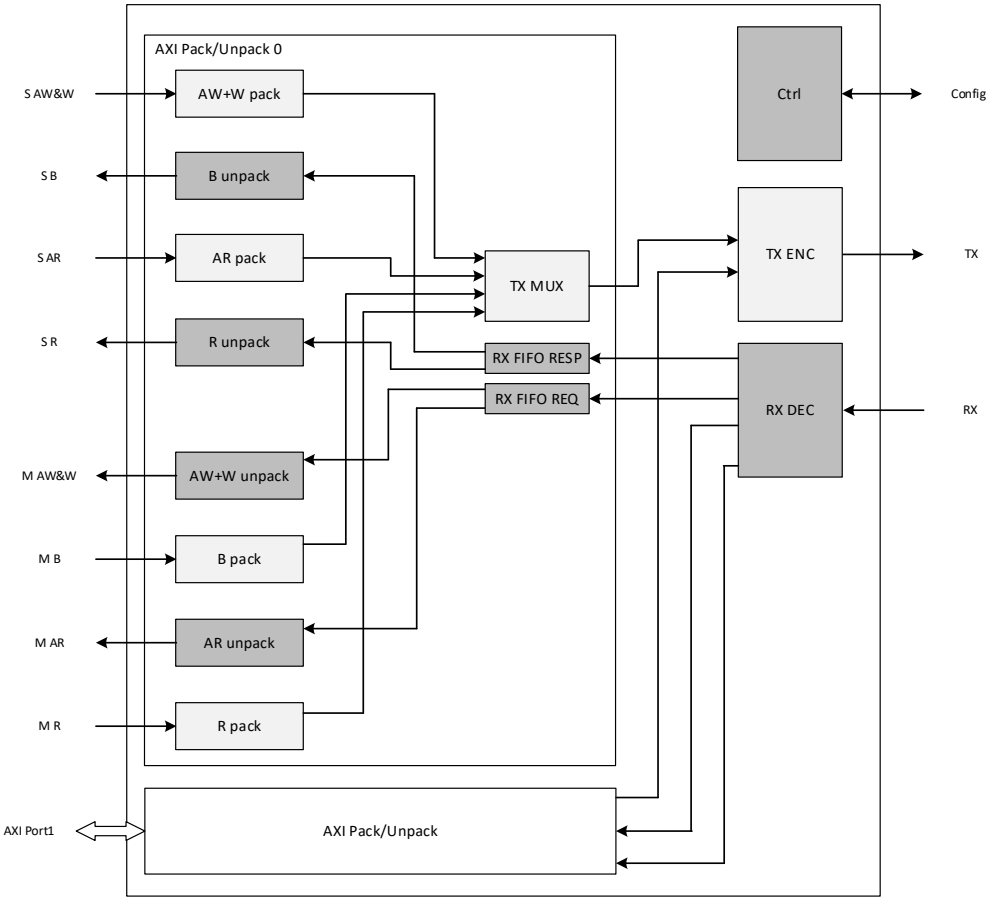


图96 AXI 和数据报文转换示例

通道语义场景，以 AXI-Stream 为例：

加速卡使用专用的拷贝引擎，通过 AXI-Stream 通道，将数据从一个加速卡的显存直接、大批量地拷贝至另一个加速卡的显存；避免对数据块进行不必要的切分和重组，可以使用大包直接传输以提升网络有效载荷率。

加速卡总线事务转换成网络报文格式示例如下，其中 GTLink Header 为精简网络头，包含目的加速卡 ID 和通道信息，用于网络转发使用，Ext Header 为总线事务信息，目的端根据此信息将数据报文恢复成事务输出到总线。

GT Link Header	Ext Header	Payload

除了协议转换，事务层还提供端到端控制功能，这些功能在网络层和链路层之上，提供了最终的可靠性保障。

端到端 Credit-Based 流控：加速卡发送端 Link 根据队列状态向目的端申请 Credit，目的端根据接收能力向源端分发 Credit，通过该机制控制源端发送的数据量在 GT Fabric 和目的端加速卡处理的能力范围内，避免 GT Fabric 和目的端加速卡出现拥塞情况。

端到端重传：尽管链路层重传解决了大多数单跳瞬时错误，但系统仍可能面临诸如硬件链路永久性故障、交换机故障等罕见异常场景。针对此类故障场景，GTLink 和 GT Fabric 组成的系统可以进行路由自动隔离，切换使用正常链路，故障窗口期间的数据丢失，可以通过端到端重传完成数据无损恢复。

D2D 互联

当前，为应对超越单卡算力极限的挑战，国内外加速卡厂家都在通过类似 NVLink+NVSwitch 方案实现卡间互联，然而该领域正处于起步、多种技术方案并存的阶段，技术路线演进存在不确定性。这种局面为整个行业带来了共同的挑战与新的机遇。

将互联协议 IP 直接集成到加速卡芯片中，是初期的自然选择，但随之暴露出诸多劣势：

- 当前有多种不同的互联技术路线，加速卡厂商若想保持生态开放性，就需要内置多种不同的协议，极大地增加了芯片设计的复杂度；
- 随着网络带宽的增加，SerDes 所需要的面积、各种互联协议数字逻辑所需要面积会越来越多，互联 IP 挤占宝贵的芯片面积，从而直接牺牲加速卡的算力性能；

采用封装技术将加速卡计算芯粒与专用通信芯粒进行合封，已成为业界公认的突破性路径。芯粒封装集成的模式，使用户可以自主选择 IO 芯粒的数量和类型，针对不同的市场区间做出不同的权衡。

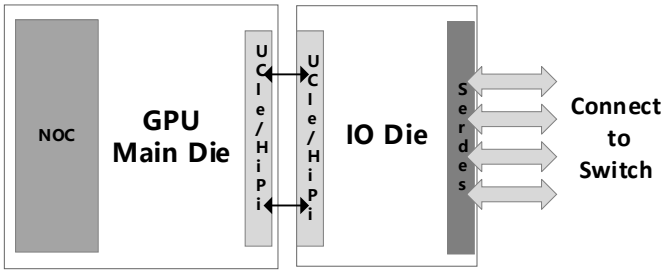


图97 芯粒互联示例

GTLink 支持通过 UCIe 标准接口，与加速卡计算芯粒在同一个封装内进行紧耦合互连。

加速卡计算芯粒：专注于执行矩阵运算、CUDA 核心等核心计算任务，无需集成复杂的高速 SerDes 和网络协议栈，从而最大化其面积用于提升算力密度和能效。

GTLink 通信芯粒：作为专业的“网络接口芯粒”，其内部集成了 GTLink 协议栈的全部功能，包括物理层（PHY）、控制器及与外部 GT-Fabric 交换网络对接的高速接口。

通过 UCIe 互联方案，核心价值与优势：

开放互联生态：UCIe 作为由行业巨头共同推动的开放标准，为不同厂商的“计算芯粒”与“通信芯粒”提供了互操作的基础。GTLink 通信芯粒是一个标准化的“网络接入模块”，赋能更广泛的 AI 加速器。

解耦技术路线：如前所述，加速卡厂商可通过更换不同的通信芯粒，灵活支持 GTLink 或其他互联协议，从根本上消除了技术路线选择风险。

极致性能：UCIe 提供了远超传统板级 PCIe 的超高带宽和超低延迟的裸片间互连。这使得加速卡计算核心与 GTLink 通信单元之间的数据交换几乎无瓶颈，确保了外部互联网络的极致性能能够无损地传递至计算核心。

工艺与成本优化：计算芯粒可采用最先进的制程节点追求极致算力，而通信芯粒可采用更适合模拟/混合信号设计的成熟或专用工艺。在优化整体性能的同时，有效控制制造成本。

快速迭代：通信芯粒可独立于加速卡进行快速设计和流片，以适配新一代 GT-Fabric 网络或新的传输协议，使得加速卡产品线能快速跟进网络技术的演进。

总结

GLink 是一套完整的高性能、可扩展互联技术体系，专为满足下一代 AI 算力集群中千卡乃至万卡级加速卡的高效协同而设计。它通过创新的技术架构与灵活的部署模式，旨在消除大规模分布式训练中的通信瓶颈，释放集群的整体算力潜力。

GTLink 提供了两种关键的实现路径，以适应不同的产品战略与供应链需求：

IP 集成模式：将 GTLink 协议 IP 内核集成到加速卡主芯片中。此路径适合追求高度集成化、对功耗和成本有极致要求的场景，能够提供最优的片上互联效率。

IO Die/Chiplet 芯粒互联模式：GTLink 可作为独立的通信芯粒，通过 UCIe 等先进互连标准与加速卡计算芯粒在封装内集成。此模式具有显著优势：

解耦与敏捷：使加速卡计算芯粒专注于算力提升，通信芯粒可独立快速迭代，灵活适配多种网络协议与拓扑。

工艺优化：计算芯粒与 IO 芯粒可采用最适合的半导体工艺，实现性能、成本与能效的最佳平衡。

降低行业风险：为加速卡厂商提供了互联技术路线的灵活性，规避了技术锁定风险，促进了产业生态的开放与协作。

总结而言，GLink 互联方案提供 AI 计算基础设施的端网协同解决方案。GTLink 与 GTFabric 交换系统协同，共同构成了从芯片到集群的完整高性能网络平面。GLink 通过其全栈技术能力、极致的性能与可靠性设计，以及前瞻性的芯粒化部署策略，为业界提供了一个强大、开放且可持续演进的大规模互联选择。它致力于将分散的加速卡算力资源整合为一台高效的“虚拟超级计算机”，从而加速万亿参数大模型的创新与落地，赋能 AI 产业的持续跨越式发展。

3.3 ETH-X 和 SUE/ESUN 协议介绍

背景技术介绍

当前针对以太网的技术路线主要有 ODCC 的 ETH-X 标准和 OCP 建立的 ESUN(SUE)标准，两者相关区别和联系可参考如下：

维度	ETH-X	SUE/ESUN
生态与主导方	ODCC社区，信通院联合腾讯等多家国内厂商联合制定；据称可支持开放的UALink和OISA等scale-up的协议。	OCP基于博通的SUE是成立ESUN，旨在建立统一的以太网交换和帧技术协议框架，SUE-T是ESUN兼容的协议之一。

维度	ETH-X	SUE/ESUN
特点	通过PAXI实现极致低延迟 全局释放一致性内存模型 支持内存语义	通过头部压缩和多实例部署提升带宽 共享内存机制 支持单边内存语义+跨 XPU 直接内存访问
寻址能力	二次译址：GPU ID->编码AXI User ID->映射到目的 GPU 的MAC 地址	由 XPU 在 SUE 实例之外完成
XPU指令层接口	基于AXI 4和APB3优化的PAXI 数据和控制通过DAXI、CAXI区分	信号线接口和AXI 4 接口两者可选 数据和控制通过接口不同字段区分
底层技术	事务层流控：基于次数Credit流控机制 链路层流控: 增强CBFC + 基础PFC FEC：KP4 FEC（IEEE RS-544）	链路层流控：UEC-CBFC + PFC FEC：IEEE RS-272
适配规模	256/512卡互联（一次交换转发）	最大1024卡互联（单跳）
协议实用测试	样机点亮，出具测试报告	理论推测阶段

通过上述对比可以发现，这两种协议的核心思想，均建立在两大成熟的工业标准之上：AXI 和以太网（Ethernet）。AXI（Advanced eXtensible Interface）作为芯片内高性能、高可靠性的片上总线，定义了模块间数据与控制交互的“普通话”；而以太网（Ethernet）为芯片间无处不在的网络基础，其标准数据帧格式构成了物理链路上信息传递的“通用载体”。理解 AXI 的事务模型与以太网帧的结构，是洞悉 ETH-X 与 SUE 如何将片内总线语义“翻译”为可在片间网络高效、可靠传输的数据包的基石。

AMBA 的 AXI4 和 APB3[63]

APB 和 AXI 作为 ARM 高级微控制总线（AMBA）协议家族的关键成员，是片上系统（SoC）设计中连接和管理功能模块的核心标准。AMBA，Arm Advanced Microcontroller Bus Architecture，作为一个开放的片上互连规范，定义了各模块间通信的规则，确保系统高效运行。APB 协议专门针对低带宽的控制访问场景设计，重点优化外围设备的低功耗要求，如系统外设的寄存器接口。而 AXI 协议则通过集成多种特性，支持高速亚微米级芯片内部的互联需求，广泛应用于连接 CPU、GPU 以及高速外设（如 DDR 内存和 PCIe 控制器），是高性能系统互联的基础。

APB3（Advanced Peripheral Bus 3）于 2003 年随 AMBA 3.0 发布，专为访问外设的可编程控制寄存器而设计，其特点如下：

- APB 接口是一种低成本接口不采用流水线架构，是一种简洁的同步协议；
- 每次数据传输至少需要两个时钟周期才能完成；
- APB 外设通常通过 APB 桥接器连接至主存储系统；
- APB 传输由 APB 桥接器发起。APB 桥接器被称为请求方（Requester）；外设接口负责响应请求，被称为完成方（Completer）。

AXI4(Advanced eXtensible Interface 4) 作为 AMBA4.0 规范的一部分，于 2011 年正式发布，AXI 互联接口采用标准化信号进行规范，显著简化了不同 IP 核之间的集成与互联，从而提高了设计的互操作性和灵活性。

- **读写通道分离：**独立的读写通道，可并行传输，提升接口带宽，避免读写操作阻塞；
- **异步事务执行：**主设备可连续发起事务，无需等待前序事务结束；地址发送与数据传输没有严格的执行时序要求，进一步提升主从设备交互灵活性；
- **支持事务乱序：**基于事务 ID 区分不同事务流，支持乱序完成，单个物理端口实现多逻辑端口功能，优化系统资源利用率。

相比 AXI3，AXI4 增加了 QoS 信号、多 region 信号和用户自定义信号三种信号类型，最大突发长度（Burst Length）也从 16 增大到 256，使得 AXI4 更加高效、灵活和可靠。

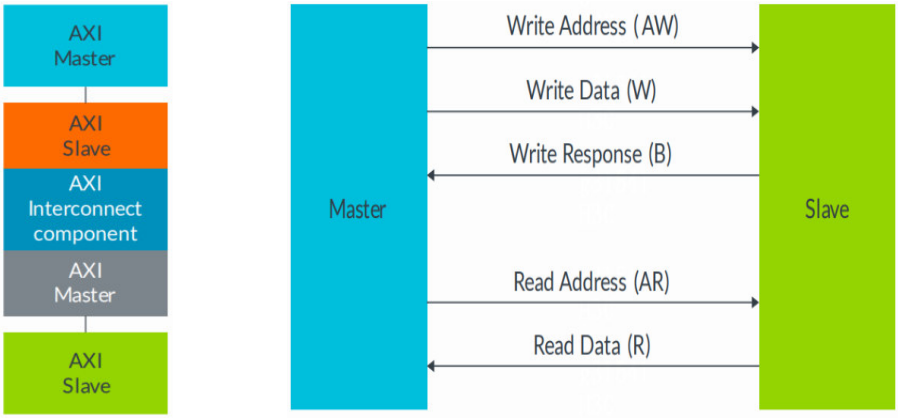


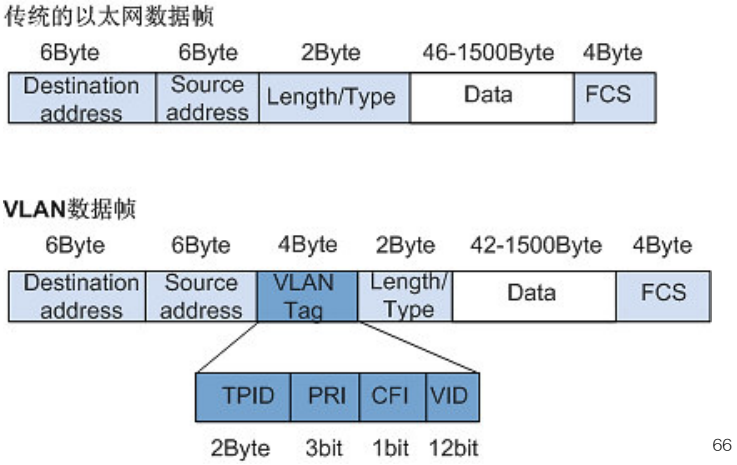
图98 AXI 互联组件接口的实现方法（左）和 AXI 的五种独立通道（右）[64]

以太网的数据帧[65]

以太网数据帧的结构包含几个关键字段：目的 MAC 地址（DMAC）、源 MAC 地址（SMAC）、类型、数据以及帧校验序列（FCS），这些都由 IEEE-802.3 标准详细定义。在此标准中，数据链路层被细分为两个子层：媒体接入控制子层（MAC）和逻辑链路控制子层（LLC）；数据帧在 MAC 子层被封装，因此通常被称为 MAC 帧。

在需要增强网络安全隔离的场景下，交换机必须具备识别不同 VLAN 报文的能力，这就要求它能够解析报文中的 VLAN 信息字段。IEEE 802.1Q 协议对此进行了规定，通过在以太网数据帧的 MAC 地址域后、协议类型字段前加入一个 4 字节的 VLAN 标签（VLAN Tag）来标识数据帧所属的 VLAN。如图所示，VLAN 标签中最重要的是 VLAN ID（VID），它是一个 12 位的字段，取值范围从 0 到 4095；另外，优先级（PRI）表示数据帧的 802.1P 优先级，标签协议标识符（TPID）占用 2 字节用来指示数据帧类型，而标准格式指示位（CFI）则用来表明 MAC 地址在不同传输介质中是否以标准格式封装，从而确保与以太网和令牌环网的兼容性。

基于优先级的流量控制（PFC）应用于以太网中，旨在应对网络拥塞并提供无损链路。该功能定义在 IEEE 802.1Qbb 标准中，并通过使用 802.1Q 报头中的优先级（PRI）来控制特定优先级的流量。如果数据包中没有 IEEE 802.1Q 的 VLAN Tag 头，许多 PFC 实现会使用 IP 报头中的 DSCP 值映射到相应的 IEEE 802.1P 优先级值。



66

图99 普通数据帧和 VLAN 数据帧

ETH-X[67]

ODCC 和 ETH-X 项目

在中国通信标准化协会的指导下，致力于打造具有竞争力的生态圈和开放平台开放数据中心委员会（ODCC）成立，推动统一规范和标准的形成。目前，ODCC 的决策组由腾讯、阿里巴巴、百度、中国电信、中国移动、中国信通院、京东和美团组成，会员单位已超过 200 家。

为促进算力产业的发展，ODCC 网络组于 2024 年 5 月启动了 ETH-X 超节点系列项目。该项目由中国信通院与腾讯牵头，联合快手科技、燧原科技、华勤技术、锐捷网络、新华三、云豹智能、云合智网、立讯精密和光迅科技等合作伙伴共同推动，旨在通过产品化样机和相关技术规范，构建大型多 GPU 互联算力集群系统。

ETH-X 是基于以太技术扩展的 Scale Up 互联技术，支持 ETH、UALink 和 OISA 等多种协议。项目输出的规范涵盖了机架、计算节点、交换机节点和互联设计规范，并发布了《ETH-X Scale Up 互联协议白皮书》；2025 年 4 月，ETH-X 超节点首台原型机成功点亮。2025 年 6 月，中国信通院与腾讯联合合见工软等单位成立了“ODCC AI 网络实验室”，并发布了《ETH-X 协议测试报告》，进一步推动了相关技术的发展和应用。

ETH-X 协议框架

ETH-X 协议聚焦于“超节点”（Super-Pod）架构中跨 GPU 的高效数据访问，采用“事务层 – 数据链路层 – 物理层”三层架构，提供了一个基于以太网扩展的 GPU 互联框架：

- **事务层：**定义 PAXI（Peer-to-Peer AXI）协议，实现 AXI 总线在以太网上的传输，支持直接访存（Direct Access）和 直接拷贝（Direct Copy）语义。
- **数据链路层：**包含 PRI（Packet Rate Improvement，报文速率提升）、LLR（链路层重传）、CBFC（基于信用的流控）等机制。
- **物理层：**支持 50G/100G/200G 单通道速率，接口速率涵盖 100G/200G/400G/800G。

当 IO 实现为独立的 Die 时，可 Die-to-Die 互联，即基于标准 UCIe 实现计算芯粒（Compute Die）与 IO 芯粒（IO Die）解耦互联。

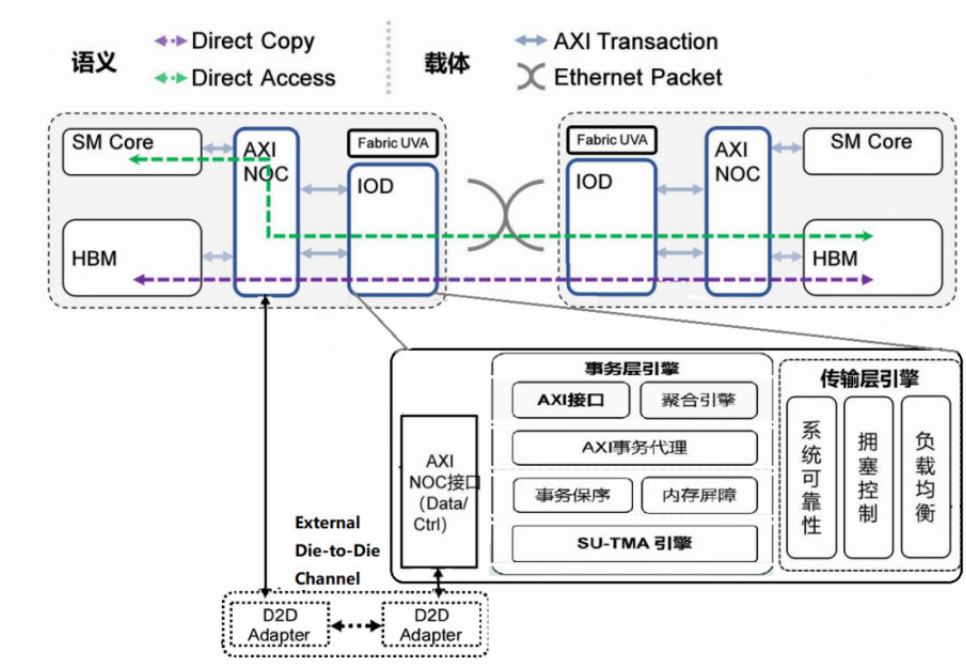


图100 ETH-X 协议栈概览图（参考整理）[68]

事务层 PAXI 框架

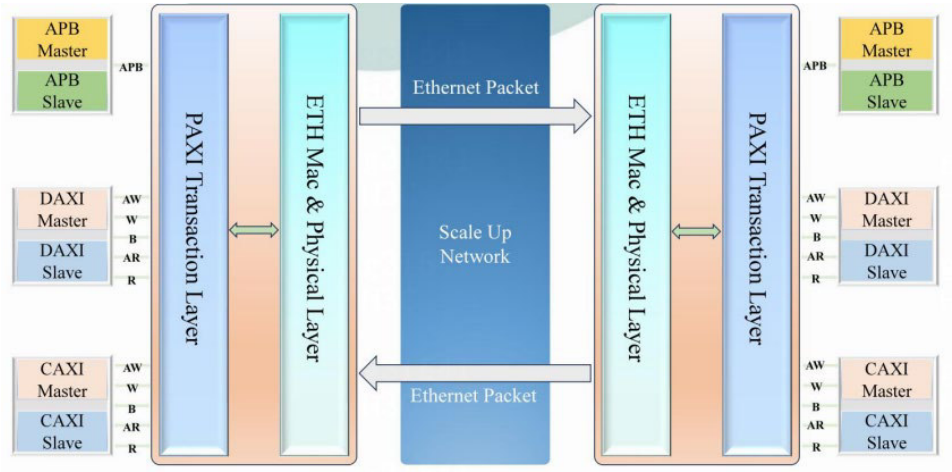


图101 PAXI 整体架构图 [69]

PAXI 系统总共包含**两组标准 AXI4.0 接口和一组 APB 3** 接口，这两组 AXI 接口分别被命名为 CAXI 和 DAXI，它们具备高度的相似性，除了信号位宽可根据具体设计而不同外，并无其他显著区别。其中，CAXI 接口实例化与否完全依据 GPU 的实际需求而定；而 DAXI 接口则是系统运行的必备组件，必须进行实例化以确保系统的正常运行和高效性能。这种设计不仅确保了系统的高度灵活性和可扩展性，还为设计提供了定制化的可能性。

- CAXI 接口主要通过 AXI 在 **GPU 间传递控制**信息；

- DAXI 接口主要通过 AXI 在**GPU 间来传递数据**信息；
- APB 接口用来**端到端传递和交换** GPU 侧 Scale Up 上的**控制信息**，例如信令申请，PAXI 内部寄存器访问等。

GPU 上的 AXI NOC 可直接连接到 PAXI 上，而 APB 接口通常需要一个 APB Bridge 实现协议转换后才可连接到总线上。

1. DAXI 或 CAXI 接口信号处理工作流

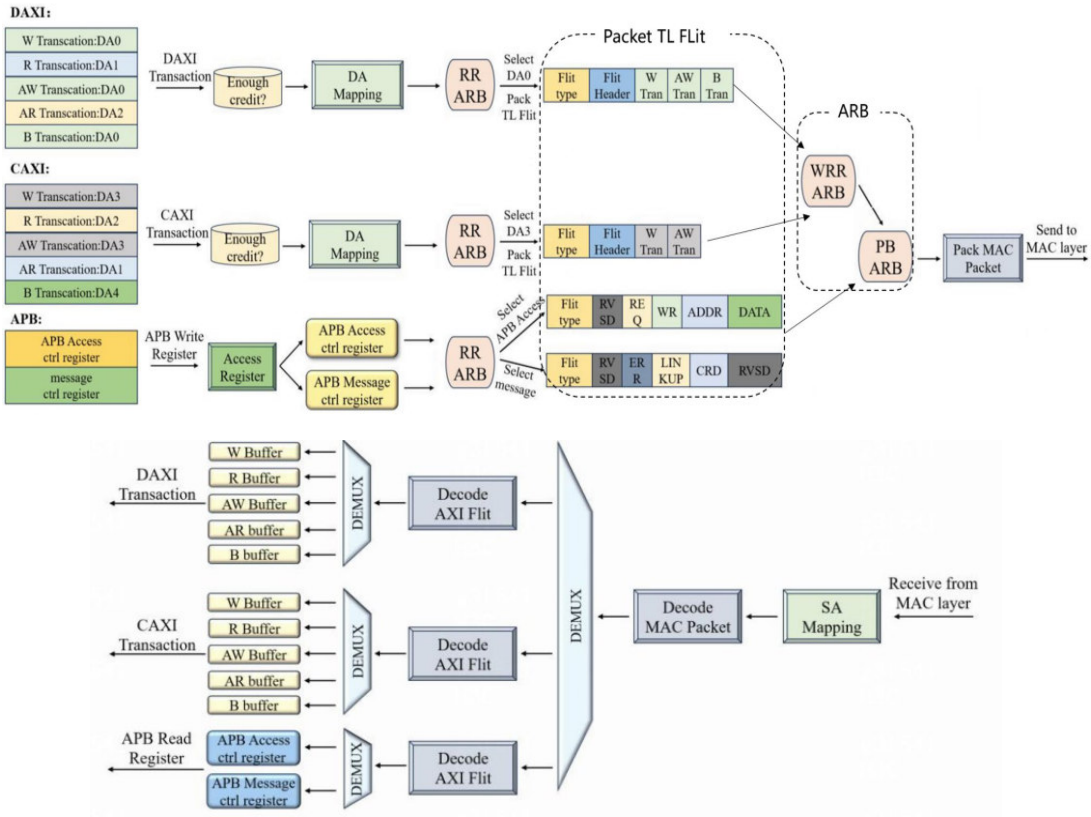


图102 PAXI 发送端（上）和接收端（下）事务层处理流程 [70]

当 GPU 发送远端 GPU 内存访问请求事务或响应事务时，该事务到达 PAXI 事务层时，会呈现为 DAXI 或 CAXI 接口信号，发送处理流程如下：

- 首先判断是否有足够的信用点（credit）允许将该事务发送至目标端的 GPU。如果信用点充足，则接收该事务；若不足，则进行反压操作；
- 接收事务后，PAXI 事务层会查找与该事务相对应的 AXI 通道下的 Userfields 信号。针对 AXI 接口的五个通道，GPU 实例均增加了额外的 Userfield 字段，这些字段位于每个通道的字段域高位。Userfield 中包含事务的目的地址索引（即 GPU ID）。PAXI 根据每个 Userfield 中的值，在网络地址寄存器（DA Mapping Register）中查找对应的目的 GPU 的 MAC 地址，该 MAC 地址将会被写入到 PAXI 帧的 DA 域中。所有网络地址寄存器均可通过 APB 总线进行访问。在系统启动阶段或准初始化状态时，上层软件会将 MAC 地址注册到 PAXI 的网络地址寄存器组以及以太网交换机上，具体的 MAC 地址注册和初始化配置由上层软件决定。

- 鉴于AXI具有五个独立的通道，事务层采用轮询仲裁规则(RR ARB: Round-Robin Arbitration Rule)，选择特定的 DA，并将该 DA 所有通道的事务整合为一个 TL flit 数据包。在每个周期内，DAXI 接口和 CAXI 接口仅分别打包一个 TL flit；如果在同一周期内，有多个不同 DA 的事务需要打包，除了该周期内被打包的事务外，其他 DA 事务将继续在相应的 AXI 通道中等待打包，通过 AXI 相应通道的 ready/valid 握手信号来表明这些通道中的事务是否已被接收和处理。TL Flit 包包含三个部分：Flit 类型、Flit 头部和有效载荷。其中，Flit 类型使用 3 位二进制指示当前 Flit 的类型接口；Flit 头部用 6 位二进制指示当前 Flit 包中包含的 AXI 通道事务；而有效载荷则是这些通道事务的具体信息。并且，有效载荷中各个通道事务的排列顺序是固定的，从最高有效位 (MSB: Most Significant Bit) 到最低有效位 (LSB: Least Significant Bit) 依次排列为 W>>R>>AW>>AR>>B，如图所示。Flit 头部 6 个 bit 定义如图所示，若为 6 'b000000 时，表示该包没有任何事务，为 Nop 包。Flit 头部的 WDATA_TYPE 位域用来表示该数据包写数据是否包含字节使能，当值为 0 时 Sub-Flit 中需要包含 AXI 的 wstrb 信号，Sub-Flit 标识为 WDATA 通道；当值为 1 时，Sub Flit 无需包含 AXI wstrb 信号，AXI 中 wdata 信号所有字节全部有效，此时为 WDATA2 通道。

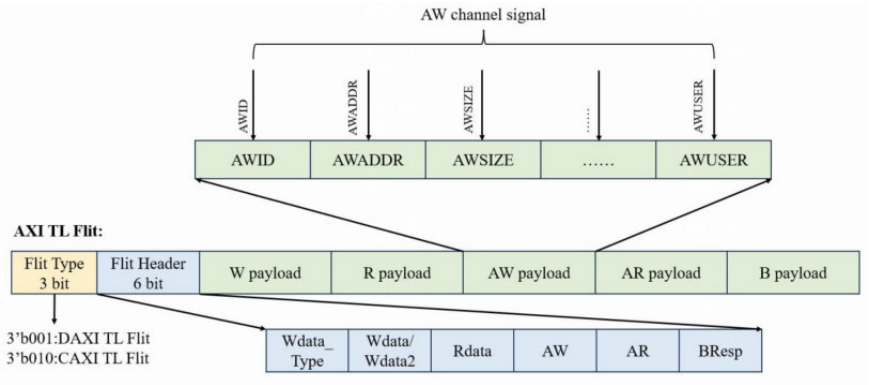


图103 DAXI/CAXI 对应的 TL Flit 包格式 [71]

- 根据 MAC 帧的组包规则，系统会判断当前事务包是否可以合并到现有的 MAC 包中。如果可以合并，则将事务包添加到 MAC 包中；否则，会将现有的 MAC 包发送至 MAC 层，并将该周期的 flit 事务作为新的 MAC 包处理。在处理 CAXI TL flit 和 DAXI TL flit 时，采用加权轮询仲裁规则，用户可以自由设置 CAXI 和 DAXI 的权重比例，权重范围在 1 至 255 之间。MAC 包组装完成后，会被发送至 MAC 层，进而组成 MAC 帧，最终发送到以太网网络中，确保数据传输的准确性和效率。

2. APB 事务处理工作流

在 APB 事务处理过程中，Scale Up 内部的一系列状态控制寄存器起到了关键的触发作用。这些寄存器的交互通过 APB 接口协调完成，确保各类事务能够顺畅进行。在 APB Message 事务中，首先需要对 Message Control Register 进行设置，然后将相关信息打包成 APB Message TL flit；这类事务不仅承载着信息传递的功能，还在配置 Credit 寄存器中发挥重要作用，从而实现对事务层信用的有效管理。对于 APB Access 事务(包括读和写操作)，则需要依赖 Remote APB Control Register 和 Remote APB R/DATA Register 的协同工作。最终，这些操作会被封装为 APB Access TL flit。

在 TL flit 进行仲裁并组装成 MAC 包的过程中，采用了严格的优先级 (Strict-Priority) 仲裁机制。这种机制确保了 APB TL flit 具有最高的优先级，即在同一周期内，如果同时存在 APB TL flit 以及 CAXI、DAXI TL flit，

仲裁器会优先选择 APB TL flit 进行 MAC 包的组装。这种设计保证了 APB 事务在系统资源竞争中的优先处理地位，从而提高了系统的整体效率和响应速度。

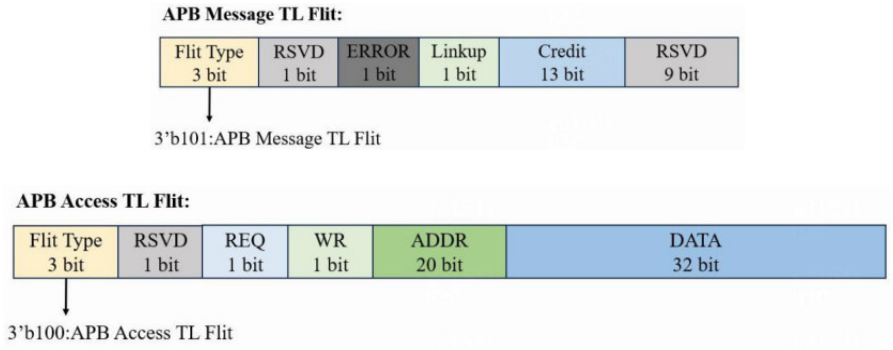


图104 APB TL Flit 格式 [72]

3. 事务层流控 Credit 机制

PAXI 架构支持全面的端到端流量控制，专门设计了一组 32 比特的 Credit 配置寄存器，每个目的 GPU 一个 Credit 寄存器，其中高 16 位用于 AR Credit，低 16 位用于 AW Credit。这一机制为每个 GPU 维护了当前剩余的 AR Credit 和 AW Credit 数目，以监控并限制未完成的以及当前并发的读事务和写事务数量。当 DAXI 或 CAXI 接口有事务待发送时，仅在 Credit 充足的情况下，该事务才能被传输至目的 GPU，从而确保流量的平稳流动。

PAXI 通过 Credit 动态更新来进行流控。在系统启动阶段或者准初始化态阶段，PAXI 对剩余 AR credit、剩余 AW credit 配置初始值，即最大 AR credit、AW credit 数目。每当 PAXI 发送一笔新的读事务请求时，会将对应目的 GPU 剩余 **AR credit 数量减 1**，每当收到读事务响应时，会将对应目的 GPU 剩余 **AR credit 数量加 1**。同样每当 PAXI 发送一笔新的写事务请求时，会将对应目的 GPU 剩余 AW credit 数量减 1，每当收到写事务响应时，会将对应目的 GPU 剩余 AW credit 数量加 1。

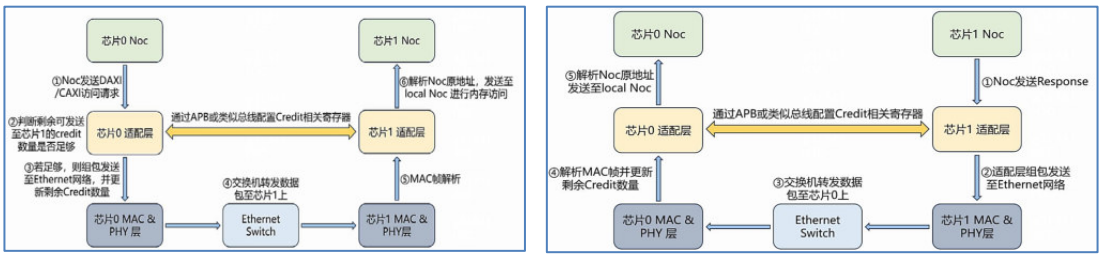


图105 利用 Credit 控制访问请求（左图：Credit 数量减 1）、响应（右图：credit 数量加 1）的流程图 [73]

数据链路层优化技术

GPU-Switch 互通协议层面，Eth-X 定义了 PRI 帧头格式，采用了 UEC 的 L2 LLR 增强可靠性，允许与 PFC、CBFC 等流控技术进行协同流控。

1. 帧头 PRI 格式

Eth-X 通过重新定义网络报文格式，即帧头 PRI 格式，显著优化了转发效率并提升了整体性能。新的 PRI 统一转发头，由 12 字节组成，取代了传统的 DMAC 和 SMAC，并保留了“Network DeviceId（路由字段）、TTL（生存时间）、DSCP（差分服务代码点）、ECN（拥塞标识）”等核心字段。PRI 使用设备地址，在 DA 域(UDA，即 User Defined Address，共 10 字节)中进行设备内部寻址。其中包含的流标识(Flow Entropy)，其字段位置、长度及掩码由 GPU 上层实现指定，这可能因不同的 GPU 实现而有所差异，目的是用于后续的负载均衡。值得注意的是，DA 域会被网络转发节点忽略，而对流标识的解析则需要交换机侧针对不同 GPU 进行定制化处理。

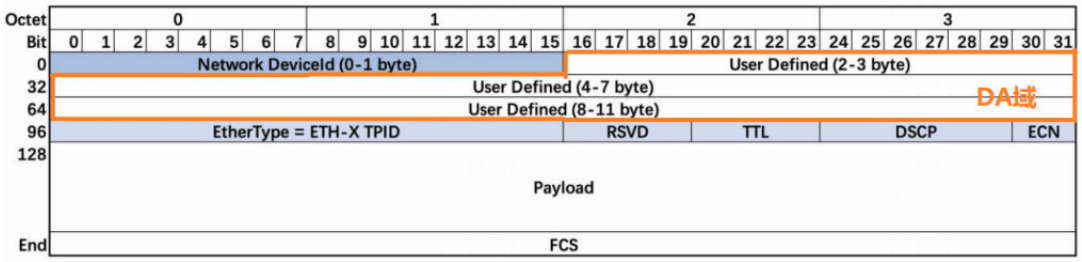


图106 ETH-X PRI 帧格式 [74]

在 Eth-X EtherType 之前，增加四字节的 VLAN Tag，从而实现 Eth-X 增强帧（VLAN+PRI）。这一格式不仅最大程度地兼容了现代以太网交换机，而且 GPU 可以利用这种增强帧来实现报文的压缩，以及安全隔离与多租户双重功能，为系统解决方案提供更大的灵活性与扩展性。在 Eth-X 增强帧中，PCP（Priority Code Point）的码点对应 IEEE 802.1Q 标准中 VLAN tag 的 PRI 部分。当 GPU 端发出增强帧时，VLAN tag 内的 PCP 字段和 PRI 域内的 DSCP 字段都需要写入相应的 QoS 标识，确保 QoS 信息的一致性。

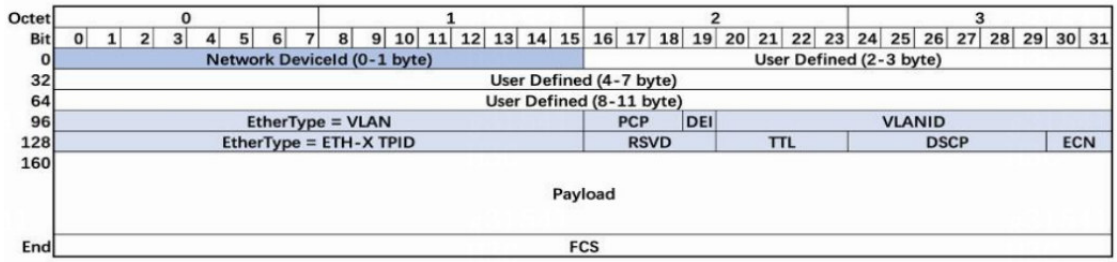


图107 ETH-X VLAN+PRI 帧格式 [75]

2. 数据链路层的 CBFC

数据链路层的 CBFC 和 LLR 机制，参考 UEC 的相关机制实现。数据链路层的 CBFC 是基于虚拟通道 VC 进行的，每个 VC 的信用释放数量（CF，Credit Freed）和消耗的信用数量（CC，Credit Consumed）用 2 组循环计数器记录，通过跟踪 CF 和 CC 进行流控。事务层和数据链路层 CBFC 机制说明可参考下表：

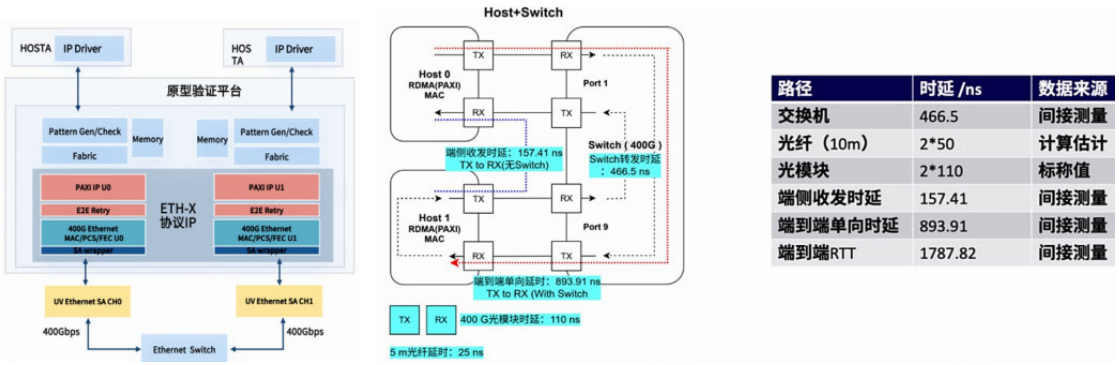
协议层次	事务层	链路层
作用对象	端到端（Peer-to-Peer）的内存读写事务	逐跳（Hop-by-Hop）的物理链路传输

协议层次	事务层	链路层
控制变量	每个GPU的AR Credit、AW Credit	每个VC的CC、CF
影响维度	并行度（Outstanding 能力）	吞吐量（Throughput）

性能指标及测试情况 [76]

依据《ETH-X Scale Up 互联协议规范》，ODCC AI 网络实验室基于全场景硬件验证系统 UVHS，搭建超节点组网的原型测试平台。该平台完整实现了 ETH-X 事务层、链路层及物理层协议，并与国内领先的 51.2T 交换机通过 400G 接口组网，全面验证了 PAXI（包括 PRI）增强承载在典型场景下的可行性及性能指标。

维度	设计目标	实测结果
协议延迟	PAXI + 以太网MAC/PCS/FEC + Serdes < 150 ns	ETH-X PAXI IP时延为157.41 ns
带宽利用率	以太网带宽利用率可达99%以上	1K报文读写分别达到91.25%和 91.5%
协议承载效率	支持内存语义操作	小荷载(128Byte)时承载效率优于RoCEv2
组网验证	支持通用以太网交换机实现256/512卡组网	4台主机,400G端口,端到端单向时延经间接测量为893.91 ns



路径	时延 /ns	数据来源
交换机	466.5	间接测量
光纤（10m）	2*50	计算估计
光模块	2*110	标称值
端侧收发时延	157.41	间接测量
端到端单向时延	893.91	间接测量
端到端RTT	1787.82	间接测量

图108 ETH-X 原型测试平台（左）和测试路径时延说明（右）

SUE/ESUN [77]

OCP 和 SUE、ESUN 项目

在 2024 年 4 月，博通 Broadcom 首次提出了 SUE（Scale-Up Ethernet）架构。该架构倡导以标准以太网为基础，通过强化链路层和传输层来实现高性能的 Scale-Up。到了 2025 年 10 月，在开放计算社区 OCP（Open Compute Project）的全球峰会期间，Broadcom 将 SUE 规范贡献给了 OCP。OCP 成立了 SUE-T（Scale-Up Ethernet-Transport）工作组，致力于将其推动为开放标准。与此同时，OCP 启动了更底层

的 ESUN（Ethernet for Scale-Up Networking）项目，重点关注 L2/L3 层，而 SUE-T 则被定位为 ESUN 之上的可选传输层协议。ESUN 的成立得到了 AI 领域几乎所有关键厂商的支持，其工作组创始成员包括 AMD、Arista、ARM、Broadcom、Cisco、HPE、Marvell、Meta、Microsoft、NVIDIA、OpenAI 和 Oracle 等 12 家企业。

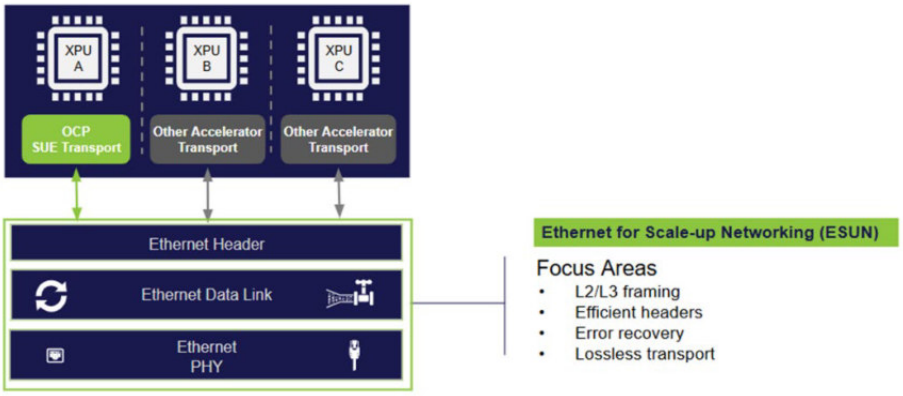


图109 OCP 的 ESUN 和 SUE-T 项目 [78]

SUE 协议框架

技术架构上，SUE 采用类 AXI 的双工数据接口，通过虚拟通道（VC）将事务映射到不同流量类别，支持两种传输模式严格有序模式（保序模式）和无序模式（多端口负载均衡）。其标准的 SUE 协议栈分为三层：

- 事务层：负责映射与打包层（Mapping and Packing），即将来自 XPU 的多个小事务按目的地和虚拟通道（VC）进行聚合，形成最大 4096 字节（4KB）的协议数据单元（PDU），以提高线路效率。
- SUE 可靠传输层：负责端到端的流量控制、可靠性和保序（SUE Lite 中此层被移除）。
- 网络层：构建网络头部，支持标准以太网/IP/UDP 格式或专有的 AI Fabric Header（AFH）。而数据链路层技术包括 MAC、链路层重传（LLR）和基于优先级的流控（PFC）或基于信用的流控（CBFC）。

SUE Lite 旨在将 SUE IP 的面积最多减少 50%，移除了 SUE 的可靠传输层（因此拥塞控制功能也随之被移除），改用节点间 LLR（链路层重传）。为进一步减小面积，数据包大小被限制为 标准 SUE 的 1/4，即 1KB；并且仅保留一个信号接口而不支持 AXI4 协议接口了。

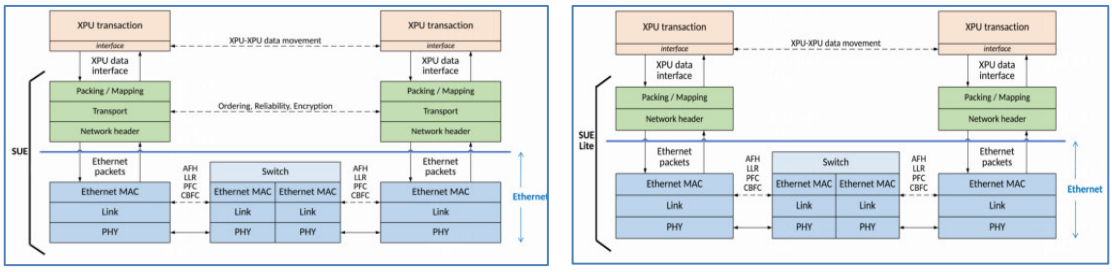


图110 SUE 标准协议栈（左图）和 SUE Lite 协议栈（右图） [79]

Attributes	SUE	SUE Lite
Transaction types	Write (Full/Partial, Posted/Non-Posted), Read, Message, Barrier	
Transaction size	256B	
DMA ready interface	Yes (Block write)	
Transaction packing	Yes	
LLR, CBFC, AFH Headers	Yes	
Core side interface	AXI 4, Signal-based interface	Signal-based interface
End-to-end reliability	Yes (Relaible Transport Layer)	No (Hop-by-hop LLR only)
Congestion Control	Yes (Fixed Window)	No
Partition Feature	Yes	No (Use address separation)

图111 SUE 标准协议和 SUE Lite 协议对比 [80]

SUE 接口

1. SUE 实例构成

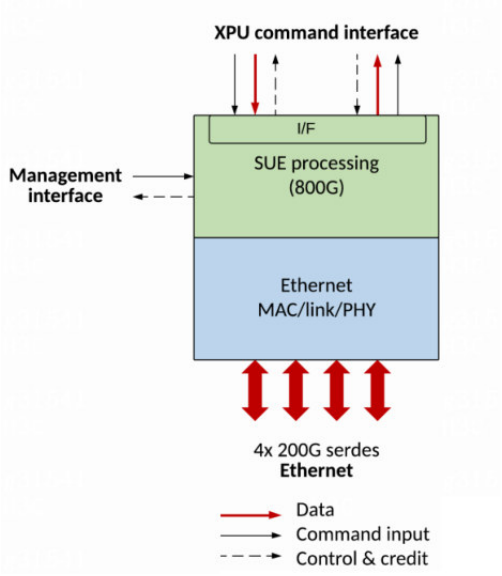


图112 SUE 接口示意图 [81]

每个 SUE 实例均包含三个接口，如图 所示：

- XPU 命令接口：支持基于信用（credit）机制的 FIFO 信号线接口（Wire Interface）或 AXI 4 接口，取决于 XPU 自身实现而确定；命令接口的作用是传输事务指令及数据（控制字段 144 bit，包含操作指令码、长度以及目标 XPU ID）；
- XPU 管理接口： AXI 从接口（SUE 为目标设备），通过对寄存器访问和配置实现配置。也可作为应急通道，进行数据不可靠传输，速率可达每秒 10K 个数据包；
- 以太网接口：支持 50G/100G/200G SerDes。

XPU 命令接口在实现 CBFC 上是有区别的。为了兼容部分不支持 AXI4 的 XPU，FIFO 信号用 6bit 的信用字段为每个 VC 维护信用值；而在 AXI4 接口中，每个主从通道都关联一个独立的信用接口，为了防止接收方缓冲区溢出，接收方通过信用接口将信用额度传递给发送方。

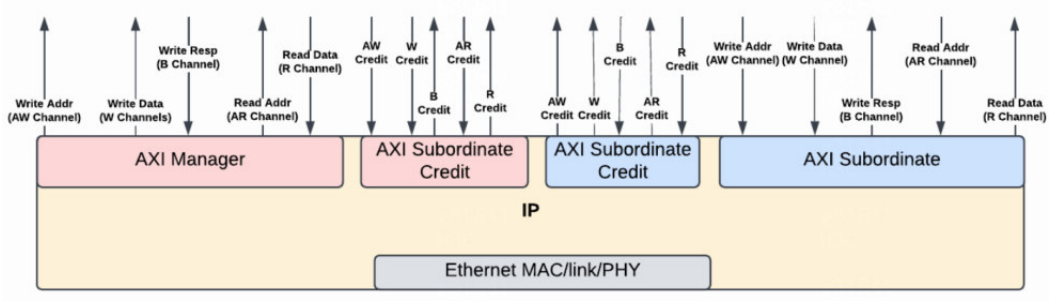


图113 SUE 主从 AXI 通道接口

2. SUE 处理工作流

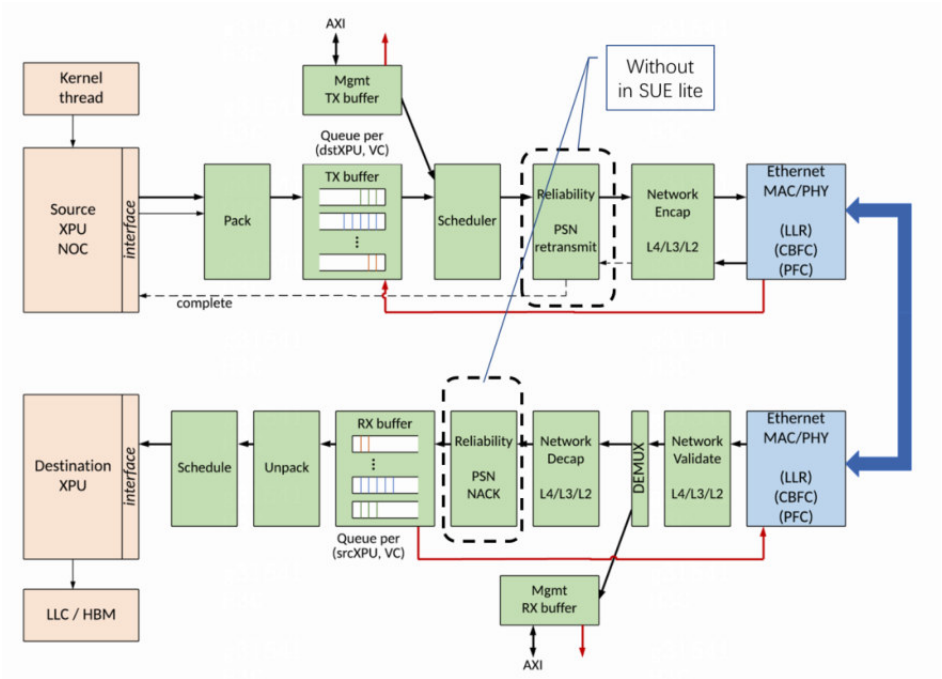


图114 SUE 工作流程示意图 [82]

SUE Lite 处理流程上没有 Reliability 处理步骤，其他与标准 SUE 处理流程相同。下面以标准 SUE 处理流程为例进行说明：

- XPU 内核通过 XPU 命令接口（AXI 4/FIFO 接口）向 SUE 发出命令（含目的 XPU ID、VC、数据）。SUE 接受该操作，并将控制信息和数据（如果存在）打包到每个目的地的发送缓冲区中。控制信息的第一个字节是命令类型和长度；根据命令类型定义，可知该命令是否存在数据，以及对应该命令的控制信息长度（以 2B 为单位）。如果存在数据，则定义数据长度。基于这两个长度，对命令（控制信息和数据）进行 SUE payload 打包。
- 缓冲区功能根据打包情况和流控制状态（例如，CBFC 输入）确定何时应处理队列，并向调度器指示队列何时符合条件。调度器在 VC 之间提供加权轮询调度，在 VC 内部基于调度实现到达保序。

- 每个打包的命令组都会在头部添加可靠性头部（RH: Reliability Header，8 字节）并在尾部添加 CRC (R-CRC，4 字节)，从而构建 SUE PDU。RH 中 npsn 字段是根据{输出端口，目标 XPU}分配的包序号 PSN；PSN 为单调递增的，每个数据包递增 1。若需发送 ACK 或 NACK，可将其添加到 RH 的 aspn 字段中。

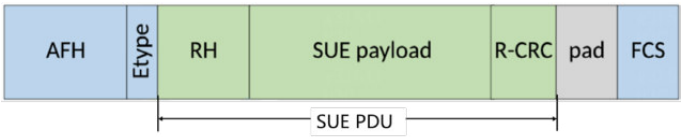


图115 封装后的 SUE PDU 格式

- XPU 之间基于每个物理端口建立连接；目标 XPU 和 VC 用于查找所需的地址字段，以太网数据包被发送并到达目的地。SUE 提供保序，使用无损 traffic 类和 LLR 显著降低了丢包的可能性；如果 SUE 检测到丢包事件，它将使用 Go-Back-N 进行恢复。
- 目的端执行以太网有效性检查，并将所有发往控制接口的数据包进行解复用。验证以太网报头，然后检查 RH 报头。如果收到预期的 PSN（源 XPU，VC），则将 PDU 放入接收缓冲区。如果收到非预期的 PSN，则向源发送 NACK，并且丢弃来自源的任何数据包，直到收到预期的 PSN。
- 接收缓冲区接收命令，在调度器安排下进行解包并发送到目的端 XPU NOC。

SUE 数据链路层

1. 三种封装头 AFH

SUE 会根据情况将接收到的事务打包到同一{目标地址, VC}，从而创建 PDU 单元。封装后的标准 SUE PDU 单元最大长度 4096 字节，而 SUE lite 由于没有传输层所以没有 RH 头，进一步将 R-CRC 也优化掉了，SUE Lite PDU 单元最大长度为 1K 字节。标准 SUE 支持灵活的网络封装，支持三种模式，

- 标准以太网格式：使用以太网头部、IPv4/IPv6 和 UDP。
- AI 转发头部第一代(AI Forwarding Header Gen 1): 一种 Layer 2 格式，保留了现有以太网 MAC 目的地址和源地址的格式，但仅需使用 MAC 地址中的较少位数（16 至 32 位）。
- AI 转发头部第二代(AFH Gen 2): 一种更优化的 Layer 2 格式，将转发信息压缩至 6 字节或 12 字节。太网目的地址和源地址中剩余的字节可用于用户自定义功能。Ethertype 字段用于区分多种传输层头部。

表14 三种 AFH 模式对比

特性	标准以太网头	AFH Gen1（重定义未压缩）	AFH Gen2（深度压缩）
典型用途	兼容传统网络	初代 SUE GPU 互联	OCP 2025+ 主流 AI 集群
基础格式	IEEE 802.3	保留MAC域，并替代IP、UDP	在Gen1基础上通过FLAGS压缩头
XPU寻址方式	标准 MAC	标准 MAC	AAI 编码 MAC (SLAP)

特性	标准以太网头	AFH Gen1（重定义未压缩）	AFH Gen2（深度压缩）
识别SUE包	用UDP端口号	特定Ethertype标识SUE包	通过FLAGS识别SUE包
基础PFC承载（有无VLAN）	有VLAN标识，用VLAN Tag的PRI		
	用IP包头DSCP	有Shim，Shim头的DSCP；否则无承载	无具体承载

标准以太网头格式

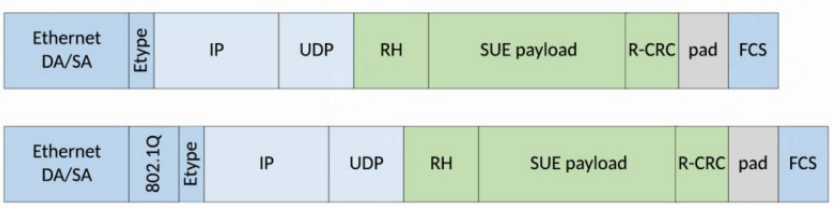


图116 标准以太网格式和携带 VLAN 的格式 [83]

AFH Gen1

第一代 AFH 使用标准的以太网 MAC 目标地址和源地址，但交换机硬件只需查看地址中的 16 位到 32 位即可执行转发决策。因为 XPU-id 被映射到 16 位目标地址，用于提升转发查找效率。如果需要区分多种传输层头部，可在封装包中添加第二个可选的 Ethertype 字段，例如 Shim 头（Etype L4）；而对非 Shim 格式的流量进行优先级映射，需要向帧中添加 IEEE 802.1Q VLAN 头部。其中 Shim 头定义通过 F 位区别字节头大小，F 为 0 时对应 2 字节 Shim 头，F 为 1 时对应 3 字节（增加了 1 字节的 Flow Label）。

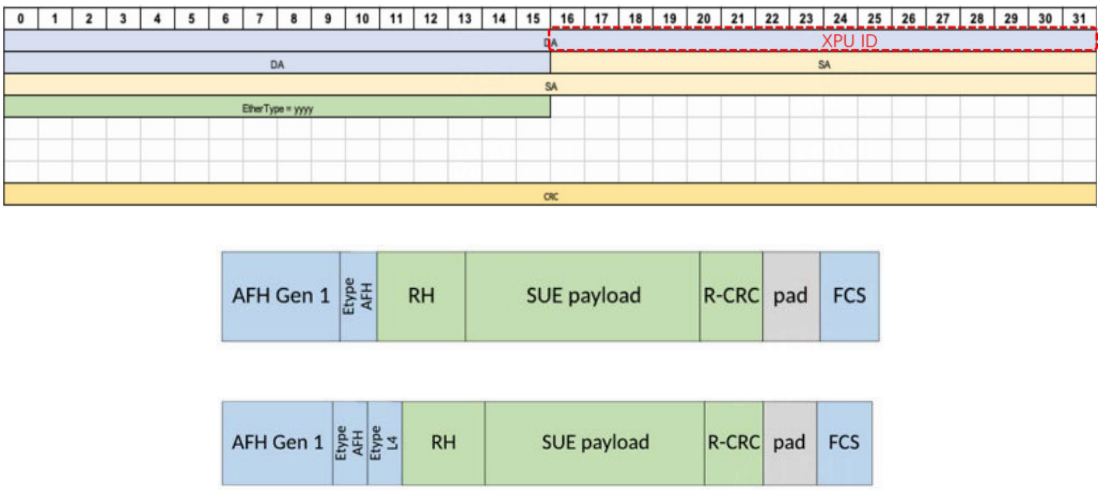


图117 AFH Gen1 标准格式和携带 Shim 头格式报文框架 [84]

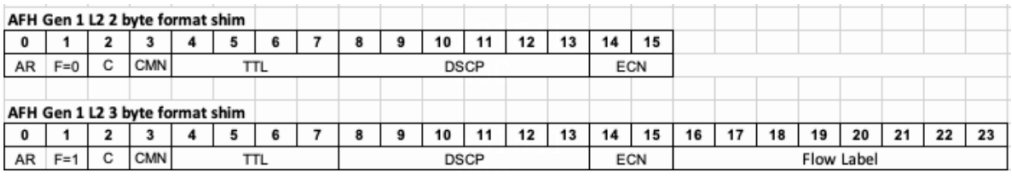


图118 AFH Gen1 中 Shim 头格式定义 [85]

AFH Gen2

AFH Gen 2 是基于 AFH Gen1 优化的报头，旨在提供更小的网络报头。AFH Gen2 压缩字段包括 MAC DA、SA、Etype 域、IP 和 UDP，参考下图；而在 AFH Gen2 中 Etype 字段是用于区分多种不同的传输层报文头。AFH Gen2 中使用符合 IEEE 标准的编码的管理分配标识符 (AAI)，XPU 标识符被映射到标准 32 位或 压缩的 16 位值，对应 12 字节和 6 字节报头。AFH Gen2 在 MAC DA 位置定义了 FLAGS 标志位，定义如下：

- V：版本，当前为 0（1 未定义）；V=0 时，W 为是否压缩，1 为压缩格式（不带跳数和熵信息），0 为非压缩的常规格式（带有跳数和熵信息）；
- M：0/1，1 为多播；
- X = 1，本地分配地址；
- Y = Z = 0（基于 SLAP 的 AAI 格式编码地址）

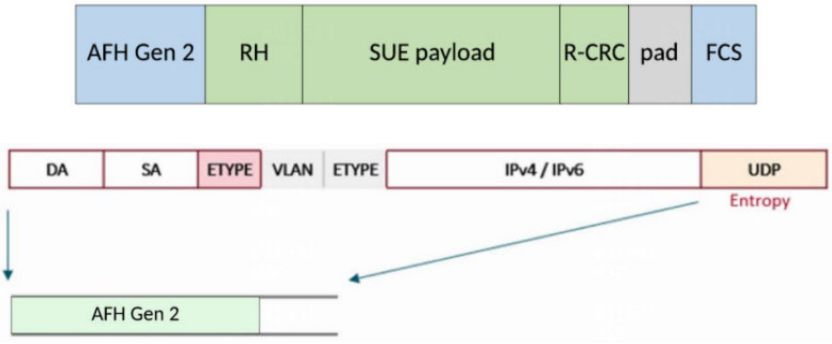


图119 AFH Gen2 头的报文框架和 Gen2 头压缩字段示意 [86]

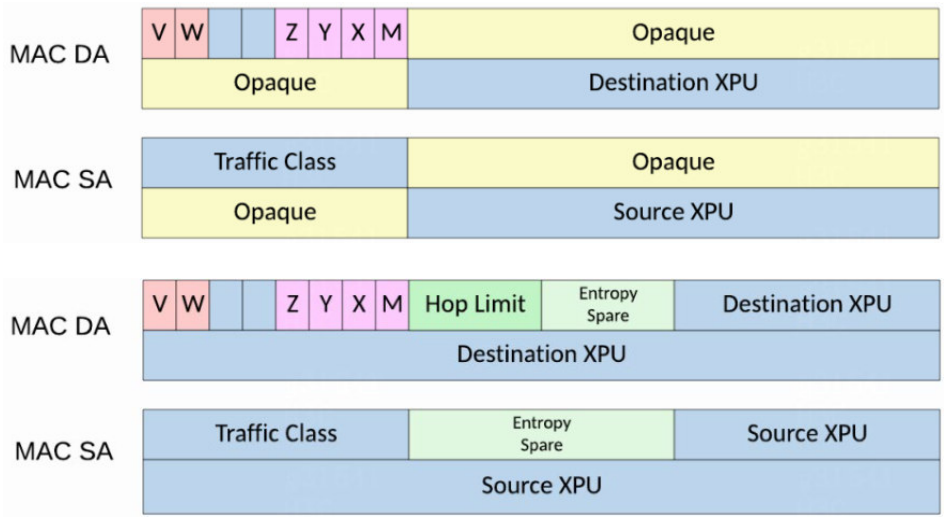


图120 AFH Gen2 头，上图：6 字节 (16 位 XPU ID), 下图：12 字节（32 位 XPU ID） [87]

2. SUE 封装包 [88]

XPU 与 SUE 的接口速度往往高于 SUE 与以太网接口速度，这种速度上的不匹配往往会导致事务在 SUE 中每个目标地址的队列中不断累积。为防止 SUE 内部缓冲区溢出，SUE 在多个虚拟通道（VC）之间实现了流量控制机制。事务依据目标地址和流量类别，在 SUE 中的缓存队列中分别排队等候处理。SUE 的调度器会

逐一检查这些队列，寻找满足条件的事务队列进行优先处理。SUE 并不会等待队列中的事务累积到某个预设的大小（例如 2K，最大 4K），而是依据当前实际情况将事务进行打包处理：当某个队列中存在多个事务时，SUE 会将它们合并至同一个协议数据单元（PDU）中；如果目标地址的队列中仅包含一个事务，则会直接发送该事务。通过这种灵活的调度机制，以及 AFH Gen2 对帧头开销的优化，SUE 有效提升了网络传输效率，并最大限度降低了事务处理的延迟。

下面通过图示进一步说明这一打包过程：XPU 向 SUE 发送事务 A 至 E，这些不同的事务携带有不同长度的数据。在 SUE 内部，这些事务依据其目标地址进行排队，由于事务 E 和事务 C 的目的地相同，因此它们被排入同一队列中，并随后一起被打包发送。调度器首先选择符合条件的事务队列，即包含事务 A 的队列，并为这些事务添加资源头（RH）和冗余校验码（R-CRC），从而形成 PDU 单元，再经过以太网帧封装 AI 头和帧校验序列（FCS）后，通过网络接口发送出去。在发送完事务 A 后，调度器接着选择包含事务 B 的队列并发送事务 B。随后，调度器选择包含事务 C 和事务 E 的队列进行处理，并继续选择包含事务 D 的队列发送事务 D。在接收端，SUE 负责解封装接收到的以太网头部，提取出其中的命令和数据序列，并将其通过 XPU 命令接口转发给目标 XPU。这种高效的处理流程确保了数据传输的顺畅与高效。

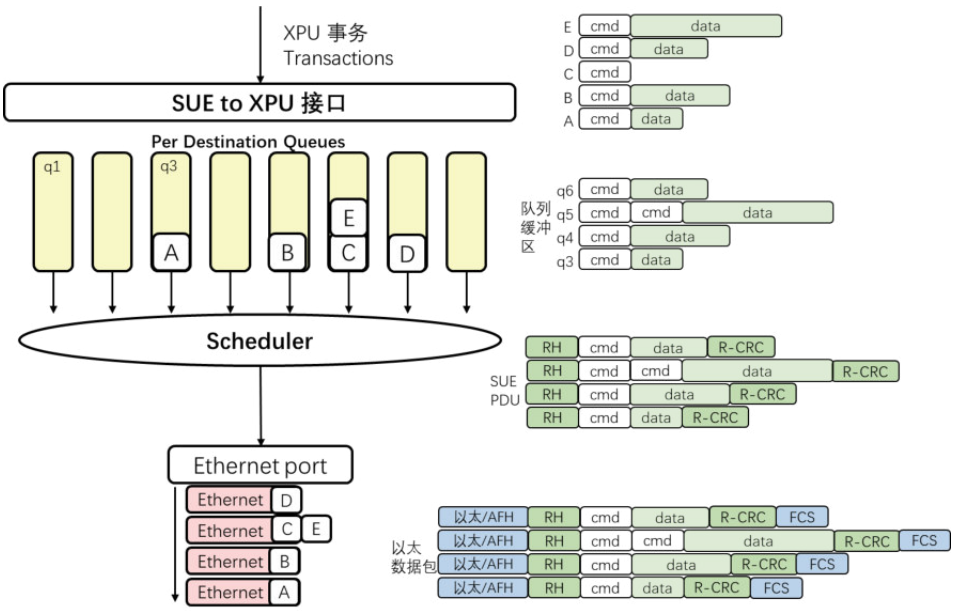


图121 SUE 封装包流程和包格式示意图 [89]

性能及理论推算

SUE 的性能设计指标 E2E 延迟为 2us（即 2000ns），还特别规定了 SUE 到交换机的线缆长度不能大于 10m。而根据线缆延迟估算，10m 空心光纤的延时为 35ns，SUE NOC 到以太延时 小于 100ns，以太链路到 PHY 延时小于 100ns，交换机内部延时小于 250ns，则 SUE 端到端协议延时为 100 + 100 + 35 + 35 + 250 = 520 ns。

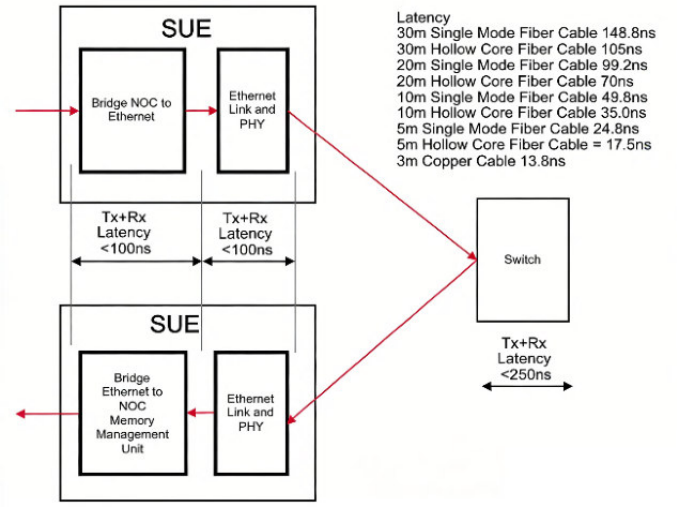


图122 理论延时推演示意图 [90]

在 2026 年 1 月的超节点大会上，合见工软宣称已完成与博通 Tomahawk Ultra 交换芯片的 SUE 协议对接测试，验证了 LLR、CBFC、AFH 等功能，首家完成 SUE 认证的商业 IP。

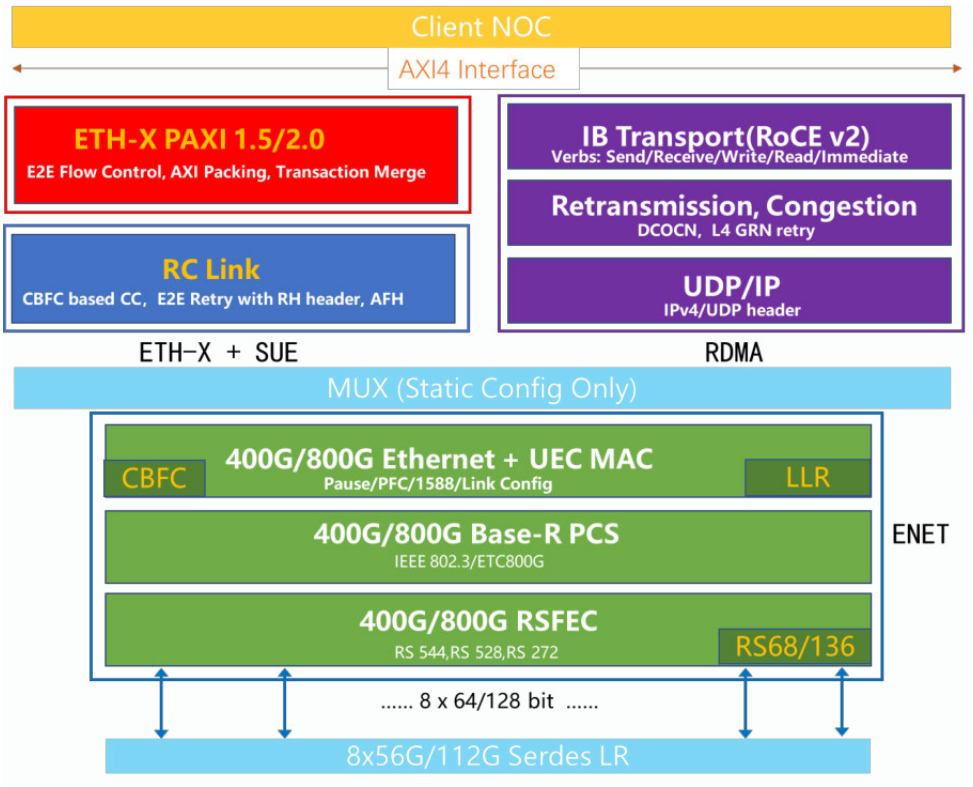


图123 合电工软底层 IP 框架示意 [91]

3.4 OSIA 协议介绍 [92]

中国移动立足智能化发展趋势打造 OISA 实现开放互联

战略牵引：保障自主可控与安全

作为国家信息基础设施核心之一，中国移动希望突破对单一外部互联技术的依赖。打造 OISA 是其构建自主可控算力技术栈的关键举措，旨在防范供应链中断与风险，确保在国家关键领域的 AI 算力服务安全、连续、可靠，履行央企的战略责任。

技术驱动：破解大规模算力效率瓶颈

面对千卡/万卡集群的通信延迟与扩展性难题，传统方案无法满足 AI 训练的超低时延与高并发需求。中国移动必须通过 OISA 重构高速互联架构，实现对算力系统从芯片层到调度层的端到端深度优化，从而将网络资源优势转化为可交付的、具备确定性性能的高效算力服务。

生态卡位：夺取未来产业规则定义权

若仅采用或改良现有标准，将持续处于产业跟随者地位。主动定义 OISA 这一开放互联协议，可促进中国移动深入产业链核心，有效牵引 GPU、NPU、交换芯片及软件生态协同发展。中国移动通过构建以自身标准与服务为核心的繁荣生态，亦可实现从成本中心到价值与规则输出中心的升级。

基于前述三点，面对人工智能从单节点向万亿参数规模异构集群的演进，计算架构的重心正从“提升算力”转向“优化连接”，中国移动希望通过 OISA 刺破“瓶颈”，为业界提供一种开放的、能够支撑超级节点大规模协同的互连标准。

OISA 技术体系设计原则及主要技术特点

OISA 技术体系设计原则

在中国移动规划中，OISA 旨在成为新一代智算互联的标准化协议体系，其核心是构建一个扁平化、去中心化、高效开放的 GPU 卡间互联架构。OISA 将专注于突破 GPU / CPU 间内存数据交换的瓶颈，通过联合伙伴共研 Scale-UP 协议标准实现大规模 AI 训练与推理提供有力支撑。

图124 OISA 技术设计六大原则

OISA 协议体系的总体设计原则体现了中国移动及其合作伙伴面向未来 AI 算力需求的前瞻性思考和系统化理念。这些原则包括全向连接、极致访存、无损传输、协同计算、智感互联与开放生态构建，为构建高效、智能、开放的智算互联生态奠定坚实基础。

- **全向连接：**该原则要求在 Scale-UP 域内支持直连与交换拓扑的全连接形态，确保任意 GPU 节点间可建立无迂回、对等的 P2P 通信路径，实现转发路径最短化与带宽损耗最小化。
- **极致访存：**该原则要求通过高带宽（支撑 TB/s 高速互连）、精简报文格式、全向连接技术和直通转发模式等手段，最大程度压缩 GPU 间通信延迟（保障百纳秒级别通信延迟）。
- **无损传输：**该原则要求数据传输过程中应避免丢包和错误的发生，在发生丢包或数据传输错误时，应通过流量控制、纠错、重传机制进行快速恢复。
- **协同计算：**该原则要求打破存储壁垒，将 GPU 显存、系统内存整合为统一的共享存储空间，并通过统一内存寻址方式实现资源直接访问。同时，通过交换芯片完成部分计算任务的卸载与加速，减少 GPU 间的通信流量，提升整体性能。
- **智感互联：**该原则要求实现 GPU 间通信状态的感知，实时监控网络状态并智能调整流量，使联接成为具备自我洞察与动态调节能力的智能网络系统，为计算任务提供确定性保障。
- **开放生态：**该原则要求 OISA 成为开放的行业标准，向合作伙伴和开发者开放规范，确保不同供应商开发的 GPU 芯片、交换芯片及其他 OISA 设备实现互联互通，规避供应商锁定风险，鼓励良性竞争与协同创新。

OISA 的六大总体设计原则形成了完整的技术闭环：从全向连接的架构设计到极致访存的性能优化，从无损传输的可靠性保障到协同计算的效率提升，再到智能随路感知的动态优化和开放生态的产业共建，共同构成了支撑新一代 AI 算力互联的核心技术框架。

OISA 技术体系的技术特点

图125 OISA 技术体系五大特征

OISA 协议体系在设计上充分考虑了 AI 算力集群的特殊需求，通过与多项核心技术特征的有机结合，构建了一套高效、智能、可靠的 GPU 互联解决方案。这些关键技术特征从协议架构、传输性能、可靠性保障到智能化感知等多个维度，全面支撑了大规模分布式 AI 计算场景下的高性能互联需求，为构建下一代智算基础设施奠定了坚实基础。

- **多模内存访问：**OISA 定义了指令直驱（ODLS-OISA Direct Load/Store）和直接内存访问（ODMA-OISA Direct Memory Access）两种内存访问模式。**ODLS 模式面向细粒度、低延迟访问场景，适用于短报文快速响应；ODMA 模式面向粗粒度、高吞吐量的数据搬运场景，适合 AI 芯片间大规模数据传输**。两种模式共享统一地址空间，简化编程模型的同时提升访问效率，为异构计算提供灵活的内存操作接口。

- **集合通信加速与协同计算：**OISA 将传统由 GPU 执行的集合通信操作卸载至具备计算能力的交换芯片，实现网络内计算。**该机制支持多种通信原语的硬件级加速，使交换芯片从被动转发设备转变为智能协同计算单元，有效降低 GPU 负载**，提升整体系统性能，交换芯片作为中间计算节点，在数据转发路径中完成 Reduce、All Gather 等操作，显著减少通信量和延迟。

- **智能感知与动态优化：**OISA 引入随路感知机制，通过在报文中嵌入感知标签，可获取链路、节点、队列等关键状态信息，并反馈给源端 GPU，基于这些信息，通过精确识别性能瓶颈，端侧支持动态调整发送策略，网络支持调整转发路径，实现"感知-反馈-优化"闭环控制。这种智能随路机制大幅提升了传输效率和系统鲁棒性。

- **确定性传输与可靠性保障：**OISA 构建了多层次的确定性无损传输体系，通过基于优先级的流量控制（PFC）、缓存感知的流量控制（BFC）、FEC 前向纠错和数据层重传机制（DLR）实现点对点快速重传，形成完整可靠性保障链条，为大规模集群稳定运行提供坚实基础，满足 AI 训练对数据可靠性的严苛要求。

- **报文聚合与即时反馈：**OISA 通过报文聚合机制将多个短报文合并传输，显著提升载荷率和带宽利用率，解决小报文多导致的转发效率低下问题。通过“先响应再处理”机制，允许发送端对满足条件的事务提前返回响应，在高并发场景下降低延迟，及时释放缓存资源。这两项技术协同工作，特别适合混合负载场景下的高效运行。

OISA 的关键技术特征体现了全栈协同的设计理念，通过多模内存访问满足多样化应用需求，借助集合通信加速实现计算与网络的深度融合，利用智能感知和流量控制构建可靠无损网络。各项技术相辅相成，共同实现了低延迟、高带宽、高可靠性的互联目标，为大模型训练、推理等 AI 核心场景提供了强有力的底层支持，展现出卓越的技术先进性和产业适用性。

OISA 技术框架与报文实现

OISA 技术体系框架

OISA 协议框架是整个技术规范的核心基础，截至 OISA2.0 版本规范发布，已定义出协议的分层结构、功能组成、访问流程和互联拓扑等关键要素。该框架为构建高性能、低延迟、高可靠性的 GPU 互联系统提供了架构指导，确保了从硬件实现到软件应用的全栈协同。

1. 协议架构

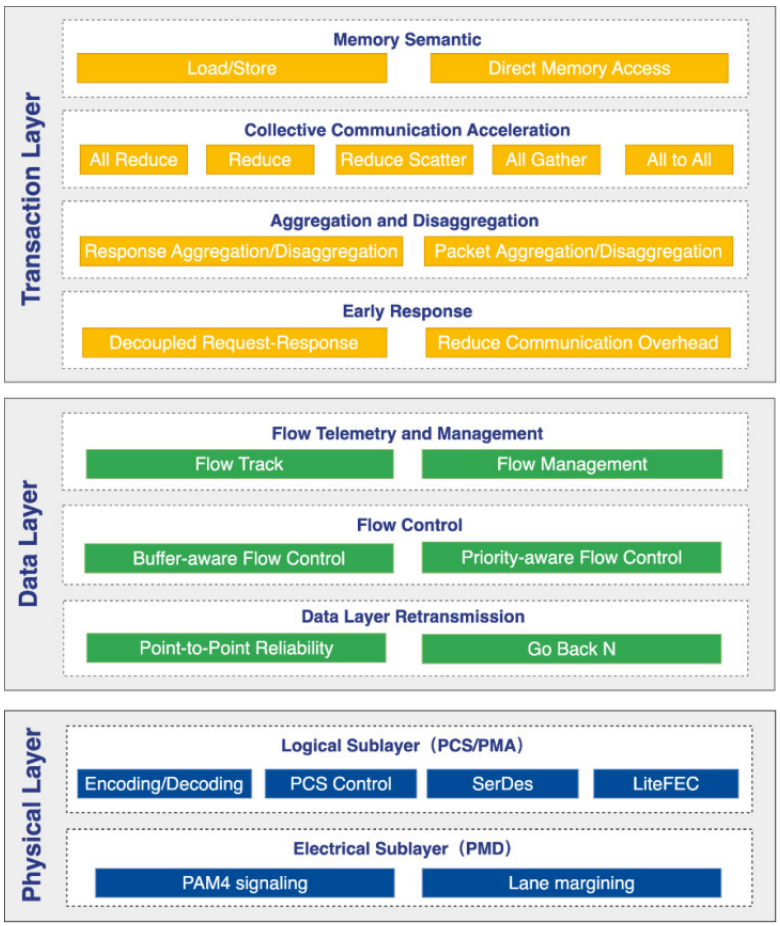


图126 OISA 三层架构及核心功能划分

OISA 协议体系采用清晰的三层分层架构设计，分别为事务层（TL / Transaction Layer）、数据层（DL / Data Layer）和物理层（PL / Physical Layer），各层级职责明确、协同高效。

事务层位于协议栈顶层，主要负责处理内存访问请求与集合通信操作，支持多模内存访问模式（ODLS 和 ODMA），并实现报文聚合分解、即时反馈及硬件集合通信加速功能。该层直接面向计算任务需求，提供低延迟、高吞吐的访存能力，是实现 GPU 间高效协同的核心支撑。通过统一地址空间管理，事务层使系统内任意计算单元可像访问本地资源一样操作远程显存或系统内存，极大提升了编程便捷性与运行效率。

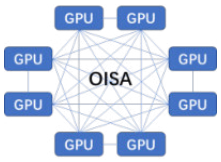
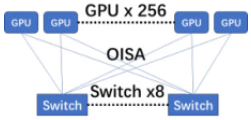
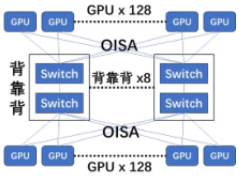
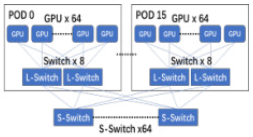
数据层作为承上启下的关键中间层，专注于保障数据在 Scale-UP 域内的可靠、有序、无损传输，具备流量感知、流量控制和数据层重传（DLR）等核心机制。它通过随路感知技术实时收集路径状态信息，结合 BFC/PFC 双模流控策略预防拥塞，并利用 GBN 滑动窗口机制实现快速点对点重传，确保高带宽环境下的传输稳定性。

物理层则基于 IEEE 802.3 标准演进，兼容多种 SerDes 速率与传输介质，支持从 50Gb/s 到 1.6Tb/s 的端口聚合能力，采用 PAM4/NRZ 编码与 RS 前向纠错技术，为上层提供高速、低误码率的物理信号传输通道，支撑 OISA 全向智感互联的高性能需求。

2. 互联拓扑

OISA 协议在设计之初就充分考虑了不同规模和应用场景下的互联需求，通过支持多种灵活的互联拓扑结构，为构建高效、可扩展的 GPU 集群提供了坚实的技术基础。OISA 技术体系支持常见的四种主要互联拓扑类型，涵盖了从简单的小规模直联到复杂的大规模多层互联等多种场景，确保能够满足不同规模集群的部署需求。

表15 OISA 支持四类联接拓扑及主要能力比较

	直接互联拓扑	单级互联拓扑	背靠背互联拓扑	叶脊互联拓扑
拓扑形式				
联接方式	GPU间直连	单级网络	单级网络+背靠背	叶-脊两级网络
GPU规模	8	256	256	1024
交换芯片	无	8	16	192/L-128+S-64
端口配置	无	固定端口映射	机柜间通过背靠背链路连接	Pod结构，支持多路径转发
卡机联接	无	多轨	多轨	多轨
有效带宽	理论高，实际聚合带宽有限	带宽确定，聚合带宽随芯片数量线性增长	带宽确定，但背靠背带来跨柜瓶颈	带宽最高，有效带宽受LB算法影响
延迟情况	最低：单跳直达	较低：固定1-2跳，路径确定	较低：多一跳背靠背链路	较高：至少2跳，路径可变
容错能力	无冗余，任意链路故障导致失效	冗余度低，依赖交换芯片可靠性	可靠性提升，机柜级链路冗余	冗余度高，支持多路径，容错性最佳
扩展能力	无法扩展	扩展成本低（按需增加芯片/交换机）	扩展成本高（需成对增机柜）	扩展成本可控（按Pod扩展）

	直接互联拓扑	单级互联拓扑	背靠背互联拓扑	叶脊互联拓扑
能效情况	能效比优	能效比良好	能效比中等	能效比一般
存在优势	成本低、功耗低、布线简单	扩展性好、确定性转发、避免乱序	适配机房环境、突破单机柜限制	支持千卡规模集群、可扩展性强
存在限制	扩展性差	需要固定端口映射关系	需双倍交换芯片数量增加了成本	数据转发跳数多，更依赖负载均衡

报文定义与接口概况

中国移动联合业界伙伴共同研发 OISA 协议并非对现有网络协议的修补，而是面向 AI 智算业务这一特定领域开展重新设计，建立以内存为中心（Memory-Centric）的技术体系。通过 OISA，超节点系统将分散的 GPU 显存与内存系统映射为统一寻址的全局共享空间，使跨节点协同回归为标准的内存读写语义，将实现百纳秒级别的极低延迟。这一逻辑最终通过 OISA 统一报文格式进行了落地。

1. 报文类型与三类访存模式

图127 OISA 报文结构

OISA 协议通过标准化的报文结构设计，实现了高效、可靠的数据传输。其报文由事务层头、数据载荷和 CRC 字段三部分组成。事务层头采用强制性头部结构，通过支持灵活的可选字段配置，确保了在不同通信场景下都能提供最优的传输效率和可靠性保障。数据载荷为条件可选字段，由事务层头中的特定字段定义是否携带。CRC 字段为强制性字段，用于整个报文完整性校验。

OISA 在协议报文格式上的设计体现了其核心特质：既要保证协议处理的低开销，又兼具适应多样化应用需求的能力。

为了适应 AI 智算场景中的多种应用需求，OISA 基于协议报文特质定义了三种各自独立的报文访存模式。GPU 与交换芯片可根据报文格式类型、事务类型、系统配置等信息，识别并区分不同类型的通信事务，进行相应转发或依策略处理。

这三种模式包括 1）极致访存模式；2）智感访存模式；3）集合通信加速模式。

1）极致访存模式：该模式以最小跳转时延和最高载荷效率为目标，在交换拓扑架构中，交换芯片通过精简处理实现线速、低延迟转发，主要面向延迟敏感型内存语义通信事务。该模式下 OISA 报文仅包含有效转发所需必要字段，此时禁用智能感知及集合通信加速模式相关功能。

在该模式下报文基础结构由 20 字节强制必选字段构成，包括 16 字节的统一报文头与 4 字节的 CRC 校验，为所有 OISA 报文必须遵循的最小格式要求。

- **强制必选字段**
  - 统一报文头

    为网络交换设备和端侧设备提供报文分类、路径决策及事务跟踪的核心信息。其定义的字段支持（但不限于）以下能力：

    通过格式与类型字段解析事务语义；通过源/目标 ID 实现路径寻址；通过虚拟通道（Virtual Channel, VC）和标签 ID（Tag ID）提供服务质量保障与事务匹配等。
  - 全局 CRC 校验

    用于保障报文在物理介质中的可靠传输。源端在将报文传输至物理层前，需在数据层对完整内容进行校验计算，生成 32 位 CRC 值并填入此字段；目的端通过校验该值，判定报文在传输过程中是否发生比特错误，从而确保端到端传输的完整性与正确性。
- **条件可选字段**

  条件可选字段根据事务类型动态包含，由内存地址（Memory Address）和变长数据载荷（Data Payload）组成。其存在与否及长度由 16 字节统一报文头中的事务类型（Packet Type）决定，并通过 Format 字段具体指示是否携带：
  - 内存地址

    该字段紧跟 16 字节统一报文头之后，为 64 位精确地址，仅在需要内存寻址的事务中出现，如内存读请求（Memory Read Request）、内存写请求（Memory Write Request）等，用于指定目标内存位置。变长数据载荷

    用于承载事务相关的事务数据，例如内存写请求（Memory Write Request）中待写入的数据、内存读响应（Memory Read Response）中返回的目标数据等。Data Payload 的长度由统一报文头的 Length 字段明确描述；当事务无需携带报文数据时，Length 字段设置为 0。部分定义字段支持能力如下。
  - 变长数据载荷

    用于承载事务相关的事务数据，例如内存写请求（Memory Write Request）中待写入的数据、内存读响应（Memory Read Response）中返回的目标数据等。Data Payload 的长度由统一报文头的 Length 字段明确描述；当事务无需携带报文数据时，Length 字段设置为 0。Format 用于指示该数据载荷字段是否携带。

    部分定义字段支持能力如下：

    - Virtual Channel（**虚拟通道**）字段支持 8 个虚拟通道，为不同类型的事务提供精细化的服务质量管控与流量隔离能力。Packet Type（**报文类型**）字段用于描述事务层报文操作语义，指导目的端解析、执行与应答流程。
    - Entropy（**熵值**）字段为多路径互联环境提供高效的负载均衡辅助机制，作为交换芯片等价多路径算法的高质量随机化输入，解决哈希碰撞问题。目前支持源端 GPU 填充方式。

2）智感访存模式：该模式为传输性能优化提供数据支撑，通过 GPU 与交换芯片协同实现端口缓存监控、性能检测及元数据提取。通过在报文传输中嵌入随路感知信息，为系统提供端网协同的流量感知能力。

GPU 或交互芯片的 OISA 端口通过识别 OISA 报文中的 Sensing Protocol Type 字段判断当前传输模式为智能访存模式，随即开启随路智能感知功能，并探测收集相关信息。每当报文到达 OISA 端口或从 OISA 端口发出时，相应的端口设备会根据感知标签将自身的关健状态信息通过比较替换的方式，实时更新到报文的感知标签中。

智感模式下的报文与极致访存模式报文相比，主要在于协议类型字段和新增的感知标签字段。OISA 定义了一种简单、低开销的感知标签字段。感知标签内包含由跳数（Hop Count）和设备端口号（Device Port ID）组成的设备位置信息，以及感知信息类型（Sensing Type）和状态程度值（Degree Value）。在报文中部分定义字段如下：

- **Hop Count（跳数）**字段基于数据发送路径，表示感知标签内的信息对应的设备位置。通过该字段信息，结合拓扑信息可定位到传输路径上的具体设备。
- **Device Port ID（设备端口号）**表示对应的设备上的端口号。根据感知标签中的跳数、设备端口号，结合拓扑以及数据报文中的 GPU ID、VC、Entropy 等字段，可以定位感知信息对应的具体位置。
- **Sensing Type（感知类型）**字段表示不同的感知内容。实现源端感知探测不同类型的网络拥塞信号，源端会选择其希望从网络中获取哪种类型的拥塞信息并告知数据层，数据层在封装报文时将其填充到感知标签内的本字段中。目前已支持最小可用带宽比、最大单跳时延、最大已用队列深度、最小队列释放速率四类感知能力。
- **Degree Value（状态程度值）**字段将复杂的网络指标通过数据（8 档－128 个梯度）进行非均匀量化，在较低报文开销下实现状态的实时更新与快速比对，为精准的流量调度决策提供核心支撑。状态值采用了非均匀量化方式进行配置，越偏向较差状态时，可使用更细粒度进行量化标注。

智感访存模式中，OISA 定义的**反馈报文**是网络状态信息回传的核心机制。当携带感知标签的探测报文到达目的端 GPU 后，GPU 会解析其中的路径状态信息，并在状态劣化至预设阈值时，触发生成一个标准反馈报文。反馈报文有单独的协议类型，其数据结构中包含了反馈信息对应的位置和内容，通过设备类型、编号、端口及 VC 通道号精确定位问题路径位置，并将量化后的 Degree Value（状态程度值）回传给源端 GPU，从而实现网络状态的闭环感知与优化，同时通过阈值触发机制有效避免了频繁反馈对链路带宽的占用。反馈报文主要定义字段如下：

- Valid (Vld)字段：反馈信息中的“有效位”，有效为“1”，无效为“0”。
- Device Type (DT)字段：反馈信息中的“设备类型”，“0”表示 GPU，“1”表示交换芯片。
- Device ID Valid (DID Vld)字段：反馈信息中“设备 ID 有效”，表示其后面的 Device ID 字段是否有效，“0”表示无效，“1”表示有效。在目的端 GPU 可以直接解析定位出具体的设备位置和编号时，会直接反馈对应的设备 ID 信息，此时该字段标识有效；当目的端 GPU 无法根据所掌握的信息准确定位出设备具体编号时，则不会反馈有效的设备 ID，此时该字段标识无效。
- Device ID 字段：反馈信息中的“设备 ID”，最多可表示 1024 个同类型设备；根据此信息可以定位到具体的设备位置。在二层组网下，目的端 GPU 可能无法定位部分交换设备的具体位置，此时该字段填零。
- Entropy Valid (EVld)字段：反馈信息中的“熵有效”，“1”表示段有效，“0”表示无效。
- Entropy 字段：反馈信息中的“熵”，与 OISA 报文协议头中熵字段长度一致。在二层组网下，源端 GPU 可根据此信息可以定位到一层交换设备的具体位置；该字段可通过解析数据包中的熵字段得到。

- Device Port ID 字段：反馈信息中的“设备端口 ID”，最多可表示一个设备上的 512 个端口；根据此信息可以定位到具体设备的具体端口位置；该字段可通过解析感知数据包的感知标签中设备端口号字段获得。
- VC 字段：反馈信息中的“虚拟通道”，对应一个端口的 8 个虚拟通道；根据此信息可以进一步定位具体端口的具体通道；该字段可通过解析感知标签中感知数据包的 VC 字段获得。当感知信息类型的粒度为端口时，该字段无效。
- Sensing Type 字段：智感反馈信息中的“感知类型”，该字段可通过解析感知数据包的感知标签中感知类型字段获得。
- Degree Value 字段：反馈信息中的“状态程度值”，该字段可通过解析感知数据包的感知标签中状态程度值字段获得。
- Reserved 字段：反馈信息中的“保留”字段，为以后扩展使用。

3）集合通信加速模式（CCA Mode）：旨在通过将传统上完全由 GPU 执行的集合通信操作（如 Reduce、AllGather、Broadcast 等）卸载到交换芯片（Switch）上执行，从而显著降低集合通信操作的整体延迟、节约 GPU 计算资源并提升交换芯片的利用效率。该模式下，交换芯片内部集成专用计算单元，能够对流经的报文执行聚合、归约等算术或逻辑运算。CCA 模式通过在交换芯片内部完成部分计算任务，避免了大量数据在 GPU 间循环传递，减少了网络传输量。在报文中部分定义字段如下：

- **强制性必选字段**

  CCA 模式的转发报文基于极致访存模式基础之上进行了扩展，此模式下必须用到 32 字节强制性必选字段：包含 28 字节的报文头和 4 字节的循环冗余校验 CCA 报文。

  Memory Address（内存地址）字段：与极致访存模式不同的是，集合通信加速过程交换芯片需要确认来自不同 GPU 的数据中哪些数据需要做同一批归约操作，以及确认组播需要发送到哪些地址空间中。这需要依靠 Memory Address 指定数据对应内存的映射地址确认这些数据之间是否对齐，保证交换芯片计算和组播操作准确，所以在 CCA 模式下 Memory Address 为必选字段。

  CCA 报文字段：在 CCA 模式下，通过 Format 指示该报文是否开启 CCA 功能，OISA 头部字段需要额外携带的 CCA 扩展头（CCA Group ID、CCA RSV、CCA Data Type、CCA OP Type）。

- **条件可选字段**

  条件可选字段为数据负载 Data Payload 字段，在需要传输数据的操作中出现。

  在 CCA 模式下，OISA 报文头部分字段基于极致访存模式增加专用说明，并增加额外 CCA 报文字段，所涉及主要报文字段定义如下：

  - Format 字段：在 CCA 模式下，Format 表示集合通信加速模式指示位，该指示位用于标识一个报文是否参与集合通信加速操作。
  - Packet Type 字段：集合通信加速中的常用通信事务类型，共七类如下：
    - GPU 归约请求（GPU Reduce Request, GR-Req）
    - Switch 归约请求（Switch Reduce Request, SR-Req）
    - Switch 归约响应（Switch Reduce Response, SR-Resp）
    - GPU 归约响应（GPU Reduce Response, GR-Resp）

- GPU 投递型组播请求（GPU Posted Multicast Request, GPM-Req）
- 非投递型组播操作（Non-Posted Multicast Request, NPM-Req）
- 非投递型组播响应（Non-Posted Multicast Response, NPM-Resp）

- Length 字段：在 CCA 模式下，Length 字段功能与极致访存模式一致，Length 字段的准确释义取决于其所在报文的 Packet Type。
- Tag ID 字段：在 CCA 模式下，Tag ID 与极致访存模式功能一致作为一种标识。开启 CCA 后需要交换芯片为每一个 GR-Req 的报文分配新的 Tag ID，从而确保后续每次归约的数据都能对齐。
- Memory Address 字段：在 CCA 模式下，Memory Address 指示内存映射的地址信息，即数据所在的内存地址映射为虚拟内存地址。交换芯片根据 Memory Address 和 Tag ID 识别不同 GPU 上的待归约数据是否为同一批，以及通过 Memory Address 和 CCA Group ID 确认组播的对象。交换芯片不编辑修改此字段，只负责识别和转发。
- CCA Group ID 字段：用于标识归约组序号，即为参与同一个跨 GPU 归约操作或组播操作的一组 GPU 分配一个共同的、唯一的标识符（ID）。CCA Group ID 通常需要根据并行策略由 GPU 进行配置，交换芯片需要依据 CCA Group ID 确认参与归约操作对象以及组播对象。
- CCA RSV 字段：为集合通信加速模式下自定义部分，规范允许用户利用 CCA RSV 编码空间，为满足特定体系结构需求而进行自定义的操作类型扩展，以此辅助和优化集合通信加速功能。
- CCA Data Type 字段：表示集合通信加速的数据类型。GPU 集合通信中，数据类型是指参与计算与通信的数据格式。不同 GPU 可支持多种不同的数据类型，集合通信加速模式下可支持数据类型常包括：
  - 浮点数类型：Float32、Float16、Float8、Float4。
  - 整数类型：支持 INT32 等标准整数类型，具体支持范围取决于 GPU 架构。
  - 特殊类型：Brain Float16：专为神经网络设计的半精度浮点数格式。

  所有参与同一集合通信加速任务下的 GPU 和 Switch 必须使用相同的数据类型，否则可能导致通信失败或结果错误。
- CCA OP Type 字段：表示集合通信加速中归约操作类型，要求支持的归约操作类型包括：SUM、MAX、MIN。可按需支持 AND、OR、XOR、BOXR、PROD 等计算操作。

2. 通信过程与比特序定义及操作

支持 OISA 的设备在通信过程中，源端设备的协议引擎将上层数据封装成 OISA 报文并发送，经过物理层编码传输至交换芯片或目的端设备，目的端设备接收后解析报文并处理，整个过程通过 T(X)和 R(X)接口实现全双工通信。

比特序定义了在协议报文封装与解析过程中，每个字节内部各个二进制位的传输与解读顺序。OISA 明确规定采用最低有效位（Least Significant Bit）优先的传输机制。

3. 接口定义与概况

事务层接口定义与功能：通过 OISA 协议实例化单元（OPE）作为关键桥接角色，负责将芯片内部总线事务与 OISA 协议在外部物理链路上定义的报文格式进行双向转换。该接口包括了 OISA“上行”的主机接口负责与 GPU 核心或交换矩阵等芯片内部功能模块交互；以及 OISA“下行”的链路接口负责与物理层收发器（PHY/SerDes）连接，处理串行化数据的收发。OISA 主机接口的通信由一组逻辑上的事务原语构成，而非简单的物理信号集合。协议定义了六类核心事务原语，包括：

- 发起写数据请求（WriteFull）
- 写数据（Write）
- 接收写操作完成状态（WrRsp）
- 发起数据读取（Read）
- 接收读操作返回数据（RdRsp）
- 设置内存屏障（Memory Fence）

物理层接口定义与功能：对 OISA 协议实例化单元（OPE）的"下行"链路接口进行规范性定义。该接口负责 OPE 与物理层收发器之间的信号交互，是实现 OISA 报文在物理介质（如光纤或铜缆）上传输的逻辑边界。其核心功能是处理待发送报文的并行到串行转换，以及目的端串行比特流的串行到并行转换。OISA 对物理层接口设计上仅定义接口信号的功能、方向和交互逻辑，并不限制具体的位宽或通道数量。

OISA 核心功能与关键技术

OISA 事务层核心功能与关键技术

OISA 协议在事务层定义了四项核心能力，旨在解决高性能计算与 AI 训练中数据传输的延迟、带宽和处理器利用率等核心瓶颈问题，支撑高效、灵活的互联通信。包括：

- **多模内存访问：**提供两种数据读写方式。一种是细粒度访问，由 GPU 计算核心直接发起，适合处理零散数据；另一种是批量传输，由专用 DMA 引擎负责，适合大规模数据块的搬运。系统可根据任务特性自动选择最佳路径。
- **集合通信加速：**将分布式计算中常用的“归约”等集合操作，从 GPU 卸载到网络交换芯片中执行。报文数据在通过网络时即被实时聚合计算，无需全部返回源端再处理，从而大幅降低通信延迟和 GPU 的计算负载。
- **报文聚合与分解：**为了提升小数据包传输的效率，发送端会将多个发往同一目的地的小报文打包成一个大数据包统一发送，以减少重复的协议开销。接收端在收到后，会将其拆解还原成原始的多个独立报文。
- 即时反馈机制：改变了传统“发出请求后必须等待最终答复”的模式。当 GPU 将一个请求可靠地发送给**网络的中间节点**（如交换机）后，即可立即获得“已发出”的确认，并继续后续工作，无需等待请求最终到达目标 GPU 并返回结果。这极大地减少了处理器的空闲等待时间。

1. 多模内存访问

多模内存访问技术旨在解决高性能计算（HPC）与人工智能（AI）训练集群中，计算节点间（尤其是 GPU 间）高效数据通信的根本需求。所有场景都围绕一个核心：让数据在分布式内存间快速、透明地移动，使大规模并行计算如同在单一大内存机器上运行。

目前 OISA 的 2.0 协议事务层在架构上支持一个多模内存访问模型，为不同粒度的内存操作提供最优化的执行路径。此模型融合了用于细粒度访问的指令驱动 Load / Store 语义（ODLS），以及用于粗粒度数据搬运的直接内存访问（DMA）模式（**ODMA**）。同时，为保障在并行计算环境中内存操作的顺序性与可见性（内存一致性），OISA 协议在事务层定义了一套标准的内存栅栏（**Memory Fence**）操作，该操作是一种强制性的同步原语，其在并发执行数据报文中建立一个严格的排序点。

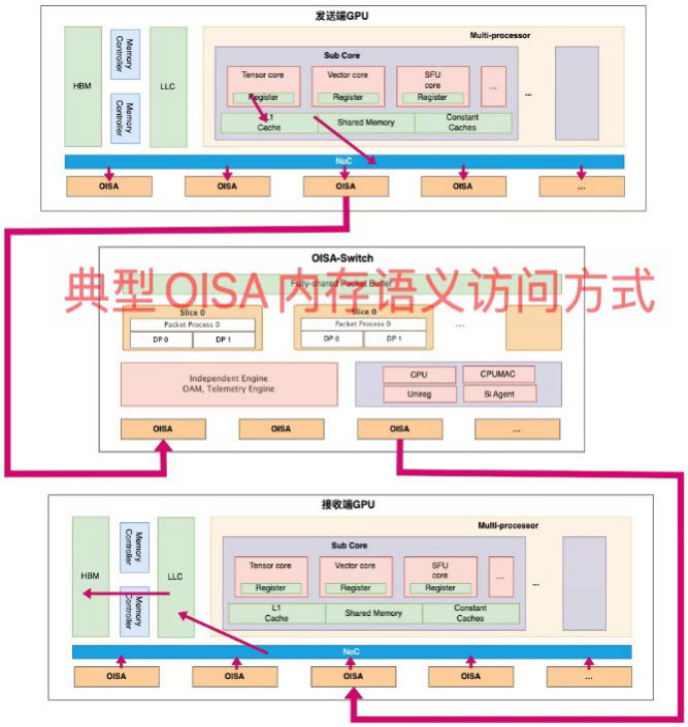


图128 OISA 共享内存访问模型

ODLS（指令直驱访问 Load/Store）：通过向计算核心提供统一虚拟地址空间，屏蔽底层内存的拓扑差异，在 GPU 计算核心直接发起的细粒度远程内存访问过程中，实现透明化的低延迟数据交互。协议对此模式处理流程如下：

- 在 OISA 系统中，源端 GPU 上的计算核心可用其原生 Load /Store /Atomic 指令直接访问互联域内、已映射到其虚拟地址空间的目的端 GPU 显存，其内存语义与访问本地全局内存一致。
- 协议要求，当使用虚拟内存地址情况下，源端 GPU 的内存管理单元（MMU）负责解析此虚拟地址。当 MMU 通过查询页表（Page Table）判定该地址所对应的物理地址位于一个远程目的 GPU 上时，MMU 将此内存访问请求重定向至 OISA 协议引擎（OPE）进行处理。

- 源端 GPU 的 OPE（OISA 协议引擎）将上述指令级内存访问自动、透明地转换为相应的 OISA 事务层报文（如 Memory Read/Write Request、Atomic Operation Request），后续目的端 GPU 收到该报文后执行指定的内存操作，按协议完成响应/完成包处理。
- OISA 事务层报报文被递交到物理层后，通过物理介质传输至交换芯片。交换芯片的入口逻辑解析报文的 OISA 协议头部，以获取转发所需信息，交换核心使用目的 ID 作为其内部转发表的索引，以确定唯一的出端口。报文从该指定的出端口转发至连接目的端 GPU 的物理链路上，传输至目的端 GPU 的 OISA 接口。
- 目的端 GPU 的 OPE 对接收到的报文进行 CRC 校验、解包，提取出 Memory Address、Length 以及 Data Payload 等关键信息，并将解包后的写请求（Store）及数据传输至 NoC 总线。NoC 总线将接收到的请求和数据传输至内存控制器，内存控制器将接收到的数据，写入由 Memory Address 指定的、位于其本地物理显存中的地址空间。完成整个远程存储操作。

ODLS 技术需要考虑如何让一次远程内存访问在微秒级内完成，并且对程序员而言，其代码看起来就像访问本地变量一样简单。基于该技术理念，OISA 需要实现：

1）统一地址空间（UVA）：系统软件为所有参与 GPU 的显存建立一个全局统一的虚拟地址空间。GPU 计算核心发出的任何 Load、Store、Atomic 指令中的地址，都在此空间内。

2）硬件地址翻译与协议卸载：GPU 内的 OPE（OISA 协议引擎）会拦截这些指令。它首先判断目标地址是本地还是远程。若判断为本地，直接访问本地显存控制器；若为远程，硬件自动将访存指令封装成 OISA 网络报文，通过网络发送至目标节点，并由目标节点的 OISA 硬件执行实际的读写操作，再将结果或应答返回。整个过程完全由硬件完成，计算核心无需知晓网络细节。

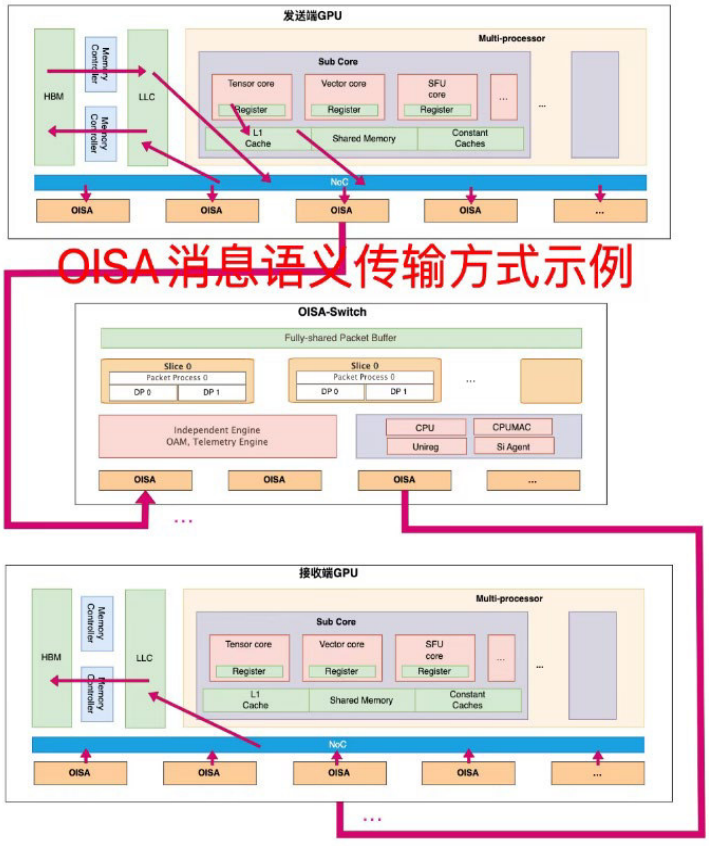


图129 OISA 消息传递的模型

ODMA（直接内存访问）OISA 规范需要 GPU 应包括包 OISA 直接内存访问（OISA Direct Memory Address, ODMA）引擎。该引擎的核心职责是独立于计算核心，在源端 GPU 准备好数据后，将一块连续的内存区域通过互联端口传送至目的端 GPU 的指定内存地址主要面向需要高带宽和异步、粗粒度数据搬运的场景应用。协议对此模式处理流程如下：

- ODMA 传输流程的启动，以待传输的数据块已由源端 GPU 的计算核心处理完毕并置于其本地内存（如 L2 Cache 或 HBM）中为先决条件。
- 源端 GPU 的 OISA DMA 引擎在被上层软件编程触发后，启动数据传输操作。ODMA 引擎负责从源端 GPU 的本地 HBM 中读取指定的数据块，并通过内部 NoC 总线将其以流式方式传送至 OISA 协议引擎 (OPE)。OPE 在接收到数据流后，将其被封装成一个独立的 OISA 写请求 (Write Request) 报文，并包含正确的 TLP 头部信息及 CRC 校验值。
- 该封装完成的写请求报文经由物理介质传输至交换芯片。交换芯片对每一个报文独立地执行解析与转发。交换核心根据每个报文头部的目的 ID 通过其内部转发表进行查询，并将报文从指向目的端 GPU 的对应端口转发出去。报文从指定端口转发至目的端 GPU 的 OISA 接口。
- 目的端 GPU 的 OPE 对接收到的每个报文进行 CRC 校验和解包，提取出的数据载荷 (Data Payload) 及目标地址，并将解包后的数据传输至目的端 GPU 的 NoC 总线。NoC 总线将数据递交给内存控制器（通常会途径 L2 Cache 进行缓冲）。内存控制器将接收到的数据块写入由报文地址指定的、位于其本地内存中的地址空间。随着写入最后一个报文数据，整个批量数据块在目的端 GPU 完成重组。

ODMA 技术需要考虑如何在不干扰 GPU 核心计算的前提下，以接近线速的带宽搬移海量数据，并简化编程。基于这种“将大块数据搬移任务卸载给专用引擎”的技术理念， OISA 建议实现：

- **专用 DMA 引擎：**OISA 建议 GPU 内集成独立的、用于数据搬移的硬件处理单元。该引擎独立于计算核心，在源端 GPU 准备好数据后，将一块连续的内存区域通过互联端口传送至目的端 GPU 的指定内存地址。
- **专用 DMA 联接：**ODMA 通过 DMA 引擎在 GPU 之间构建了一条面向批量数据的高效可靠传输通路。它实现了从源端内存经网络到目标端内存的完整直接访问路径，数据绕过 CPU 及其系统内存进行直达搬移，从而在保证可靠性的同时达成接近线速的吞吐性能。

通过在协议中同时规范这两种访问模式，OISA 架构确保了无论是延迟敏感的精细化操作，还是带宽敏感的批量数据搬运，都能在统一的互联链路上找到最高效的执行路径。

OISA 协议定义了内存栅栏（Memory Fence），旨在解决 Scale-up 架构中跨 GPU 内存访问的一致性挑战：操作顺序的不可预测性与结果可见性的延迟。内存栅栏是一种同步原语，它强制要求：在栅栏指令之后的内存操作执行之前，所有在其之前的操作必须完成，并确保其结果对其他处理器可见全局。

Fence 作为强制性的同步原语，为程序员提供了明确的“顺序点”与“可见性点”。协议对 Fence 操作的行为提出如下强制性要求：由任意远端 GPU 发起的 Fence 操作，能够使其在该 Fence 之前提交的所有内存写操作，在 Scale-up 域的范围内完成并且其结果对其它指定范围内的设备可见之后，该请求方的后续任何内存操作方可开始执行。

OISA 协议支持的 GPU 内存栅栏设计具备多级作用域，包括用于较小范围同步的局部（local）栅栏、用于更大范围同步的全局（global）栅栏，以及面向更大规模同步需求的集群（cluster）栅栏。这种分级机制允许应用程序根据实际交互的范围和粒度来选择合适的同步强度，从而避免不必要的性能开销。

以下三种作用范围的栅栏指令：

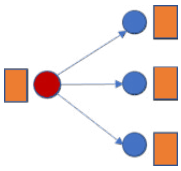
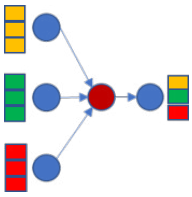
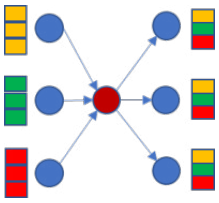
- **局部栅栏（Local Fence）：**此栅栏的作用域被规定用于在一个有限的、紧密耦合的域内进行同步。其典型应用场景包括同步同一 GPU 芯片内不同线程块之间的操作。其具体作用范围的边界由具体实现定义。
- **全局栅栏（Global Fence）：**此栅栏的作用域被规定用于在更大范围内（例如，一个包含多个 GPU 芯片的计算节点）确保内存操作的顺序和可见性。
- **集群栅栏（cluster Fence）：**此栅栏的作用域按内存地址范围限定内存栅栏。Fence 操作可被一个或多个内存地址范围所限定，所提供的顺序性与可见性保证将仅适用于那些目标地址落在指定范围内的内存操作。此机制使得协议无需强制进行全地址空间的同步，从而在一定程度上节省互联带宽。在 AI 训练参数更新及梯度计算过程中可起到关键优化作用。

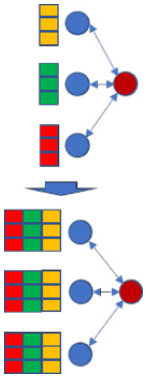
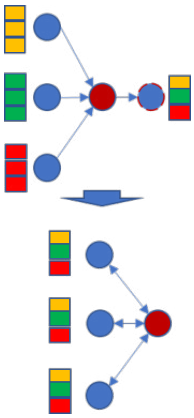
此分级机制旨在允许应用程序根据其实际的同步粒度，灵活选择开销最小的栅栏类型，从而避免因使用非必要的大范围栅栏而导致的性能损失。通过分级机制具备的可扩展性，以高效支持高带宽域内 Scale-UP 全局同步需求。

2. 集合通信加速

集合通信加速技术主要面向大规模分布式 AI 训练和高性能计算（HPC）场景中的集合通信操作，包括 Reduce、Broadcast、ReduceScatter、AllGather、AllReduce 等核心通信原语。这些操作在大模型训练中频繁出现，是影响整体训练效率的关键瓶颈。

表16 训练过程中涉及的各类通信情况

通信	联接	过程	特点	AI 训练场景
Broadcast		一个节点将相同的数据发送给所有其他节点。	1对N通信，根节点出口带宽压力高，带宽受限操作。网络拓扑和根节点位置对性能影响高。	初始状态同步。在训练开始前，将全局一致的参数、数据从主节点分发到所有工作节点。
Reduce		所有节点都向一个节点发送数据，该节点对所有数据进行运算（如求和），并保留结果。	N对1通信，对目标节点入口带宽和算力有要求。数据被节点汇总后，其他节点无法直接获取。	中心化聚合：将各节点评估指标、损失值汇总回主节点，记录日志或监控，或在小规模参架构中收集梯度。
Reduce Scatter		1. 规约阶段：每个节点的完整数据（如梯度张量）被切分成N块。所有节点对第i块数据规约（如求和）。 2. 散射阶段：规约后第i块结果被唯一发送给第i个节点。最终，每个节点只持有一个片段。	将全量规约的计算和通信压力分散到了所有节点。每个节点既是数据的发送者也是接收者，但只处理全量数据的一部分。能有效利用所有节点的聚合带宽和计算力，通常比简单的Reduce更高效。	分散式规约的第一步：通常与AllGather配对使用，以实现高效的AllReduce。在一些模型并行策略中，直接用于在不同设备间分配及规约部分激活值或梯度。

通信	联接	过程	特点	AI 训练场景
All Gather		每个节点都向所有其他节点发送自己持有的唯一数据块。最终，所有节点都收集到来自所有其他节点的数据块，拼凑成一份完整的数据集。	每个节点的出口数据量较小（一个数据块），但需要从所有其他节点接收数据。通信量随节点数增加而线性增长。是典型的多对多、全连接通信。	数据整合与同步：作为高效AllReduce的第二阶段，用于将ReduceScatter后分散在各节点的结果片段收集成全量结果。也用于整合不同节点上计算出的不同数据分区。
All Reduce		所有节点都提供输入数据，并最终获得完全相同的规约结果（如梯度求和）。 这是分布式训练中最核心、最频繁的通信操作。	让所有节点结果一致。其实现是由更基本的原语组合而成（ReduceScatter+ AllGather）。性能优化核心在于平衡所有节点计算和通信负载，最小化瓶颈。	全局梯度同步：在数据并行训练中，每个训练步（Step）结束后，将所有工作节点计算出的局部梯度进行运算，并将更新后的全局梯度同步回所有节点，以保证模型参数的一致性。

OISA 的集合通信加速功能由硬件层（GPU、Switch）与软件层（OISA 管理器、集合通信库）共同定义了技术架构。

- **CCA 相关组件**

  软件层组件包括 OISA 管理器、集合通信库（Collective Communication Library, CCL）。在集合通信加速中的主要功能为：

  - OISA 管理器：OISA 管理器包含 GPU 管理器和 Switch 管理器，实现配置管理、拓扑发现、状态监控、硬件维护等功能。
  - 集合通信库：协同 GPU 与 Switch，响应应用框架的调用，启动特定的内核函数实现集合通信算子调用和集合通信加速功能调用。
  - 硬件层组件包括 GPU、Switch。在集合通信加速中的主要功能为：
  - GPU：提供与之配套的 OISA 管理器（GPU 管理器）、集合通信库组件，完成除 Switch 可支持的归约计算之外的计算类型，收集归约计算结果并在 GPU 组内转发。
  - Switch：提供与之配套的 OISA 管理器（Switch 管理器），完成归约计算，并向 GPU 转发归约计算结果。

- **交互接口**

GPU 及其配套的 OISA GPU 管理组件、Switch 及其配套的 OISA Switch 管理组件之间依托标准 OISA 协议，不同类型事务报文以及 LLDP 报文进行通信，实现流程的协同和信息的交互。集合通信加速各组件存在以下接口：

接口	接口描述
GPU管理器--GPU	管理 GPU 及 Switch 间的拓扑关系，对GPU进行配置管理、状态监控、硬件维护等
Switch管理器--Switch	对Switch进行配置管理、状态监控、硬件维护等
GPU管理器--集合通信库	OISA GPU管理器为集合通信库维护并提供了GPU与Switch之间的拓扑信息及链路状态，使能集合通信库得以建立集合通信和集合通信加速所需要的最短路径
集合通信库--GPU	集合通信库作为GPU上运行的软件库，响应应用框架的调用，实现内核（Kernels）启动、集合通信函数和集合通信加速功能调用等操作

- **初始化配置**

初始化配置用于协同 GPU 及 Switch，使其将 GPU 和 Switch 分别配置为能够支持对本次 AI 任务进行集合通信加速的状态。初始化配置由应用框架发起，集合通信库响应，生成内存语义报文通过 GPU Header 的所有端口传递给对端 Switch 处理器完成配置，包括建立对应 GPU 组、分配 Switch 缓存等。

在集合通信加速实现过程中，交换芯片不再只是被动的数据管道，而是成为主动的协同处理器，通过在转发路径中直接执行聚合、归约等计算操作，减少数据在网络传输中的总量，主要通过下列关键技术实现：

- **交换芯片卸载计算技术：**当支持 CCA 模式的报文经过芯片时，通过集成在交换芯片内部的专用计算单元能够直接对报文载荷执行事先定义好的的归约（如求和）或其它聚合运算，计算结果更新到报文中继续转发。这直接减少了 GPU 的计算负担和需要传输的最终数据量。
- **协议扩展与 CCA 报文识别：** OISA 协议定义了集合通信加速类型访存模式（CCA Mode），通过报文头中增加用于集合通信加速（CCA）的专用字段，包含了本次操作所必需的语义信息，如归约组标识、具体操作类型（如 SUM/求和）、数据类型（如 FP16）等。交换芯片硬件具备解析这些头部信息的能力，从而能准确识别一个报文是否属于 CCA 操作，并确定应对其执行何种处理（直通、复制或调用 ALU 进行计算）。
- **软件栈与配置管理协同：**集合通信库（CCL）需要与 OISA 管理器协同工作。在通信开始前，必须通过标准协议完成一系列的初始化配置，将本次集合通信的参数（如组、操作类型）告知相关的交换芯片，使其进入准备状态。整个加速过程依赖于软硬件间预先建立的良好配置与状态同步。
- **稀疏感知加速：**MoE 通过激活少量专家来减少计算量但其动态路由机制也带来了通信不确定性，OISA 通过在报文中添加稀疏感知字段（Sparse Sens）标识出哪些专家是激活的。交换机可以识别出对应的激活专家并仅聚合这些激活专家的数据再路由回 token 分发的源 GPU，并在此过程中完成归约计算，进一步实现训练加速。

为实现集合通信加速的目的，针对不同集合通信原语，交换芯片可以实现不同的加速操作，包括：

- 对于 AllReduce、ReduceScatter 这类包含算术运算的操作，Switch 芯片可以直接执行计算密集型的归约计算（如浮点数求和），避免数据在多个 GPU 之间循环传递。
- 对于 AllGather、Broadcast 这类数据分发操作，可以利用 Switch 芯片执行高效的硬件组播（Hardware Multicast）能力，一次性将数据分发至多个目标 GPU，优化了归约组 GPU 之间的数据复制和分发。
- 对于 All-to-All 这类复杂的数据交换模式，可以利用 Switch 芯片的高带宽、无阻塞交换核心进行路径优化，减少网络拥塞和传输冲突。
- 在 All-to-All 集合通信加速时，OISA 定义了动态组播请求事务，面向稀疏类模型场景(MoE)，聚焦多 GPU 局部数据动态交互，通过交换机缓存内建动态组播表实现“按需组播”，避免传统全量广播的带宽浪费。

3. 报文聚合与分解

为应对分布式应用中高频次、小报文通信带来的高开销问题，OISA 协议在事务层定义了标准的报文聚合和分解机制，支持最多 64 个报文进行聚合。该机制需要相应的硬件逻辑予以支持，并在链路上源端 GPU 进行聚合、目的端 GPU 进行分解。通过这种方式，最大限度地减少单次传输的协议头开销，提升小报文场景下的有效数据吞吐率与系统整体效率。

源端聚合要求：通过设设定一个聚合控制器将多个去往统一目的地的报文拼接成单个 OISA 事务层报文的数据载荷并记录聚合报文数量。不能聚合不同目的地的报文。

目的端分解要求：是聚合机制在目的地的逆操作，需要通过检查聚合报文标记确定报文数量，并发送至剥离控制器还原为独立的报文，报文完整还原后再进行处理。

- **报文聚合触发条件**

  OISA 协议将源端 GPU 的聚合操作定义为机会性的，旨在平衡传输效率与处理延迟。为实现这一目标，协议要求合规实现必须提供可配置的聚合上限参数，包括成员报文数量上限（MNSP）、聚合包长上限（MLAP）。

  聚合调度器在构建聚合包时，一旦满足以下任一条件即应立即发送：队列中可聚合报文数量达到 MNSP、聚合包总长度达到 MLAP，或队列中已无可聚合报文，以此避免无故延迟。

  调度器行为需遵循严格约束：若虚拟通道（VC）未启用聚合模式，则发送单个标准报文（Packing Number 为 0）；若启用聚合模式，则根据可聚合报文数量（NSP）决定——当 NSP 大于 MNSP 时，最多聚合 MNSP 个报文发送；当 NSP 在 1 到 MNSP 之间时，聚合所有报文发送；即使仅有一个可聚合报文（NSP=1），也必须将其封装为聚合包（Packing Number 为 1）发送，确保逻辑一致性。

- **报文聚合格式要求**

  OISA 协议对聚合报文格式的核心要求如下：

  首先，通过 Packing Number 字段明确标识报文状态——值为 0 表示非聚合的单事务报文，值大于等于 1 则表示聚合报文，且该数值精确等于其包含的成员报文（Sub-packets）的数量。

  其次，为确保目的端能正确解析，每个被聚合的成员报文都必须封装在一个独立的成员报文头之后，该头必须包含诸如 Tag ID 等该报文独有的标识信息。

最后，聚合策略需根据报文类型进行差异化配置：

- 包含数据载荷的报文（如读完成），其聚合单元为“报文头+数据载荷”，只需合并少量报文即可达到所需包长。
- 无数据载荷的报文（如读请求），其聚合单元仅为报文头，因此协议要求为其配置更高的数量上限（MNSP）聚合更多报文来摊销开销，提升链路利用率。

- **报文聚合配置要求**

  OISA 协议的报文聚合配置与操作的核心在于对聚合上限参数（如 MLAP 和 MNSP）的合理设定，避免因过于激进的策略（如设置过高上限或长时间等待）而牺牲事务的端到端延迟。在设定这些参数时，实现方必须综合评估以下系统级因素：

  - 交换芯片针对不同尺寸报文的线速转发能力、不同类型成员报文的典型长度特征以制定差异化策略；
  - 聚合与分解硬件逻辑本身引入的处理延迟；
  - 聚合执行逻辑必须与流量控制机制紧密协同，当一个虚拟通道（VC）配置为采用缓存感知的流量控制（BFC）时，聚合调度器构建的任何待发送聚合报文的总长度，不得超过目的端的缓存剩余空间，从而避免引入不必要的队头阻塞延迟或在非理想情况下可能导致的报文丢失。

4. 即时反馈

在高并发场景中，即时反馈机制（Early response）可以降低延迟，是优化数据传输效率的重要功能。该机制的核心是通过在 GPU 本地提前返回响应，提前释放相关操作的上下文 Cache 资源，避免计算核心因等待而产生大量停顿，减少业务的计算时间和片上存储资源的浪费，解决系统性能和扩展性的瓶颈，提高集群资源的利用率。

即时反馈机制要求：将事务通信的“逻辑完成”与“物理完成”解耦。物理完成指一个 TLP 穿越物理链路，目标 GPU 的应用层返回响应。逻辑完成指源端 GPU 的计算任务完成。该机制旨在提升系统性能和扩展性。

OISA 的即时反馈机制仅适用于投递型写请求（Posted Write），必须在 GPU 内部事务保序性得到保证的前提下，由集群管理系统统一全局启用。实现时，发送端 GPU 需在请求发出后立即本地生成响应以快速释放资源；同时，目的端 GPU 在完成实际写入后，必须丢弃其内部的完成响应，避免产生冗余网络流量。该机制通过缩短响应路径，有效提升高并发场景下的系统吞吐与计算效率。

OISA 数据层核心功能与关键技术

OISA 协议在数据层主要通过以下三种能力建立数据传输可靠性保障体系，为高性能计算和 AI 训练提供了稳定、高效的网络支撑。

- **智能流量感知：**OISA 数据层支持智能流量感知机制，通过在报文中嵌入感知标签，使报文在传输过程中能被路径上的交换机记录关键状态，实现对网络状态的实时探测。源端 GPU 可收集传输路径上的链路及设备状态信息，目的端 GPU 解析后反馈给源端，帮助其动态调整发送策略，优化网络传输效率。
- **流量控制：**数据层提供两种流量控制机制来防止接收端缓存溢出并确保无损传输：基于优先级的流量控制（PFC）允许接收方向发送方发送暂停帧，针对特定的优先级流量实施暂停，从而避免拥塞扩散；缓存感知的流量控制（BFC）是更精细的机制，接收方通过周期性地向发送方通告其可用缓存空间，使发送方能够动态调整发送窗口，实现链路的充分利用与无丢包传输。

- **数据层重传：**为确保链路级数据传输的可靠性，OISA 数据层提供了基于序列号（Sequence Number）的点对点快速重传机制。当接收端检测到报文丢失或错误时，会通过反馈机制通知发送端，触发对特定缺失报文的重传。这种设计旨在最小化重传延迟和冗余数据量，是与事务层“即时反馈”相配合、实现高效可靠传输的关键基础。

这些能力共同构建了 OISA 协议的数据传输可靠性保障体系，为高性能计算和 AI 训练提供了稳定、高效的网络支撑。

1. 智能流量感知

为确保在 AI 突发流量场景下的传输质量，并对 Scale-up 高带宽域内的互联环境进行优化，OISA 协议在数据层支持一套标准的流量感知机制。该机制的架构目标是，使源端 GPU 能够获取数据传输路径上所有交换设备及目的端 GPU 端口的状态信息，从而精确识别性能瓶颈。基于这些信息，源端 GPU 应能够主动调整其后续的数据报文发送策略，例如调整发送速率或选择新的传输路径，以实现高效、无损的数据传输。此流量感知功能在报文被配置为智能访存模式时方可启用。

OISA 协议定义了两种互为补充的流量感知操作模式，以确保 GPU 之间互联状态信息能够被及时有效地收集。

- **源端主动探测**

源端 GPU 通过向数据报文嵌入感知标签（Sensing Tag）发起带内探测。传输路径上的每个 OISA 交换芯片必须遵循“最差状态保留”原则：比较本地量化状态与报文携带值，并用更差的状态及自身位置信息更新标签。目的端 GPU 负责解析最终标签，并根据状态程度值与预设阈值的比较结果，决定是否触发反馈。

- **目的端主动告警**

作为对源端探测的补充，目的端 GPU 必须持续监控本地关键性能指标（如缓冲区占用率）。当任何指标突破其预设告警阈值时，即使未收到对应探测请求，目的端也必须立即主动生成并发送告警报文至源端。该报文需通过高优先级队列发送，并应包含基于定时器的重传机制以确保可靠性。

- **信息反馈与调整**

目的端生成反馈报文需遵守严格规则：①反馈抑制—仅当状态劣于对应类型配置阈值时才触发；②位置报告—在拓扑确定时采用直接报告模式反馈准确设备信息，在复杂拓扑导致无法唯一定位时采用间接报告模式，并设置熵有效位指示源端结合自身上下文完成最终解析；③特殊规则—对二层交换芯片的拥塞，目的端不向源端反馈。

协议不规定具体调控算法，但所有调整必须遵循基础约束，必须保障强保序数据流的顺序性。对此类数据流，禁止采用可能引发报文乱序的路径切换策略（如改变 Entropy），仅允许进行发送速率调整等不影响路径的策略。

2. 流量控制

OISA 数据层通过两种协同的流量控制机制保障高速互联中的无损传输。

第一种是缓存感知流量控制（BFC），其核心在于源端根据实时感知到的目的端虚拟队列剩余缓存空间来动态调控发送数据量，实现逐跳的精准反压，从而从根本上避免因接收端缓存溢出导致的数据丢失。

第二种是基于优先级的流量控制（PFC），它在链路拥塞时为不同业务流量提供差异化的管控。该机制为每个虚拟通道（VC）设置独立的 XON/XOFF 阈值，当接收端队列占用达到阈值时，可暂停或恢复特定优先级的

数据流，而非中断整个链路，从而确保关键业务的高优先级流量不受低优先级拥塞的影响。这两种机制相互配合，共同实现了高效且精细的链路级资源管理。

3. 数据层重传

OISA 数据层支持重传机制，当接收方（GPU 或 OISA-Switch）检测到数据层报文出现错误或丢失时重传丢失的数据帧，可实现点对点快速重传，保障数据链路之间的可靠传输。

OISA 数据层报文分为 DLR 报文（需要重传保护）和非 DLR 报文（不需要重传保护），只有 DLR 报文丢失或出错时候需要进行重传。对于 DLR 报文，OISA 数据层将在其报文前导码中按照报文发送顺序添加序列号（Sequence Number，SN），同时将报文数据缓存至数据层重传缓冲区。对于非 DLR 报文，无需在其前导码中添加报文序列号，接收端也只需对其进行 CRC 校验即可。

OISA 数据层重传基于 GBN 机制实现，发送方可连续发送多个数据帧，不需等待每个帧确认。发送端收到 NACK 或预期时间内没有收到相应的 ACK 时，将重传对应序列号后的全部报文。

数据层重传与流控是保障数据层报文无损传输的两种机制。二者相辅相成，流量控制从源头预防丢包或降低丢包的发生概率，重传则在丢包或者数据传输出错发生后弥补损失，共同保障链路层数据传输的可靠性与效率。

BFC 机制下，对于重传数据包的发送，既不需要再次判断目的端缓存空间是否足够，源端也不应再次扣除其占用的缓存空间大小。PFC 机制下，在重传期间 PFC 暂停帧只暂停新的数据报文的发送，而不影响重传报文的发送。BFC 的缺点是需要收发双方均支持该机制，否则无法实现互通。BFC 与 PFC 都是 OISA 支持的数据层流量控制机制，需在设备初始化阶段选择流量控制机制并配置相关参数。

OISA 物理层核心功能与关键技术

OISA 在物理层端口和通道上支持多种数据传输速率，满足不同性能与成本需求：

- 聚合端口速率：50Gb/s、100Gb/s、200Gb/s、400Gb/s、800Gb/s、1.6Tb/s
- SerDes 通道速率：25Gb/s、56Gb/s、112Gb/s、224Gb/s
- 支持介质类型：铜缆、背板、光纤等

这些能力共同构建了 OISA 协议的物理层基础支撑，为高速、可靠的数据传输提供了硬件保障。

OISA Gen2.0 物理层定义了完整的信号处理链以实现高速可靠互连。其核心特色在于采用了高效的 256B/257B 编码，相比传统 64B/66B 编码进一步降低了开销，并结合自同步扰码与周期性对齐标志插入机制，以抑制抖动、辅助通道同步与纠偏。在纠错方面，PCS 子层支持 RS(544,514)与轻量级 RS(272,257)两种前向纠错方案，在提供强大纠错能力的同时兼顾不同场景对延迟与复杂度的要求。

在物理媒介适配层，PMA 负责数据的串并/并串转换与时钟恢复。它支持从 NRZ 到 PAM4 的多阶调制，并采用 Gray 编码与(1+D)模 4 预编码等技术优化信号完整性。SerDes 模块集成自适应均衡器、锁相环型时钟数据恢复电路及高性能驱动器，以补偿通道损耗、抑制码间串扰并确保信号在接收端被准确采样与恢复。

物理层对外提供了广泛的高速接口能力，支持从 25.78125Gb/s 至 212.5Gb/s 的多种单通道信号速率，并全面兼容从 500BASE 到 1.6TBASE 的 IEEE 以太网接口家族。其关键性能指标要求严格：在进入 FEC 解码前，链路误码率须优于 1E-6；针对不同速率，对通道插入损耗（从 TP0 到 TP5）规定了具体门限，例如对于 112Gb/s PAM4 信号，损耗需控制在 28.5dB@28GHz 以内。

为保障在真实封装与 PCB 环境下的稳定工作，OISA 对 SerDes 的驱动能力提出了明确要求。其设计目标是在计入封装损耗并预留 2dB 余量后，芯片间（Bump-to-Bump）的总通道损耗在特定频点下仍需满足 BER 要求，例如对于 56Gb/s PAM4 信号，需能支持高达 37dB 的总损耗，体现了其对恶劣信道条件的适应性与鲁棒性。

OISA 管理机制与运行环境要求

OISA 初始化流程关键信息总结

支持 OISA 的超节点从物理上电到业务就绪采用分层递进的方式，依次建立并验证各层级平面的功能：

带外管理平面：首先通过 BMC 建立独立于操作系统的硬件远程管控通道，完成安全加固、网络配置及所有组件（计算/交换/电源托盘）的固件对齐与统一纳管。

固件/平台层：随后执行系统上电，通过 BIOS/UEFI 配置关键参数（如高位地址空间、PCIe 链路），完成硬件枚举与自检，确保 CPU、内存、GPU 等核心硬件可被正确识别且资源无冲突。

带内管理平面：在操作系统内部配置网络、时间同步并进行安全加固，构建主机级的可观测性与安全管理能力。

带内控制平面：将节点安全注册到上层集群控制器（如 Kubernetes），注册硬件资源但暂时阻断业务调度，同时安装并待命 OISA Fabric Manager。

带内数据平面：配置并验证高性能网络、存储及 GPU P2P 等核心数据传输通路，为应用通信奠定基础。

拓扑发现与建链：最后由 OISA Fabric Manager 自动探测并构建 OISA Link 物理连接拓扑，协商建立链路，并计算分发全局最优转发路径，形成高带宽、低延迟的 GPU 互连网络。

整个过程强调前置条件检查、安全合规、配置标准化与阶段性验证，确保系统最终以稳定、可控、高性能的状态交付给业务。

管理框架核心原则总结

OISA 管理框架的设计遵循三大核心原则，旨在构建一个统一、高效且适应未来发展的管理体系：

开放性：严格基于主流开放标准构建接口与数据模型。带外管理采用 gNMI 协议，数据模型与 Redfish 标准兼容，确保能够无缝集成到现有数据中心管理生态中。

互操作性：通过对管理组件间的接口、协议和数据模型进行确定性定义，保障来自不同厂商的 OISA 设备能够在统一管理平面下协同工作，实现“即插即用”，降低集成复杂度。

可扩展性：采用“集中式控制，分布式执行”的架构。集中的 OISA 管理器负责复杂决策，分布式代理负责本地执行。这种设计确保管理复杂度随集群规模呈线性增长，从而能够有效管理从单个节点到超大规模 GPU 集群的各种部署场景。

管理架构与组件总结

OISA 管理架构是个层次化、基于标准协议的分布式系统，各组件及交互定义如下：

OISA 管理器：作为逻辑集中的唯一控制平面，是整个管理域的“大脑”。其核心职责包括：进行全局拓扑发现、集中计算并分发转发表、执行租户子域管理与隔离策略、汇总全集群监控与告警信息，并协调故障处理等 RAS 流程。

管理代理：分为交换芯片代理和计算节点代理。它们是 OM 在硬件设备侧的“执行臂”，负责接收 OM 的抽象指令，并将其转换为对本地硬件的具体操作。SA 通过 gNMI/Redfish 与 OM 通信，并通过标准化的南向接口控制交换芯片。

受管硬件：主要包括 OISA 交换芯片和符合规范的 OISA GPU。它们通过代理暴露管理接口，执行具体的转发和计算任务。

该架构通过清晰的职责分离（OM 决策、代理执行）和标准的开放接口，实现了对复杂异构硬件资源的统一、自动化管理。

参考拓扑与部署模型总结

OISA 推荐两种典型的物理部署模型，以满足不同规模和应用场景的需求：

分离式架构：计算节点与交换平台在物理上分离部署于不同的机箱或机架。这种模型适用于追求极致扩展性的大规模 AI 训练集群，支持成千上万个 GPU 跨机架互连，便于计算和网络资源的独立扩容与维护，但布线和管理网络相对复杂。

集成式架构：将 GPU 计算单元与 OISA 交换芯片高度集成在同一台物理服务器或机箱内。这种模型提供了更高的密度和更优的成本效益，减少了外部线缆和占用空间，适用于规模稍小、对部署紧凑性和总拥有成本有更高要求的场景，如企业级 AI 推理或中等规模训练集群。

OISA 对交换芯片的要求

OISA 交换芯片是实现高速 GPU 智算互联的核心引擎，其设计超越传统数据交换，深度融合了网络控制、计算卸载与深度感知能力，以满足大规模 AI 集群对自动化、确定性及高可靠性的严苛要求。

在基础转发与网络自治层面，交换芯片需具备完整的 OISA 协议处理能力。这包括对数据报文及拓扑发现等控制平面报文的精准识别与解析，后者需上送本地 CPU 以支撑全局拓扑的自动构建。转发时，必须依据目的 ID 查找转发表，并强制使用报文头中的熵值字段在多路径中执行确定性的、无状态的负载均衡，确保流内报文不乱序，保障 RDMA 等高性能通信效率。同时，芯片需集成 PFC 及更先进的缓存感知流量控制（BFC），并建议与 GPU 端实现协同一致的 BFC 缓存管理框架，从根本上避免缓存死锁，实现稳定无损传输。

在性能与高级特性上，交换芯片须以高带宽 SerDes 为基础，提供无阻塞的线速转发与端到端的超低延迟。其特别强调深度感知与带内遥测能力，能够实时采集链路时延、队列深度等信息，并通过 Sensing Tag 直接写入转发中的数据报文，为全局拥塞控制与性能调优提供实时数据。此外，强制性的硬件状态监控（如光模块功率、芯片结温）及 Link Down 等事件的即时硬件中断上报，是保障超大规模集群运维可视性与高可用的关键。

面向 AI 计算的核心优化体现在硬件级集合通信加速（CCA）。交换芯片需内嵌专用引擎，能卸载并执行包括 FP8、FP4 在内的多精度归约计算，明确支持单次最多 128 节点、4KB 数据包的集合操作。为此，它需分配专用缓存与虚拟通道，并实现从超时检测、生成异常报文到上报 GPU 的完整容错链，从而将 GPU 从通信协调负担中解放出来，显著提升分布式训练效率。

OISA 对 GPU 芯片的要求

OISA 对 GPU 的核心要求，旨在将其从一个孤立的高性能计算单元，彻底转变为深度集成、可全局调度、具备网络感知能力的智算节点。其重点并非仅提升单卡算力，而是确保 GPU 能在一个庞大、统一的集群中高效、可靠、透明地协同工作。

首要重点是实现硬件级的通信与计算融合。GPU 必须内嵌完整的 OISA 协议栈，其通信行为需严格遵循规范，例如生成的报文载荷不得超过 8KB。尤为关键的是对集合通信加速（CCA）的硬件支持：GPU 需能作为成员参与多达 128 个节点的归约操作，直接生成和处理包括 FP8、FP4 在内的多种低精度数据，并与交换芯片协同完成计算卸载。这要求 GPU 具备专用的缓存管理与地址映射机制，以配合网络侧完成高效的归约计算，从而将自身从繁重的通信协调开销中解放出来。

其次，OISA 强制要求 GPU 支持全局统一内存寻址（UMA）与 I/O 一致性模型。这是实现“内存池化”和简化编程模型的基石。GPU 必须能够识别并访问集群中任何其他设备（如其他 GPU）的内存，此过程依赖于全局地址映射表，并通过 MMU/IOMMU 进行安全的地址转换与权限检查。为确保数据一致性，GPU 硬件必须原生支持针对远程地址的原子操作，并严格执行分层级的内存栅栏指令，以保障跨设备数据操作的顺序与可见性，为分布式训练提供如同单机一样的编程语义。

最终，所有这些能力的实现都依赖于深度的软硬件协同。GPU 的硬件负责高效执行底层操作，如协议封装、地址转换和原子性保证；而驱动、运行时等软件栈则负责管理系统级的资源映射、同步原语并提供简洁的 API。OISA 通过这一整套设计，使 GPU 超越了传统的“加速卡”角色，成为可被网络智能调度、具备全局内存视角、并能与交换芯片协同完成计算任务的核心智能算力单元，从而支撑起大规模 AI 集群的高效运转。

3.5 NVLink 协议介绍 [93]

NVLink 是英伟达（NVIDIA）开发的一种高速互连总线及通信协议，采用点对点结构与串行传输技术，主要用于实现 CPU 与 GPU 之间，以及多个 GPU 之间的高效互联，显著提升数据传输带宽与通信效率，广泛应用于高性能计算（HPC）、人工智能训练和数据中心等领域。NVlink 是英伟达的专有协议，没有标准化组织支持，英伟达 2025 年开始通过 NVlink Fusion 计划向部分合作伙伴授权 NVlink 技术，但其核心协议仍保持私有性质。本章节仅对 NVlink 已公开的技术特性进行介绍。

英伟达于 2014 年推出 NVLink 技术，历经六代演进，从第一代 160GB/s 带宽到第六代 3.6TB/s 带宽，NVLink 实现了超过 22 倍的性能提升。第六代 NVLink 于 2026 年 1 月随 Rubin 架构的 R200 GPU 正式发布，是当前最新的 NVLink 技术版本。同时衍生出 NVLink-C2C 等专用版本，适配不同异构计算场景。

表17 NVLink 各代技术参数对比

参数	NVLink 1.0	NVLink 2.0	NVLink 3.0	NVLink 4.0	NVLink 5.0	NVLink 6.0
发布时间	2016年	2017年	2020年	2022年	2024年	2026年(预计)
支持架构	Pascal	Volta	Ampere	Hopper	Blackwell	Rubin
SerDes速率	20Gbps	25Gbps	50Gbps	112Gbps	224Gbps	双向224Gbps

参数	NVLink 1.0	NVLink 2.0	NVLink 3.0	NVLink 4.0	NVLink 5.0	NVLink 6.0
编码方式	NRZ	NRZ	NRZ	PAM4	PAM4	PAM4
单链路双向带宽	40 GB/s	50 GB/s	50 GB/s	50 GB/s	100 GB/s	200 GB/s
每GPU最大链路数	4	6	12	18	18	18
每GPU总双向带宽	160 GB/s	300 GB/s	600 GB/s	900 GB/s	1,800 GB/s	3,600 GB/s
NVSwitch 支持	无	第一代 (32 端口)	第二代 (64 端口)	第三代 (64 端口)	第四代 (72 端口)	第五代 (72 端口)
主要特点	首次推出，支持 GPU-CPU 互联	引入 NVSwitch，支持 8GPU 全连接	速率翻倍，链路数翻倍	采用 PAM4，带宽提升 50%	速率翻倍，带宽翻倍，支持 NVLink-C2C	速率再次翻倍，带宽达 3.6TB/s

关键特性

- 超高带宽：NVLink 6.0 总双向带宽达 3600GB/s，是 PCIe 6.0 x16 带宽（256 GB/s）的 14 倍。
- 低延迟：通过专用通道和优化协议，NVLink 显著降低数据传输延迟，适合实时计算任务。
- 缓存一致性：提供统一虚拟地址（UVM）能力，允许共享统一内存空间，使多个 GPU 可像访问本地内存一样访问远程 GPU 显存。支持 GPU 间或 GPU-CPU 间的缓存一致性，减少数据拷贝开销，显著提升大规模模型训练效率。
- 可扩展性：支持 GPU-GPU、GPU-CPU 直连，通过 NVSwitch，可构建全互连网络（all-to-all 通信），使每对 GPU 之间都有直接高速连接。

协议栈

NVLink 采用精简的三层协议栈设计（物理层、数据链路层、事务层），基于专有的高速信令互连 NVHS（NVLink High Speed）技术构建。

1. 物理层

物理层核心依赖高速 SerDes（串行器/解串行器）技术，采用差分信号对传输，支持多速率自适应与极性/通道反转，减少 EMI 干扰。链路基础单元为 sub-link（8 对差分线，单向），多条 sub-link 可堆叠形成双向 link，提升带宽；编码方式随版本升级，NVLink 1.0 和 2.0 使用 NRZ 编码， 4.0 以后使用 PAM4 编码，实现带宽翻倍；引入时钟数据恢复（CDR）与自适应均衡技术，确保高速传输下的信号完整性。

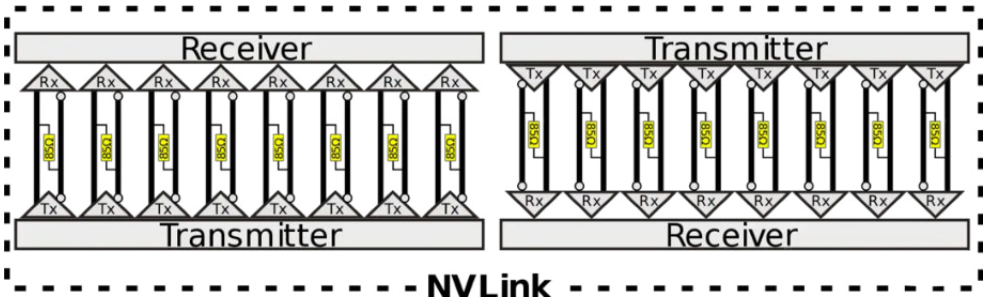


图130 NVLink 示意图 [94]

如上图所示，每个 NVLink 是一个双向接口，由 8 个差分对组成，总计 32 条线。

2. 数据链路层

数据链路层位于物理层之上，负责数据的可靠传输、错误控制与链路管理，采用 CRC 校验和选择性重传机制。数据链路层采用 flit（flow control digit）作为基本传输单元，基于信用机制（credit-based flow control)的双向流量控制，防止接收端缓冲区溢出，确保链路高效利用。

数据链路层特性参数：

- 最小传输单元：128-bit Flit（16 字节）
- 数据包组成：Header Flit + Payload Flit
- 数据包长度：1～18 Flit（16B～288B）
- 链路层错误校验：25-bit CRC
- VC：2 个（VC0 高优 / VC1 大数据）

3. 事务层

事务层负责事务包封装、地址映射与一致性协议管理。采用“固定包头+可变数据”的事务包（TLP）格式，支持内存事务（读/写，64 位寻址）、I/O 事务（兼容 PCIe 生态）与原子事务；基于目录式 MESI-like 协议，以 GPU 的 L2 缓存作为一致性节点，实现 GPU 间全版本缓存一致性，NVLink 2.0+支持 CPU-GPU 基础一致性，NVLink-C2C 实现完整系统级一致性；配合 CUDA 统一内存，自动管理数据迁移与一致性。

NVLink Switch 交换机

2018 年，英伟达 推出了 NVLink Switch 技术，实现了在 8 个 GPU 的网络拓扑中每对 GPU 之间高达 300 GB/s 的 all-to-all 带宽，为多 GPU 计算时代的 scale-up 网络奠定了基础。随后，在第三代 NVLink Switch 中引入了 NVIDIA 可扩展分层聚合与归约协议（SHARP）技术，进一步提升了性能，有效优化了带宽性能并降低了集合操作的延迟。

2022 年，英伟达将 NVSwitch 芯片独立出来，并制作成 NVLink 交换机，用于连接主机之间的 GPU 设备。

NVSwitch 物理交换机将多个 NVLink GPU 服务器连接成一个大型 Fabric 网络，即 NVLink 网络，解决了 GPU 之间的高速通信带宽和效率问题。每个服务器都有独立的地址空间，为 NVLink 网络中的 GPU 提供数

据传输、隔离和安全保护。NVLink 在系统启动阶段执行连接建立，也可以在软件通过运行时 API 调用时建立连接。实现网络随服务器上下线、用户接入与退出实时动态重配置。

英伟达 NVL72 互联如下所示：

每个 B200 芯片有 18 个 NVLink 端口，每个端口连接一个 NVSwitch 芯片，整个系统 18 颗 NVLink Swtich 芯片 72 个端口，72 颗 B200 芯片实现全互联。

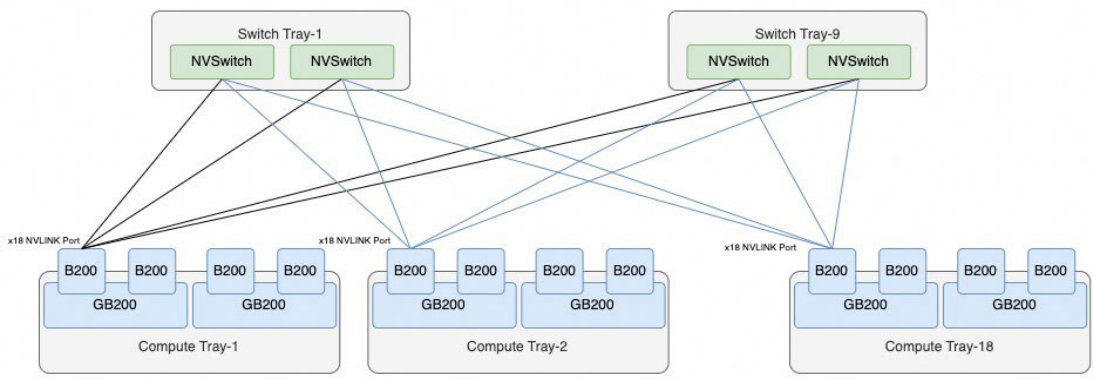


图131 NVL72 互联拓扑 [95]

NVLink C2C

NVLink-C2C（Chip-to-Chip）是指基于 NVLink 协议的、用于芯片到芯片 极近距离高速互连的物理封装和连接技术。它能够在单个封装中将两个处理器连接成一块超级芯片。例如可以通过连接两块 CPU 芯片，使 Grace CPU 超级芯片具有 144 个 Arm Neoverse V2 核心。还可以将 Grace CPU 和 Hopper GPU 连接成 Grace Hopper 超级芯片，将 CPU 和 GPU 的计算能力集合到一块芯片中。它提供了统一的缓存一致性内存地址空间，将系统和 HBM GPU 内存融合在一起，极大简化了编程体验。通过先进封装和单端并行信号设计，实现 900GB/s 双向带宽。

关键特性：

- 高带宽与低延迟：支持 900GB/s 双向带宽（450GB/s/方向），采用单端并行信号设计（9 条数据信号+1 条时钟信号），配合 40Gbps per pin 的传输速率，实现低延迟通信。
- 内存一致性：支持硬件缓存一致性：CPU 与 GPU 的核心可缓存彼此内存中的数据（以缓存线粒度），无需软件干预即可保证数据一致性。例如，在 Grace Hopper 超级芯片中，Grace CPU 可缓存 Hopper GPU 的 HBM 内存数据，GPU 也可缓存 CPU 的 LPDDR5X 内存数据，大幅减少数据迁移开销。开发者无需手动管理设备内存分配与数据传输，可像访问本地内存一样访问跨芯片内存，提升开发效率。
- 先进封装支持：依托硅中介层（Silicon Interposer）、CoWoS（Chip-on-Wafer-on-Substrate）等先进封装技术，实现芯片间的超近距离互连(如 Grace CPU 与 Hopper GPU 封装在同一超级芯片内)。这种封装方式不仅提升了信号完整性（减少串扰、衰减），还降低了功耗。

NVLink Fusion

NVLink Fusion 技术可让定制芯片（包括 CPU 和 XPU）与 NVIDIA 的 NVLink scale-up 网络技术以及机架级扩展架构相集成，从而实现半定制化的 AI 基础设施部署。

NVLink Fusion 为定制 CPU、定制 XPU 或两者的组合配置提供了灵活的解决方案。作为模块化开放计算项目（OCP）MGX 机架架构的一部分，NVLink Fusion 可与任何网卡（NIC）、数据处理器（DPU）或横向扩展交换机集成，使客户能够根据需求灵活构建理想的系统。技术的核心是“有控制的开放”，既允许合作伙伴定制芯片，又通过 NVLink 的技术绑定保持 NVIDIA 在 AI 生态中的主导地位。

NVLink Fusion 的技术架构基于 NVIDIA 的 NVLink 5.0，通过两种方式实现异构芯片的互联：

1. 自定义 CPU 连接 NVIDIA GPU：

第三方定制 CPU（如富士通 Monaka、高通定制 CPU）可通过 NVLink IP 集成到 CPU 设计中，实现与 NVIDIA GPU 的高速互联。例如，富士通的 Monaka 处理器通过 NVLink Fusion 与 NVIDIA GPU 连接，带宽可达 PCIe 5.0 的 14 倍（约 128GB/s），显著提升 CPU-GPU 协同效率。2026 年 1 月，英特尔宣布其定制 CPU 将支持 NVLink Fusion。

2. NVIDIA CPU 连接非 NVIDIA 加速器：

NVIDIA 的 Grace 或未来的 Vera CPU 可通过 NVLink Fusion 与第三方加速器（如 ASIC）互联。这需要在加速器设计中集成 NVLink IP，或在封装中加入支持互联的 chiplet（小芯片）。例如，NVIDIA 的 Grace CPU 可与第三方 ASIC 通过 NVLink 互联，构建“NVIDIA CPU+定制 ASIC”的混合系统。

需要注意的是 NVLink Fusion 并非完全开放：

- 必须使用 NVIDIA 的 NVLink IP 或芯片（如 Grace CPU、NVLink Switch），否则无法实现高速互联；
- 无法将非 NVIDIA 芯片（如 Intel CPU）直接连接到非 NVIDIA 芯片（如 AMD GPU），必须有至少一方是 NVIDIA 芯片。

英伟达 NCCL[96]

NCCL 是 Nvidia Collective multi-GPU Communication Library 的简称，NVIDIA 集合通信库（NCCL）可实现针对 GPU 和网络进行性能优化的多 GPU 和多节点通信基元。NCCL 提供了 all-gather、all-reduce、broadcast、reduce、reduce-scatter、point-to-point send 和 receive 等例程，这些例程均经过优化，可通过节点内的 PCIe 和 NVLink 高速互联以及节点间的 IB 网络实现高带宽和低延迟。

主要特性：

- 对 AMD、ARM、PCI Gen4 和 IB HDR 上的高带宽路径进行自动拓扑检测
- 凭借利用 SHARP 的网络内 all reduce 操作，将峰值带宽提升 2 倍
- 通过图形搜索，找到更佳的高带宽、低延迟的环和树集合
- 支持多线程和多进程应用

- InfiniBand verbs、libfabric、RoCE 和 IP Socket 节点间通信
- 使用 Infiniband 动态路由重新路由流量，缓解端口拥塞

1. NCCL 基本架构

NCCL 中，每个参与通信的进程被定义为一个 “rank”，并分配全局唯一的 rank ID 作为身份标识。多个可互相通信的 rank 共同组成通信器（communicator）。单个进程可同时加入多个 communicators，且在不同 communicators 中可拥有不同的 rank ID；每一个参与通信的 GPU 设备，都必须创建对应的 NCCL communicators 对象实例，才能进行通信操作。

NCCL 面向用户主要提供 4 大模块：

- 通信器管理（Communicator）：NCCL 的全部通信行为，都必须在 Communicator 的上下文环境内执行。每一个参与通信的 GPU，都会持有一个 Communicator 对象实例（对应 ncclComm），作为发起 NCCL 操作的核心入口。用户在发起通信前，必须先完成 Communicator 的初始化流程，明确指定参与本次通信的 GPU 设备范围。
- 集合通信（Collective Communication）：NCCL 提供 5 种集合通信原语分别为 ncclAllReduce、ncclBroadcast、ncclReduce、ncclAllGather 和 ncclReduceScatter，覆盖了分布式训练中绝大多数的多卡数据交互场景。
- 点对点 P2P 通信（Point-to-Point Communication）：NCCL 具备灵活的点对点通信能力，一方面通过 ncclSend 与 ncclRecv 接口实现双边协同的点对点数据收发；另一方面也支持单边远程内存访问（RMA）操作，允许进程无需目标进程显式配合，直接访问远端内存，这类操作需要通过 ncclCommWindowRegister() 接口，基于对称内存窗口完成目标内存的预先注册。
- Group 调用（Group Calls）： 是 NCCL 为降低操作启动开销、减少通信延迟设计的批量操作机制，NCCL 提供了 ncclGroupStart 与 ncclGroupEnd 成对接口，可将一系列连续的 NCCL 调用封装为一个操作组，所有组内的操作都会延迟到 ncclGroupEnd 调用时，才会被统一提交执行。组内可封装多组 Send/Recv 调用，也可封装多组集合通信操作；通过批量提交执行的机制，可以合并启动开销从而降低启动成本，提升通信效率。

2. NCCL 数据传输模式

节点内数据传输

NCCL 针对单节点内的多 GPU 通信，设计了分层化的传输架构，核心原则是优先选用同物理机内 GPU 间延迟最低、带宽最优的传输路径。这套传输策略的核心基础是 NVIDIA 的 GPUDirect P2P 技术，该技术可让 GPU 直接读写节点内其他 GPU 的显存空间，无需经过 CPU 系统内存做中转拷贝。

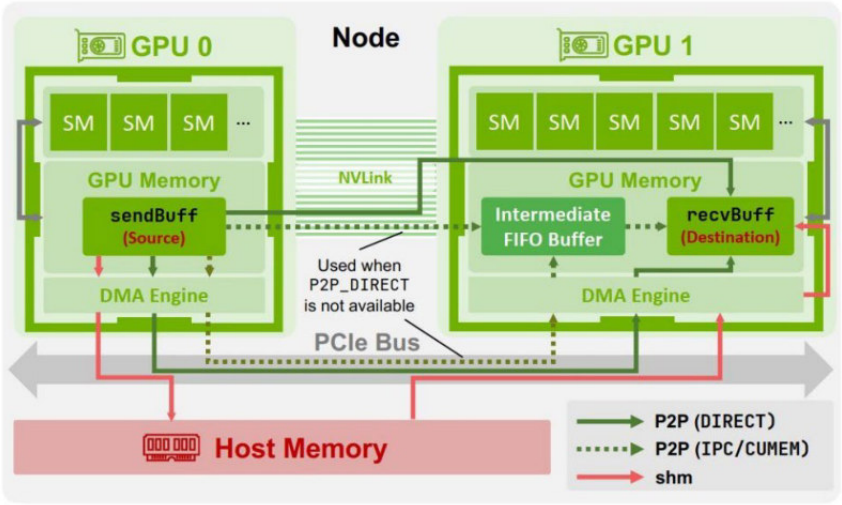


图132 节点内数据传输路径 [97]

节点内数据传输主要包含三类核心路径，分别为 P2P (DIRECT)、P2P (IPC/CUMEM) 和 SHM 共享内存传输：

- P2P Transport (P2P 传输)：P2P 传输是核心策略。P2P 传输是 NCCL 节点内通信的核心实现方案。当多 GPU 之间通过 NVIDIA NVLink 高速互联时，NCCL 会优先选择该链路，基于 NVLink 实现 GPUDirect P2P 通信，充分释放专用直连链路的带宽优势。当 NVLink 不可用时，NCCL 会自动降级，通过 PCIe 总线继续实现 GPUDirect P2P 通信，相比通过主机内存中转的方案，仍能获得显著的性能提升。
- P2P_DIRECT 优化：当发起通信的两个 rank 属于同一个进程时，NCCL 会自动启用 P2P_DIRECT 模式。该模式的效率提升主要来自两方面：其一，同一进程内的地址空间共享，可直接使用 GPU 内存指针完成寻址，无需通过 IPC 句柄做跨进程内存映射；其二，通过 directSend、directRecv 等底层原语，实现源缓冲区与目标缓冲区之间的直接数据传输，无需经过中间 FIFO 缓冲区做中转，消除了额外的数据拷贝开销。
- Shared Memory (SHM) Transport (共享内存传输)：当 GPU 间无法启用直连 P2P 通信，或是 P2P 通信无法达到最优性能时（典型场景如跨 CPU 插槽的 PCIe P2P 传输，往往会出现性能衰减），NCCL 会切换至 SHM 共享内存传输模式。该模式通过主机系统内存来路由通信流量，依托 CPU 对 PCIe－内存、内存－PCIe 传输的优化，规避跨插槽 P2P 的性能损耗。在 SHM 模式下，发送端 GPU 对应的进程先将数据写入共享内存段，再由接收端 GPU 对应的进程从该共享内存段中读取数据，完成整个传输流程。
- 通过 NIC 进行节点内通信：在部分多 CPU 插槽的服务器系统中，若不 GPU 分别挂载在不同的 CPU 插槽下，且每个插槽都配备了支持 GPUDirect RDMA 的本地网卡，NCCL 的拓扑感知逻辑会自动评估并选择通过网卡实现节点内 GPU 通信。这种 GPU-NIC-NIC-GPU 的传输路径，可直接利用 PCIe 带宽完成数据交互，无需经过 CPU 插槽间的互联链路，从而规避跨插槽 CPU 互联带来的性能瓶颈，该行为可通过 NCCL_CROSS_NIC 等环境变量进行控制。

节点间数据传输

节点间通信负责协调不同物理服务器节点上 GPU 之间的数据交互，完整的通信流程涉及三大核心组件：GPU 端负责执行 NCCL 通信内核、CPU 端的代理线程负责管理网络操作，以及底层网络硬件与驱动构成的通信底座。

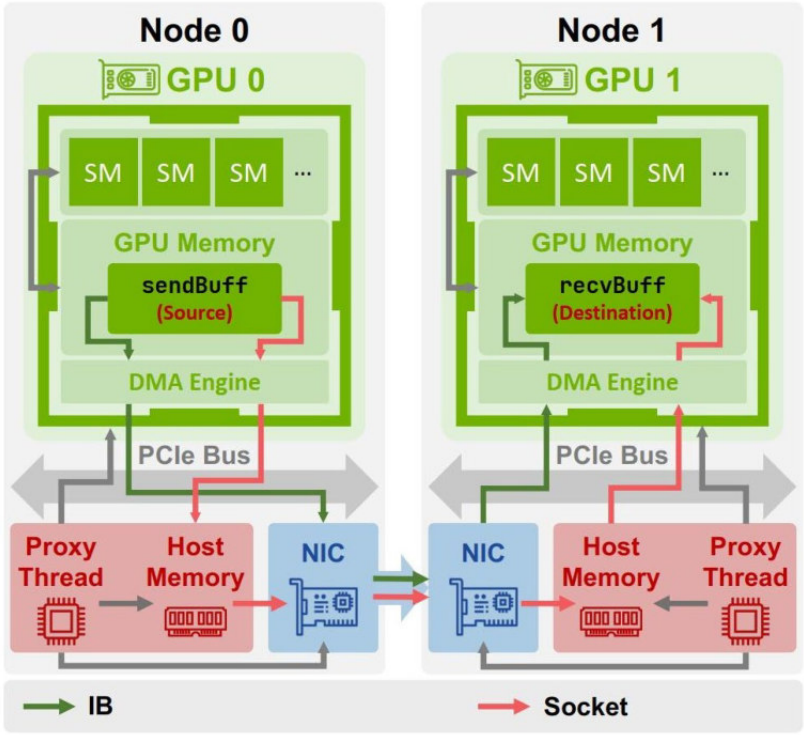


图133 节点间数据传输路径 [98]

节点间主要包含两种核心传输路径，分别为基于 InfiniBand (IB) 的高性能传输，与基于 Socket 的通用传输，其中 IB 路径可通过 GPUDirect RDMA 实现 GPU 显存与网卡之间的直接数据传输，而 Socket 路径则需要通过主机内存作为中转完成数据拷贝。

- Socket-Based Communication (基于套接字的通信)：当服务器的网络接口卡不支持 RDMA 时，NCCL 会采用 Socket 传输模式。该模式下，发送端需要先将数据从 GPU 显存拷贝到主机固定内存，再通过标准 Socket 接口完成网络发送；接收端则先将网络数据收至主机固定内存，再拷贝到目标 GPU 显存中。这种以主机内存为中转的传输模式，会带来额外的 PCIe 总线数据拷贝开销。
- IB Verbs Transport (IB Verbs 传输)：针对 InfiniBand、RoCE 等高性能 RDMA 网络，NCCL 提供了专用的 IB 传输层实现。该模式利用 RDMA 技术的远程直接内存访问能力，在几乎无需 CPU 介入的情况下，完成节点间的高速数据传输。

3. NCCL 通信算法

NCCL 可智能识别集群的网络硬件拓扑，为多 GPU 分布式环境动态选择最优的通信策略。

Ring 算法

Ring 算法是 NCCL 中实现多 GPU 通信的基础核心算法。该算法会为参与通信的 GPU 构建一个逻辑环形拓扑，每个 GPU 仅与环上直接相邻的两个上下游 GPU 完成数据收发。这种设计让数据在环上逐跳传递，直至完成完整的通信流程，凭借其简洁的实现逻辑与对等的通信特性，为中小规模 GPU 集群提供了高效稳定的通信方案。

但 Ring 算法并非适用于所有场景，实际使用中需要在通信延迟与带宽利用率之间做针对性权衡。例如基于 Ring 实现的 AllReduce 操作，在小数据量传输场景下，往往会出现延迟偏高、带宽利用率不足的问题。同时，GPU 间的物理连接拓扑存在显著的异构性，也会直接影响 Ring 算法的表现：单节点内的 GPU 通常通过 NVLink 高速互联，而节点间则多依赖 InfiniBand 网络，不同硬件厂商的服务器拓扑设计差异，也进一步提升了 Ring 算法的优化难度。

Tree 算法

Tree 算法的核心实现为双二叉树（double binary tree）架构，其设计核心是利用二叉树中近半数节点为叶子节点的特性，通过节点角色的转换构建出两棵互补的二叉树，确保每个节点在其中一棵树上为叶子节点，在另一棵树上则为非叶子节点。偶数节点数时，第二棵树通过镜像第一棵树生成；奇数节点数时，则通过单位置偏移的方式构建。这种设计让 Tree 算法在理论上能获得比 Ring 算法更低的通信延迟（时间复杂度 log2N 远低于 Ring 算法的 N），适合小数据量的低延迟通信场景。

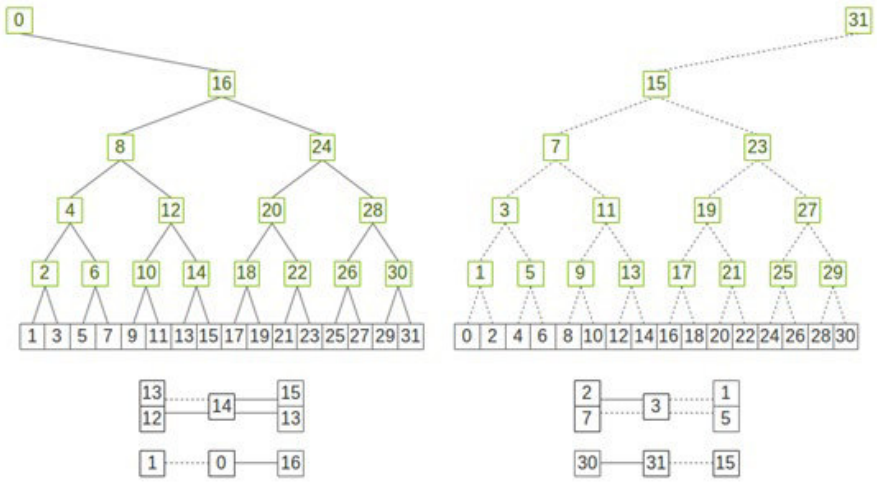


图134 节点间数据传输路径 [99]

CollNet 算法

CollNet 算法是基于 NVIDIA SHARP 技术打造的专用算法，专为 InfiniBand（IB）高性能网络场景设计，通过优化数据在网络中的传输与处理流程，大幅提升超大规模 GPU 集群的通信效率。

CollNet 算法的核心优势，在于其支持将集合通信中的计算任务卸载到网络交换机硬件中执行。该技术让交换机直接参与数据的聚合（Reduce）计算过程，无需在 GPU 间完成全量数据的端到端传输，不仅大幅降低了 GPU 的计算与通信负载，还能显著优化网络带宽资源的利用率，降低集合通信的整体延迟。

NVLS/NVLS Tree

NVLS 系列算法专为搭载 NVSwitch 硬件的平台设计，可深度调用 NVSwitch 内置的 SHARP 功能，实现更高效率的集合通信操作。

标准 NVLS 与 NVLS Tree 算法，均会调用 NVSwitch 的 SHARP 计算引擎完成节点内的数据归约计算：以 AllReduce 操作为例，节点内的 GPU 只需将数据写入 NVSwitch 的输入端口，由 SHARP 引擎完成硬件级的归约计算后，再通过硬件多播能力将最终结果同步至所有 GPU。二者的核心差异体现在跨节点通信的处理逻辑上：NVLS 算法会继续通过 CollNet 协议与支持 SHARP 的交换机，完成跨节点的归约计算；而 NVLS Tree 算法则在节点间采用 Tree 算法完成数据分发。这种设计让 NVLS Tree 既保留了 NVLS 算法极致的单节点通信性能，又解决了大规模集群下跨节点通信的扩展性问题。

4. NCCL 通信协议

NCCL 内置了 Simple、LL、LL128 三种核心通信协议，分别针对不同的通信场景做了深度优化，以平衡传输带宽与通信延迟的核心需求，运行时会根据硬件环境与数据规模自动选择最优协议。

Simple：这是 NCCL 的基础通信协议，核心设计目标是最大化带宽利用率，专为大数据量传输场景优化。通过大分块的数据传输策略，充分释放网络与 GPU 显存系统的吞吐能力，能实现接近链路峰值的带宽利用率，是 NCCL 处理大消息时的默认选择。

LL（Low Latency）：即低延迟协议，专为小数据量传输场景优化。当传输数据量较小时，传统协议的同步开销会成为性能瓶颈，LL 协议通过轻量化的基于标志位的同步机制，大幅降低同步耗时。该协议基于 CUDA 的 8 字节原子存储操作实现，将传输单元封装为 “4B Data+4B Flag” 的结构，接收端通过校验标志位即可确认数据有效性。

LL128：这是在 LL 协议基础上的增强优化版本，专为 NVLink 高速互联场景设计，同时兼顾了低延迟与高带宽两大核心需求。和 LL 协议一致，它同样采用标志位同步机制，但将传输单元扩大至 128 字节，其中 120 字节为有效数据，仅 8 字节为标志位，有效数据占比可达 93.75%，带宽利用率最高可接近链路峰值。在带有 NVLink 的机器上，NCCL 会默认使用该协议。

5. NCCL 自动拓扑检测

- 系统初始化与信息收集：

NCCL 的拓扑检测流程以 ncclTopoGetSystem () 函数为入口启动，全面采集当前节点内 GPU、网卡等各类硬件设备的物理连接信息。会从 Linux 系统的 sysfs 虚拟文件系统中，读取完整的 PCIe 设备层级结构，包括 PCI 域号、总线号、设备号、功能号（BDF 地址）等关键标识，同时会检测并记录设备的 NVLink 支持能力、PCIe 链路速率与链路宽度等核心性能参数。

检查设备是否支持 NVLink、PCIe 链路速度和链路宽度等参数。

- 全局拓扑信息聚合：

完成本地硬件信息采集后，NCCL 会通过 AllGather 集合通信原语，将集群内各个 GPU 进程采集的本地拓扑信息进行汇总，构建出整个集群的全局硬件拓扑视图。最终集群内的每一个 rank，都能获取到全集群所有 GPU、网卡的物理位置与连接关系信息。

- 拓扑建模与路径计算：

基于全局拓扑信息，NCCL 会构建出树形结构（通常以 XML 格式描述），完整还原设备间的物理连接关系。同时会综合链路带宽、传输跳数等核心指标，计算出所有 GPU 与网卡之间的最短、最优通信路径，并精准识别 NVLink、PCIe、InfiniBand、RoCE 等不同类型的互联链路，为后续通信优化提供依据。

- 通信优化：

基于完成的拓扑建模结果，NCCL 会为通信操作自动选择最优传输路径，例如优先选用 NVLink 链路而非 PCIe 链路完成数据传输；同时会根据实际的硬件拓扑结构，动态调整通信算法与通道分配策略，确保通信性能与硬件能力最大化匹配。

6. SHARP[100]

SHARP 全称 Scalable Hierarchical Aggregation and Reduction Protocol(可扩展分层聚合与归约协议)，是英伟达打造的网络内计算（In-Network Computing）技术。其核心目标是加速 AI/HPC 场景中高频的集合通信操作（如 AllReduce、Reduce、Broadcast 等），通过将集合通信操作从 GPU/CPU 端侧完全卸载到网络交换机（或 NVSwitch）硬件引擎中执行，减少网络数据传输量、降低延迟，提升分布式计算任务的整体性能。

实现机制：

- 硬件卸载：在交换机 ASIC（如英伟达 InfiniBand 交换机、NVSwitch）中集成计算引擎，支持 16/32/64 位定点/浮点运算（如求和、求平均、取最大/最小值等），可直接在网络转发层面完成数据的聚合计算，无需将全量数据传输至端侧 GPU 再做处理。
- 分层架构：通过聚合管理器（Aggregation Manager）在物理网络拓扑中构建逻辑 SHARP 树（如二叉树、双二叉树），将集合操作分解为多级交换机的并行处理。以 AllReduce 操作为例，终端节点先将数据发送至接入层交换机，由交换机完成本地节点的局部数据聚合，再将聚合后的结果上传至上级交换机，逐级汇聚直至根交换机完成最终的全局聚合，最后再将全局结果通过广播路径同步至所有终端节点。

与 NCCL 集成：NCCL 可与 CUDA、网卡驱动、NCCL-SHARP 插件等配套软件栈联动，通过标准化的库接口直接调用 SHARP 能力，将核心的集合通信操作卸载到网络设备中执行，实现通信性能的跃升。同时，NCCL 的拓扑感知能力可动态识别集群内的节点状态、链路状态，优化数据聚合路径与硬件卸载策略。

3.6 UALink 协议介绍[101]

UALink 协议的目标与核心特性

UALink 协议由 Ultra Accelerator Link Consortium 组织牵头开发，其核心成员汇聚了全球产业链各领域的巨头企业，包括 AMD、Astera Labs、AWS、Cisco（思科）、Google（谷歌）、HPE（慧与）、Intel（英特尔）、Meta、Microsoft（微软）、阿里云(Alibaba Cloud)、Apple（苹果）、Synopsys（新思科技）。

UALink 协议的核心使命十分明确：建立一套开放、可扩展、高性能、高可靠且低成本的 AI 加速器 Scale-up（纵向扩展）互联标准。它依托目前已非常成熟的以太网生态，打造能够跨厂商、支持大规模部署、实现高效协同的 AI 集群互联体系，最终替代英伟达封闭的专有互联方案，重塑整个 AI 算力基础设施的格局。

UALink 与同类 Scale-up 协议的核心差异

UALink 协议与我们之前提到的 OISA、ETH-X、SUE 等 Scale-up 协议，虽然都基于以太网技术，但在技术实现上有显著区别，主要集中在以太网层的使用、帧格式与相关机制，以及转发机制三个核心方面。

1. 以太网层使用的差异

OISA、ETH-X、SUE 等协议在使用以太网时，会同时用到以太网的物理层（负责信号传输）和数据链路层（负责数据帧的封装与传输控制），并且这些协议在数据链路层的帧格式、LLR（链路层重传机制）、CBFC（拥塞控制帧）的帧格式及实现机制上基本一致。

而 UALink 协议仅使用了以太网的物理层，其数据链路层的相关实现与上述协议完全不同——无论是帧格式，还是 LLR、CBFC 的实现机制，都有自身独特的设计，这也是 UALink 与同类协议最基础的区别。

2. 转发机制的差异

UALink 与 OISA、ETH-X、SUE 等协议的核心目标是一致的，都是实现 AXI 或 UPLI 操作在不同节点（如 GPU）之间的传递，但两者的转发方式差异极大。为了方便大家理解，我们用“交通运输”的场景做一个通俗比喻，清晰区分两种转发机制的不同。

- **OISA、ETH-X、SUE 的转发机制（“小车运输模式”）**

当 GPU 发出 AXI 操作指令时，这些协议会先识别每个指令的“目的地”——也就是目的 GPU 的 ID，然后将去往同一个目的地的一个或多个指令，装载到不同的“小车”上（这里的“小车”对应协议中的事务层报文）。需要注意的是，每一辆“小车”（一个事务层报文）只负责运送去往一个目的 GPU ID 的信号。

此时，Scale-up 交换机的作用就类似于交通路口的信号灯：它不需要拆开“小车”查看内部货物，只需要根据每辆“小车”上标注的目的地（目的 GPU ID），将“小车”从进入交换机的端口，引导到去往对应目的地的出端口，完成转发即可。

- **UALink 协议的转发机制（“公共汽车运输模式”）**

UALink 协议处理 GPU 发出的 UPLI 操作指令（功能类似 AXI 操作指令，可理解为 UALink 定义的“内存读写（Read/Write/Atomic）指令”）时，采用了完全不同的处理思路：它不识别每个信号的目的 GPU ID，而是将同一个端口收到的所有信号，全部装载到一辆“公共汽车”上（这里的“公共汽车”也是 UALink 的事务层报文）。也就是说，一辆“公共汽车”（一个事务层报文）中，会包含去往多个不同目的 GPU ID 的指令。

对应的，Scale-up 交换机的作用也发生了变化：它不再是简单的“信号灯”，而是需要在“站台”（交换机入端口）先将“公共汽车”上的所有“乘客”（指令）全部卸下来，逐一查看每个“乘客”的目的地（目的 GPU ID），再将这些“乘客”重新分配到去往不同目的地的下一辆“公共汽车”的站台上；下一辆“公共汽车”到站后，再从对应的“站台”（交换机出端口），载上所有的“乘客”（指令），驶向下一个目的地，以此完成转发。

3. 转发机制差异带来的利弊

UALink 的“公共汽车运输模式”与同类协议的“小车运输模式”相比，有明显的优势，也存在一定的额外开销，我们分别进行说明，方便大家全面理解。

- **核心优势**
    - 提升数据承载效率：相比于“小车”单独运输，“公共汽车”能更高概率地实现多个 UPLI 指令的聚合，让多个指令共用一个事务层报文和数据链路层报文。这样一来，就减少了不同报文之间的间隔开销（类似“小车”之间的空隙），从而提升了整个链路的数据承载效率。

- 降低 GPU 侧实现成本：在 GPU 端，UALink 不需要根据目的 GPU ID 对指令进行分类聚合，因此也不需要设置基于目的 GPU ID 的专属队列。这一设计简化了 GPU 侧 ASIC 芯片（专用集成电路）的实现复杂度，减少了芯片所需的硬件资源，最终降低了 GPU 芯片的成本。

- **额外开销与应对方式**
  - 增加交换机实现复杂度：由于交换机需要对“公共汽车”（事务层报文）进行“拆包（卸乘客）—分拣—组包（装乘客）”的操作，相比同类协议中交换机仅需“引导方向”的简单操作，UALink 协议下交换机的实现复杂度明显增加。
  - 易出现 in-cast 流量拥塞：in-cast 流量（可理解为“多个端口的流量同时涌向同一个出端口”）会导致交换机出端口瞬间拥塞，出现“头阻塞”（即前面的报文堵塞，后面的报文无法正常转发）。UALink 系统需要依靠交换机的 Buffer（缓冲区）来暂时存储拥堵的报文，平滑瞬间的 in-cast 拥塞，避免影响整个 AI 集群中各 GPU 之间的正常通信。若出现持续的 in-cast 拥塞，需从应用软件设计层面进行优化以规避该问题。

小结

综上，UALink 协议在设计思路上，不仅与 OISA、ETH-X、SUE 等同类 Scale-up 协议存在显著差异，与 UB、CXL 等其他互联协议也有所不同。本章对 UALink 协议的开发背景、核心使命及与同类协议的差异进行了详细讲解，目的是帮助大家从全新的视角理解 Scale-up 网络，掌握一种实现 Scale-up 网络的新方法，为后续深入学习 UALink 协议的细节奠定基础。

UALink 协议的主要功能与协议栈

UALink 协议作为 AI 加速器 Scale-up 互联的标准之一，其设计核心是实现不同系统节点之间，加速器与加速器的高性能、低时延、无损通信，让 AI 集群中的多个加速器能够协同工作、高效配合。本章将详细介绍 UALink 协议的主要功能、组网方式，以及协议栈各层级的具体作用，帮助大家全面理解 UALink 协议的实现逻辑与核心架构，为后续学习打下坚实基础。

UALink 协议的主要功能

UALink 协议的核心功能主要集中在内存语义处理和集合通信两大方面，具体可分为以下两点，大家结合 AI 集群的工作场景就能轻松理解：

第一，支持直接内存读、写及原子事务，这是加速器之间实现数据交互的基础能力。我们知道，AI 集群运行时，不同加速器之间需要频繁地读取、写入彼此的内存数据（比如共享模型参数、交互中间计算结果）。而原子事务的作用，就是保障跨加速器正确地访问内存，如同在一个加速器中，从而为 AI 模型的稳定、高效的并行运行提供保障。

第二，协议处于持续迭代优化中，功能不断扩展。目前，UALink 2.0 版本正在开发中，相比现有版本将增加“集合通信”功能，在交换机实现“在网计算”功能值得重点关注。简单来说，“在网计算”就是让数据在传输过程中，直接在交换机上完成归约的计算，不用把所有数据都传到加速器后再进行运算。这样一来，能大大减少数据传输的总量，进一步降低时延，让整个 AI 集群的处理效率再上一个台阶。

UALink 的组网拓扑

UALink 协议的组网拓扑，本质就是围绕“加速器与交换机如何连接”来设计的，目的是实现大规模加速器的高效互联，满足 AI 集群对通信可靠性和带宽的需求。其结构特点非常清晰（可参考下图），我们分两点来拆解：

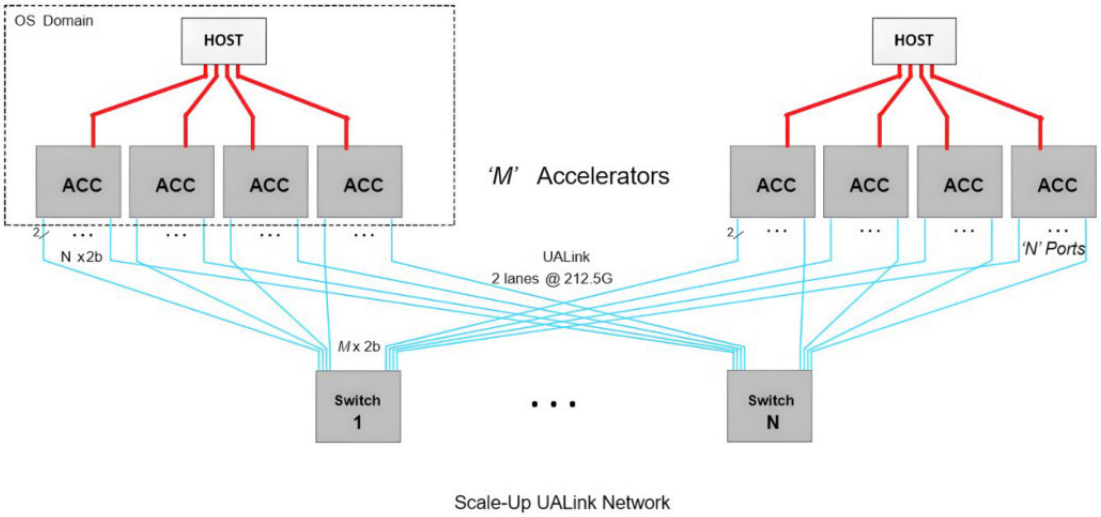


图135 UALink 多加速器系统

首先，每个加速器会配备 N 个端口，每个端口分别连接到不同的交换机。这种设计相当于为加速器之间的通信，搭建了多个独立的“数据通道”（也就是我们说的交换平面）。这样做有两个明显的好处：一是提升可靠性，即使其中一个“通道”（交换平面）出现故障，其他“通道”也能正常工作，不会导致整个通信链路中断；二是提升带宽，多个“通道”可以同时传输数据，减少单一链路的拥堵，避免出现数据传输“排队”的情况。

其次，组网中使用的交换机，有一个关键参数——Radix（交换机总端口数），它的数值 M 决定了单台交换机最多能连接多少个加速器。根据 UALink 协议的规定，目前 M 的最大值为 1024，也就是说，单台交换机最多可以连接 1024 个加速器。根据当前 UALink 的组网设计，一个 UALink Scale UP 系统最多支撑 1024 个加速器协同工作，当前可以满足超节点 Scale UP 组网需求，超过 1K 规模的超节点，可以考虑使用 Scale OUT 组网或未来协议的升级。

UALink 的协议栈结构

UALink 协议栈采用“分层设计”，每一层都有明确的分工，层层协同，完成从加速器到加速器 UPLI 所需要实现的内存访问功能。协议栈从上至下，依次分为协议层（Protocol Layer）、事务层（Transaction Layer）、数据链路层（Data Link Layer）和物理层（Physical Layer）（如下图）。下面我们逐一讲解每一层的具体作用，大家重点记住“每一层做什么、和上下层如何配合”即可。

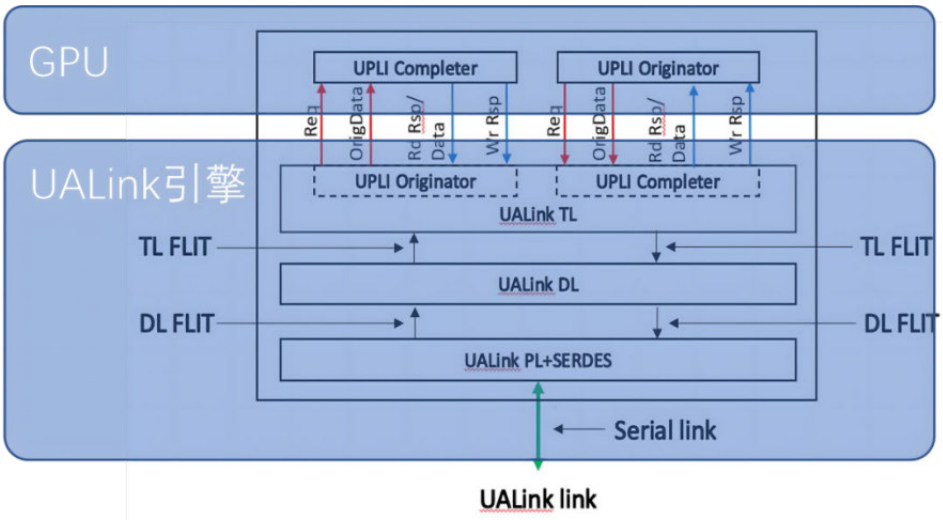


图136 UALink 协议栈

1. 协议层（Protocol Layer）：UALink 协议层接口（UPLI）

协议层是 UALink 协议的“顶层接口”，全称为 UALink 协议层接口，简称 UPLI，它的位置在 GPU 加速器和 UALink 引擎之间，核心作用非常简单：搭建一个“桥梁”，实现加速器与 UALink 引擎之间的数据和控制信息交互。

具体来说，UPLI 定义了一套标准的“沟通规则”（指令），设备之间通过这套规则，用“请求（Request）”和“响应（Response）”的方式完成交互：比如加速器要发送向内存写数据的指令，就通过 UPLI 把请求传给 UALink 引擎；UALink 引擎处理完请求后，再通过 UPLI 把写内存的结果反馈给加速器。

这里有一个小知识点可以帮助大家记忆：UPLI 的位置，和我们之前学习的 SUE、OISA、ETH-X 协议中 AMBA AXI 接口的位置是一样的，功能也基本相同，都是负责连接 GPU 内部的 NOC（片上网络）和 Scale UP IO 引擎。而 UALink 明确了 UPLI 接口的全部内容，便于不同厂商加速器只要都采用相同的 UPLI 接口，就可以实现互联互通，如果定义不完整，就会导致因出现各厂商自己定义的私有接口，而无法互通。

2. 事务层（Transaction Layer）

事务层是整个协议栈的“核心环节”，相当于数据传输的“打包和解包中心”，主要负责读懂上层指令并封装为报文，同时也可以把下层报文解封装为上层可以识别的指令。它的上下行接口分工明确，我们结合“数据流向”来讲解，更容易理解：

- 北向接口（与加速器连接的接口）：事务层通过两路 UPLI 接口和加速器相连，这两路接口分工不同、协同工作，大家可以简单记为“主动通信接口”和“被动通信接口”：
    - 一路连接 GPU 加速器的“UPLI 发起端”，负责“主动通信”：比如本 GPU 要向其他 GPU 发送数据或请求，就通过这个接口发送 Request（请求）消息；同时，其他 GPU 对本 GPU 的请求做出的 Response（响应）消息，也通过这个接口接收。

- 另一路连接 GPU 加速器的“UPLI 完成端”，负责“被动通信”：比如其他 GPU 要访问本 GPU 的资源，发送的请求消息会通过这个接口接收；本 GPU 处理完请求后，再通过这个接口向对方返回 Response（响应）消息。

- 南向接口（与数据链路层连接的接口）：事务层和数据链路层之间，以“TL Flit”为交互单元，核心任务就是发送和接收这些数据片。

  结合上下行的数据流，我们可以总结出事务层的完整工作流程：

  - 下行流量（从加速器到数据链路层）：加速器通过两路 UPLI 接口，把 UPLI 预定义的接口指令（包括动作和数据）传给事务层；事务层把这些指令，按照一个预定义的事务层报文格式，将指令“打包”成 64 字节为单位的发送（Tx）数据片 TL FLIT，传给数据链路层，进入下一步传输。

    事务层报文格式是该协议的核心所在，主要体现在三个方面：

    - 一是采用标准化格式，保证 Scale UP IO 引擎与 Scale UP 交换机能够正确识别报文，并依据报文中的信息完成相应处理，在交换机正确分发，在目的端恢复源端送来的动作和数据，供加速器执行；
    - 二是在有限的报文长度内，完整且高效地传输操作指令；
    - 三是在报文中携带流控信息，实现拥塞避免，保障传输稳定可靠。

  - 上行流量（从数据链路层到加速器）：事务层从数据链路层接收（Rx）TL FLIT，再把这些“包裹”“拆包”，按照事务层报文的定义，还原成动作和数据，通过 UPLI 的请求通道、原始数据通道、读响应/数据通道及写响应通道发送给加速器，完成加速器指令从源端到目的端的传递。

3. 数据链路层（Data Link Layer）

数据链路层位于事务层和物理层之间，相当于数据传输的“二次打包+链路管控中心”，核心作用是对数据片进行二次封装/解封装，同时负责链路控制和固件传输。为什么需要二次封装/解封装，主要是通过打包构建 640 字节 DL FLIT 大报文，可以提升 LLR（Link Layer Retry）的效率，同时 640 字节负载与 RS（544,514）FEC 的效率相关，640 字节正好可以在一个 FEC 码块中承载，FEC 效率高。具体功能我们依然结合数据流向来看：

- 数据的封装与解封装（核心功能）：

  - 下行流量（从事务层到物理层）：数据链路层接收事务层传来的多个 TL FLIT，会给它“加一层保护壳”——封装成 640 字节的数据片 DL FLIT（增加的内容主要是链路控制相关信息，比如重传机制、数据校验、链路标识等），然后再发送到物理层，为数据的物理传输做好准备。

  - 上行流量（从物理层到事务层）：数据链路层从物理层接收 640 字节的数据片，先把“保护壳”去掉（解封装，删除链路控制信息），还原成多个 TL FLIT，再转发给事务层，确保事务层能正确解析数据。

- 链路控制与固件传输：除了数据封装，这一层还有两个重要任务：一是提供控制消息服务，协调链路的状态变更（比如链路的上线、离线切换，以及其他功能配置），保障链路通信的稳定；二是配备了 UART（通用异步收发传输）机制，固件控制序列可以通过这条链路传输，相当于为整个协议栈的正常运行，提供了“底层软件支撑”。

4. 物理层（Physical Layer）

物理层是 UALink 协议栈的“最底层”，相当于数据传输的“物理通道”，核心任务就是实现数据的物理传输——把数据链路层发送的并行数字信号串行化，并通过物理介质（比如网线、光纤），实实在在地传到目标设备上。它的设计遵循明确的行业标准，具体要求我们重点掌握两点：

第一，兼容性强：物理层基于 IEEE 802.3dj 标准设计，这意味着它能和现有的以太网物理传输技术兼容，不用重新搭建全新的物理传输环境，降低了部署成本。

第二，支持多种传输速率：为了适配不同规模 AI 集群的需求，物理层支持两类基于 Serdes（串行器/解串器，作用是实现高速信号的串行传输与并行转换）的以太网速率，大家不用死记硬背，了解分类即可：

- 基于 200G Serdes 的以太网：包括 200GBASE-KR1/CR1、400GBASE-KR2/CR2、800GBASE-KR4/CR4；
- 基于 100G Serdes 的以太网：包括 100GBASE-KR1/CR1、200GBASE-KR2/CR2、400GBASE-KR4/CR4。

这种多速率支持的设计，让 UALink 协议可以根据 AI 集群的实际需求（比如传输效率要求、成本预算），灵活选择合适的传输速率，兼顾实用性和经济性。

跨域地址转换

在大规模 AI 集群中，不仅单个系统节点内部的加速器需要通信，不同系统节点之间的加速器也需要交互数据、共享资源。UALink 协议通过“跨域地址转换”，实现了这一需求。这部分内容稍显复杂，我们分步拆解，结合示例理解核心逻辑即可。

1. UALink 跨域地址转换模型

UALink 网络的一大核心优势，就是能支持多系统节点间的数据交互——既可以实现单个系统节点内部的加速器通信，也能支撑不同系统节点之间的加速器互通（如下图 UALink 跨域地址转换模型所示）。而要实现跨节点通信，关键就是解决“地址定位”问题：加速器访问不同系统域的内存时，需要使用对应的地址类型，同时通过地址转换，找到目标内存的准确位置。

- **核心地址类型**

  UALink 协议定义了两种地址转换流程，具体如下（重点区分两种地址的适用场景）：

  - 地址转换流程一：
    - 系统物理地址（System Physical Address, SPA）：专门用于访问“单个系统域”（比如一个独立的 AI 服务器节点）内部的内存资源，相当于本地内存的“专属地址”，只在本系统域内有效，能确保本地内存访问的精准性。
    - 网络物理地址（Network Physical Address, NPA）：专门用于访问“不同系统域”之间的内存资源，相当于跨域内存的“专属地址”，里面包含了地址句柄（相当于内存的“索引”）和目标标识信息，能精准定位到其他系统域的内存。
  - 地址转换流程二：

- 全局扁平寻址模型：它的好处是不用进行复杂的域间地址映射，能简化地址转换流程，降低协议部署和实现的难度，适合对地址转换效率有较高要求的场景。

- **地址转换的核心说明**

  这里有一个重要提醒，大家一定要注意：本章介绍的跨域地址转换模型，只是为了帮助大家理解核心逻辑的“示例”，并不是强制要求。因为 UALink 交换机采用的是“基于标识的路由”机制——交换机只需要根据请求中的源加速器标识、目标加速器标识，就能完成数据转发，不需要依赖固定的地址转换方法。但是在发起端需要实现 NOC 中对系统物理地址的访问，转换为全网可以认识的网络物理地址，并从网络物理地址中，找到对应的加速器标识，这个地址转换的过程是必须的，同时在完成端实现从网络物理地址到系统物理地址的转换，也是必需的。因此，UALink 协议规范中，把地址转换方式定义为“实现可选项”，开发者可以根据实际需求，灵活选择合适的转换方案。

- **跨域地址转换示例流程**

  下面我们通过一个典型的示例，一步步拆解跨域地址转换的全过程，结合流程记忆，更容易掌握（结合协议规范示例）：

  - 源加速器（比如节点 A 的加速器）想要访问目标节点（比如节点 B，属于不同系统域）的内存时，首先会通过自身的 MMU（内存管理单元，相当于“地址转换器”），把应用程序使用的客户虚拟地址（GVA，相当于我们平时使用的“快捷方式”），转换为网络物理地址（NPA），完成“虚拟地址→跨域物理地址”的第一步转换。
  - 源加速器把携带 NPA 的请求（Request），通过 UALink 网络发送出去；交换机收到请求后，根据请求中的源标识、目标标识，完成路由转发，把请求准确传递到目标节点（节点 B）。
  - 在目标节点（节点 B）一侧，通过 UALink 链路 MMU（专门负责跨域地址转换的“转换器”），把收到的 NPA，转换为目标节点本地的系统物理地址（SPA），完成“跨域地址→本地地址”的第二步转换。
  - 目标节点根据转换后的 SPA，访问本地内存、处理请求，之后生成响应（Response）消息，沿着原来的传输路径，反向传递回源加速器，整个跨域数据访问的地址转换与传输流程，就完成了。

UALink 协议的设计补充说明

这里有一个非常重要的设计特点，大家容易产生误解，需要重点说明：**UALink 协议没有实现缓存一致性（Cache coherency）机制。**

这并不是协议的设计缺陷，而是基于实际应用场景的合理选择。在大模型运行的场景中，用硬件实现跨加速器缓存一致性机制，带来的收益并不高——反而会大大增加协议实现的复杂度，提高硬件成本。因此，UALink 协议选择不支持缓存一致性机制，把精力集中在核心的通信功能（大规模、低时延、高带宽）上，优化加速器之间的通信效率，更符合 AI 集群的实际需求。

本章小结

本章围绕 UALink 协议的核心架构，重点讲解了协议的主要功能、组网拓扑，以及协议栈各层级的作用，核心知识点可以总结为以下几点，帮助大家梳理记忆：

- 核心目标：UALink 协议以“加速器间低时延、高带宽、无损通信”为核心，支撑 AI 集群中多加速器的协同工作；
- 主要功能：支持内存读、写及原子事务（基础功能），且在持续迭代（如 UALink 2.0 可能新增在网计算）；
- 组网拓扑：通过“多交换平面”（每个加速器的多个端口连接不同交换机）设计，实现大规模加速器的可靠、高效互联；
- 协议栈结构：四层结构分工明确，从顶层的 UPLI 接口（协议层），到数据封装（事务层）、二次封装与链路控制（数据链路层），再到物理传输（物理层），形成了完整的通信链路；
- 关键机制：通过跨域地址转换（SPA/NPA 两种地址类型），实现多系统节点间的加速器通信，保障访问的高效性和安全性；
- 设计特点：不支持缓存一致性机制，聚焦核心通信功能优化，兼顾实用性和经济性。

通过本章的学习，大家已经掌握了 UALink 协议的核心实现逻辑，接下来我们将深入学习协议的细节和实际应用。

UPLI 协议层接口（UPLI）

UPLI（UALink Protocol Layer Interface，UALink 协议层接口）是 UALink 协议栈最靠近加速器的顶层接口，也是 GPU 与 UALink 引擎之间进行事务交互的统一入口。

本章将详细介绍 UPLI 的主从通信模式、接口定义、通道划分、数据流路径、路由机制、流量控制以及读 / 写 / 原子三类事务的完整行为，帮助你从顶层理解 UALink 如何实现加速器之间可靠、有序、低时延的通信。

UPLI 接口定义

UPLI 采用主从式通信模型：一端主动发起事务，另一端被动处理并返回响应。协议中对两类设备的定义如下：

- 发起端设备（Originator Device）

  主动发送请求、启动一次事务的设备。

- 完成端设备（Completer Device）

  接收请求、执行操作，并向发起端返回响应的设备。

在实际系统中，一块 GPU 既要主动访问其他 GPU，也要响应其他 GPU 的访问，因此每个 GPU 会同时拥有两类 UPLI 接口：

- UPLI 发起端

  负责本 GPU 向其他 GPU 发出读、写、原子等请求，并接收对方返回的响应。每一条请求都必须收到响应，发起端通过事务标签（Tag）将请求与响应一一匹配，确保一次内存读写事务的可靠闭环。

- UPLI 完成端

  负责接收并处理其他 GPU 发送来的请求，执行本地操作后，向对方返回响应。

同时，每一个 UPLI 接口包含四条独立通道，分别承担不同方向、不同类型的数据传输：

- 请求通道（Req）
- 读响应 / 数据通道（RdRsp）
- 发起端数据通道（OrigData）
- 写响应通道（WrRsp）

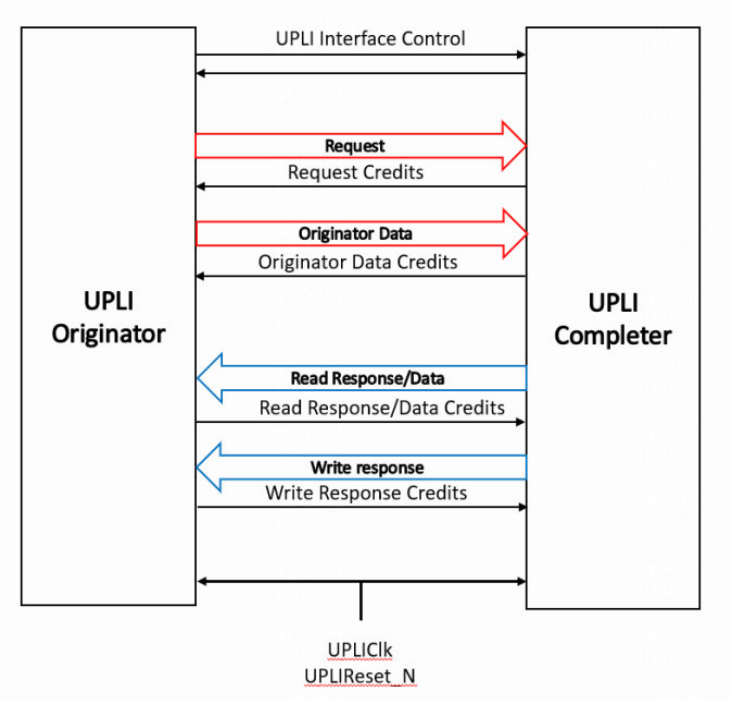


图137 UPLI 通道和控制信号

这四路通道的设计思想与我们熟悉的 AMBA AXI 协议非常相似，便于已有片上总线基础知识的同学快速理解。

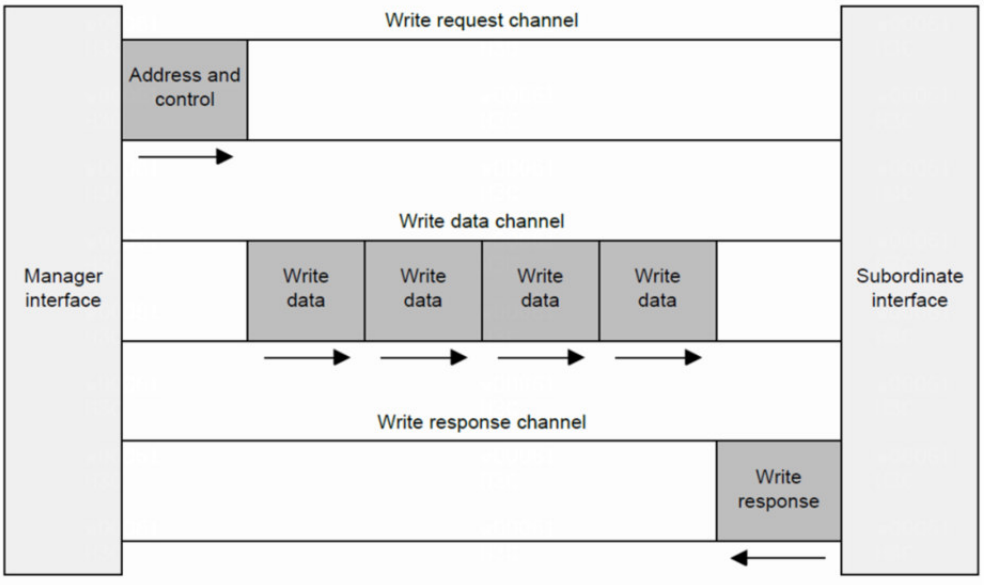


图138 AXI 通道设计 – 写指令 [102]

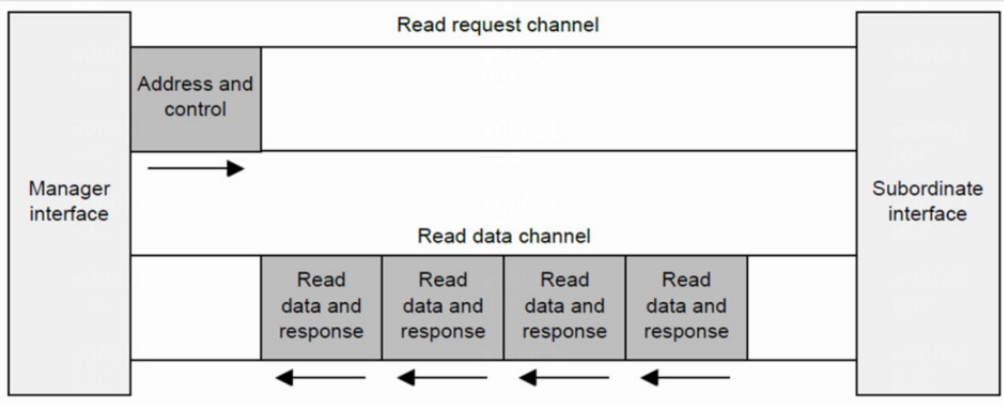


图139 AXI 通道设计 – 读指令 [103]

UALink 引擎的协议栈组件

UALink 引擎内部集成了完整的协议栈处理逻辑，结构清晰、层次分明，如图所示。

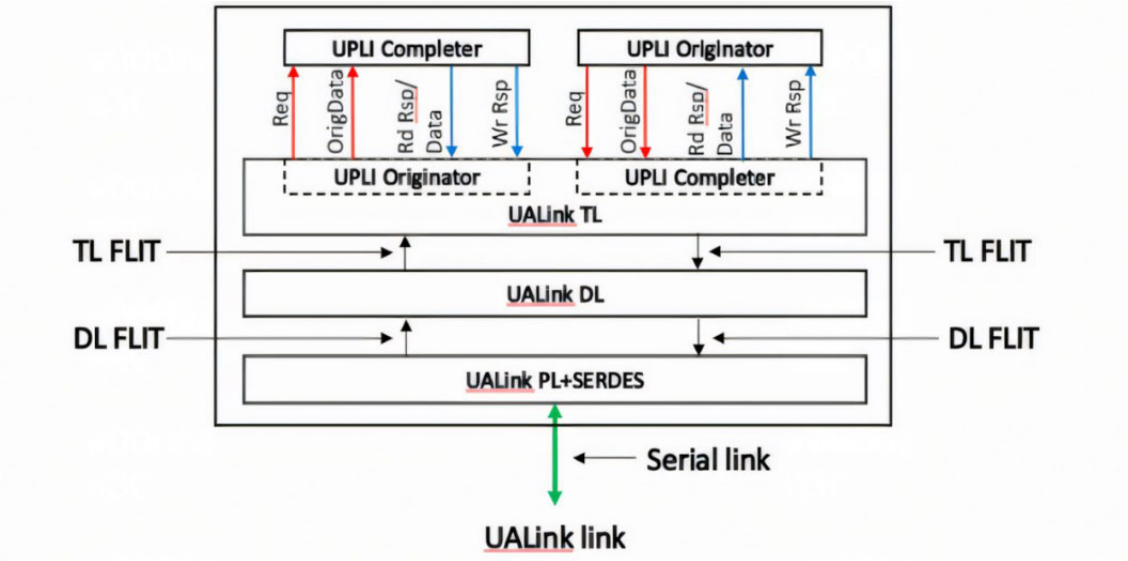


图140 UALink 协议栈

- GPU 加速器与 UALink 引擎的关系

GPU 加速器内部的逻辑结构如图：

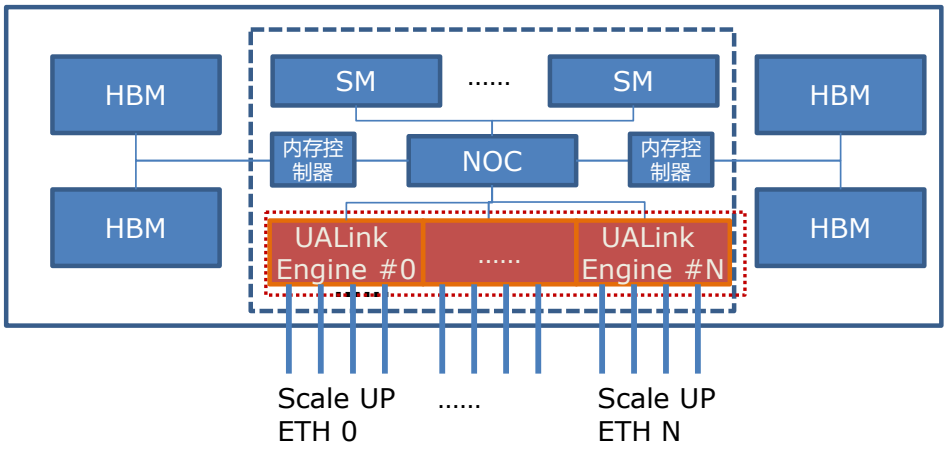


图141 GPU 加速器内部的逻辑

其中计算 Core、内存都挂在 NOC 总线完成 AI 计算，同时计算 Core 也需要通过 NOC 总线联接各种外设，并对各种外设提供标准化的接口。同时 NOC 作为 GPU 系统的核心技术，不断迭代，不便于公开全部接口，因此定义了标准化的 UPLI 接口，即保持外设接口的稳定性，也屏蔽了 NOC 总线的技术复杂性和版本迭代，持续支持未来各种外设通过此标准化接口与 NOC 总线连接。

- **UPLI 接口：**

  UALink 引擎作为一种外设，采用 UPLI 接口标准联接 NOC 总线，包括两个 UPLI 接口：

  - UPLI 发起端：处理本 GPU 访问其他 GPU 的请求与响应。
  - UPLI 完成端：处理其他 GPU 访问本 GPU 的请求与响应。

  UALink 引擎内部实现 UALink 事务层、数据链路层、物理层，最终通过以太网接口与 UALink 网络连接。

- **UALink 事务层（TL）**
  - 发送方向：将发起端和完成端 UPLI 接口下各通道的请求、数据、响应打包为 TL Flit，发送给数据链路层。
  - 接收方向：从数据链路层接收 TL Flit，解包后还原为 UPLI 通道信号，分发给发起端或完成端 UPLI 接口。
- **UALink 数据链路层（DL）**
  - 发送方向：接收多个 TL Flit，添加头部与 CRC 校验，封装为 DL Flit 后发给物理层。
  - 接收方向：从物理层接收 DL Flit，剥离头部与 CRC，恢复为 TL Flit 上送给事务层。
- **UALink 物理层（PL）**
  - 发送方向：对 DL Flit 进行 FEC 编码，串行化后通过介质发送到对端。
  - 接收方向：接收串行信号，完成 FEC 解码，恢复出 DL Flit 上交数据链路层。

可以看出：

UPLI 是协议的“用户界面”，下层所有封装、转发、纠错，最终都是为了可靠完成一次 UPLI 事务。

UPLI 请求和响应的数据流图

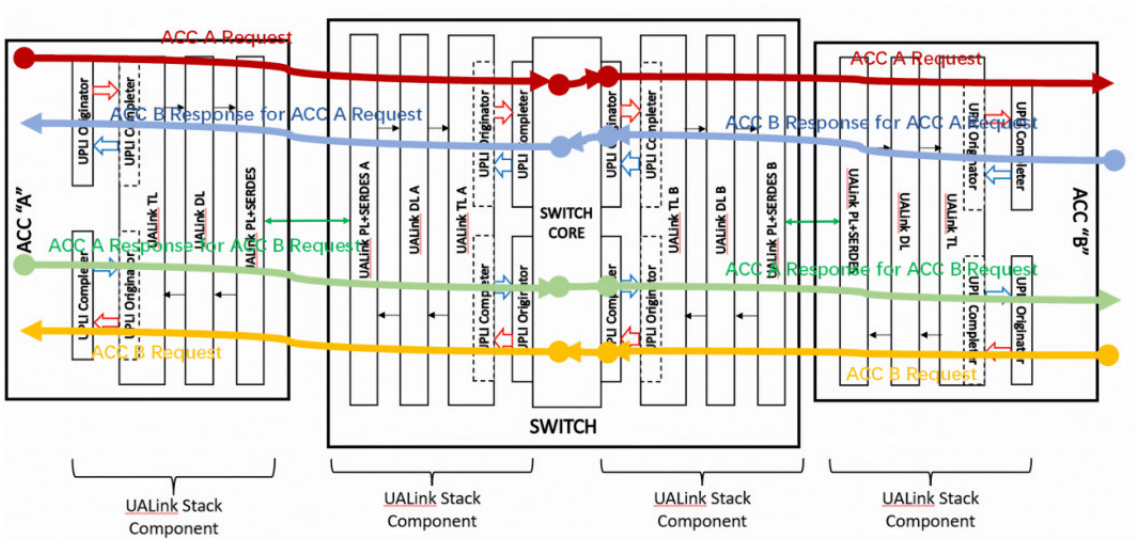


图142 加速器间端到端 UALink 联接

两个加速器之间的完整通信分为**请求路径与响应路径**，且完全对称。

- **请求路径（A→B）**
  - 加速器 A 的 **UPLI 发起端** 发出请求。
  - 请求到达交换机，交换机内部先由 UPLI 完成端接收，再由交换机内部的 UPLI 发起端转发。
  - 最终送到加速器 B 的 **UPLI 完成端**。
- **响应路径（B→A）**
  - 加速器 B 的 UPLI 完成端处理完请求，构造响应。
  - 响应沿相反路径经过交换机。
  - 最终回到加速器 A 的 **UPLI 发起端**。

交换机在整个过程中按照 UPLI 指令转发，根据 UPLI 的目的加速器决定出口，一跳一跳接力，最终完成源 加速器到目标 加速器 的读 / 写操作。

事务的端到端路由转发

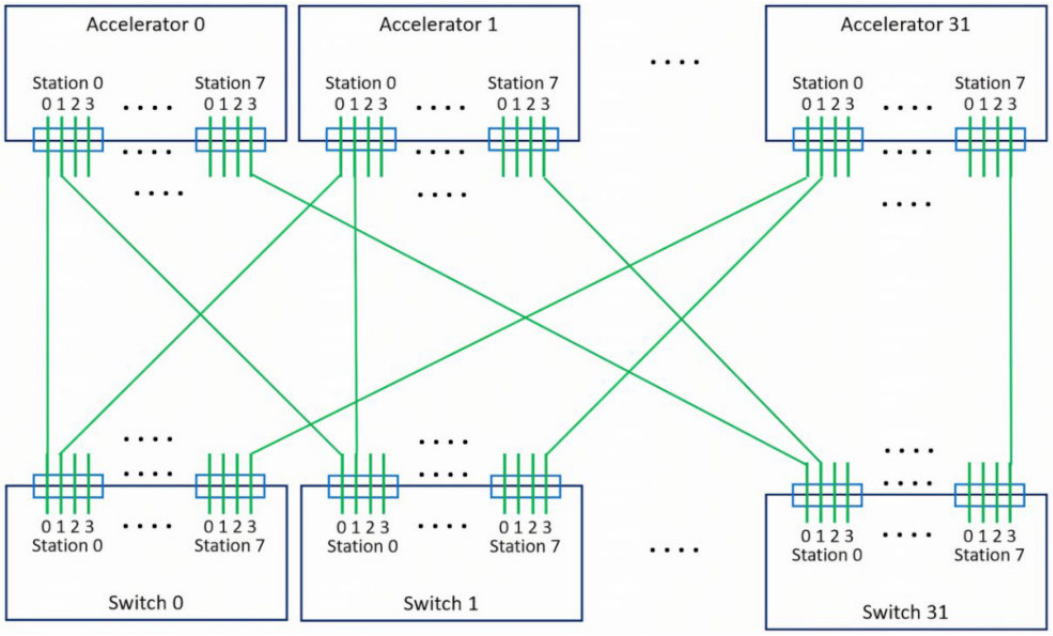


图143 32 加速器系统样例

一个 GPU 会拥有多个 UALink 端口（Port），每个端口对应一条独立链路，连接不同交换机,构成不同的交换平面。为了让请求与响应能在大规模集群中**精准送达**（包括源 GPU ID、源 Port 号，到目的 GPU ID 的请求可以精准送到目的 GPU，同时响应可以请准地送回源 GPU 和源 Port，而不会被源 GPU 的其他 Port 的 UALink 引擎处理），因此 UPLI 在通道中需要携带以下关键路由信息。

1. 路由信号定义

- **请求通道（Req）**
  - ReqSrcPhysAcclD：源加速器物理 ID
  - ReqDstPhysAcclD：目的加速器物理 ID
  - ReqPortID：端口 ID
- **读响应通道（RdRsp）**
  - RdRspSrcPhysAcclD
  - RdRspDstPhysAcclD
  - RdRspPortID
- **写响应通道（WrRsp）**
  - WrRspSrcPhysAcclD
  - WrRspDstPhysAcclD
  - WrRspPortID

其中：

源 ID 和目的 ID 在整个传输过程中保持不变，是交换机路由的核心依据。

Port ID 只在本地链路有效，每经过一段链路都会重新生成，仅用于节点内部调度。但是 Port ID 依然非常重要，如下图所示，对于源 GPU，拥有多个 UALink 以太接口，一个以太接口（协议里面称为 Station）对应一个 UALink 引擎，同时一个以太网接口可以工作在 1 个 800G，2 个 400G，4 个 200G 的模式，此时这个 800G、400G、200G 以太端口称为 Port。对 Port ID 标识的是 UPLI Req 对应的一个唯一的 UALink 引擎，以及引擎内的一个实例，联接一个唯一 Port，一个唯一的 UALink 交换平面。当目的 GPU 收到 UPLI 请求时，虽然 Port ID 已经变为内部收到此 UPLI 的 Port ID，但也标识了一个唯一的 UALink 交换平面，一个唯一的 UALink 引擎 + 实例。GPU 处理此 UPLI 时会记录此 Port ID，并在响应时使用此 Port ID，送回此 UALink 引擎 + 实例，以及此引擎 + 实例对应的唯一的 Port ID，对应的唯一的 UALink 交换平面。最终这个交换平面，也会将响应送回源 GPU，以及这个交换平面联接的唯一的源 Port ID，唯一的 UALink 引擎 + 实例。

2. 读请求流程（示例）

以加速器 0 访问加速器 31 为例：

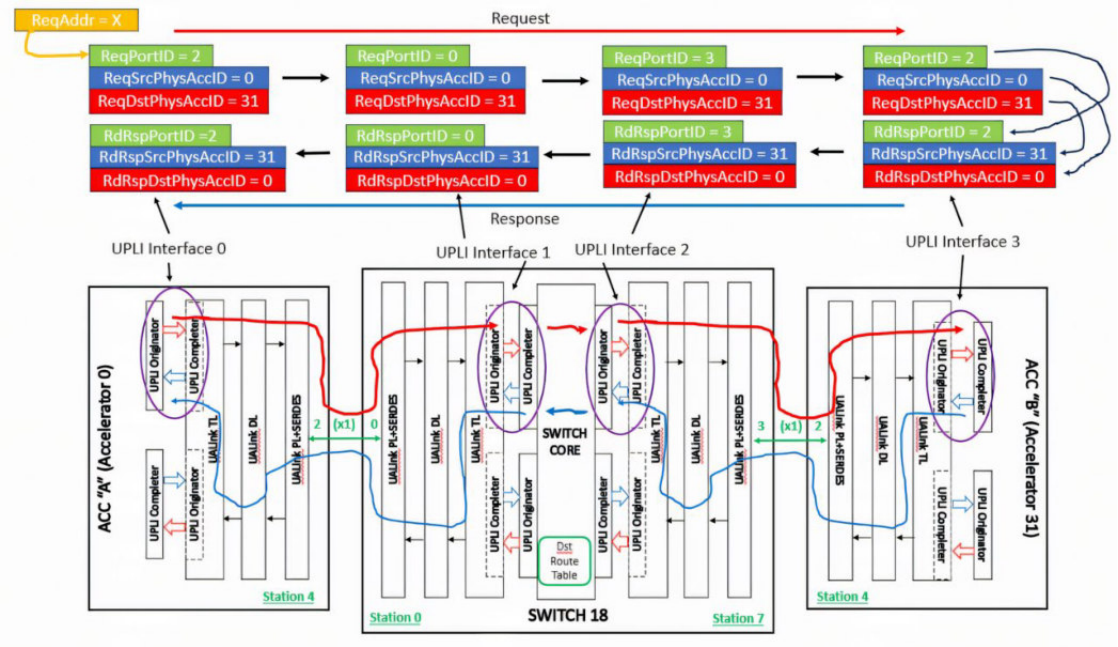


图144 端到端报文处理流程

- 为保证同一地址区域的访问有序，同一 256 字节区域的所有访问必须走同一条链路。
- 加速器 0 的 UPLI 发起端发出读请求：
  - ReqSrcPhysAcclD = 0
  - ReqDstPhysAcclD = 31
  - ReqPortID = 2（如根据地址区域选择，但是可以定义更复杂的选择机制，是各 GPU 的实现细节）
- 请求通过指定链路发送到交换机。

- 交换机根据进入的端口，**重新生成本地 PortID**，仅用于内部调度。
- 交换机根据 ReqDstPhysAcclD 查询路由表，确定应从哪个 Station、哪个端口发往目标加速器。
- 交换机将请求转发到目标端口，再次生成本地 Port ID。
- 请求进入加速器 31，根据入端口再次生成本地 PortID。
- 最终送到加速器 31 的 UPLI 完成端处理。

3. 读响应流程（示例）

- 目标加速器构造响应，使用**请求中保留的 Port ID** 直接路由到输出端口，**无需再查地址**。
- 响应沿原路反向传输，每经过一段链路，PortID 都会根据入端口重新生成。
- 最终回到源加速器的 UPLI 发起端。

写请求流程与读请求类似，区别在于：

- 写请求不仅包含地址，同时包含需要写的数据，数据通过 OrigData 发起端数据通道 发送。
- 写操作只返回不带数据的简单写响应。而前述的读操作要返回带数据的读响应。

UPLI 和 UPLI 之间的 流量控制

两个 UPLI 之间采用基于信用（Credit）的流量控制机制，保证链路与缓冲不会溢出。如下图所示，源加速器 A 和交换机入接口建立 UPLI 和 UPLI 之间的流控，交换机出端口和目的加速器 B 再建立 UPLI 和 UPLI 的流控。交换机入出接口之间的流控，是交换机内部实现。

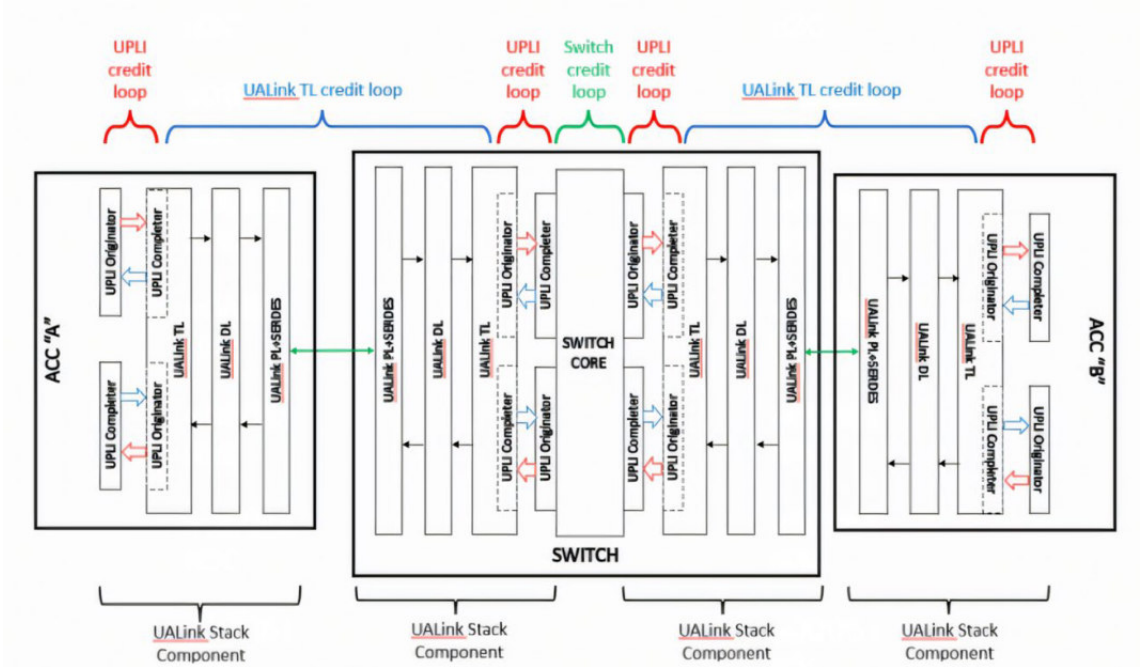


图145 流控机制

核心机制：

- **按端口、按通道**分配信用。
- 信用分为两类：
  - **池信用（Pool Credit）：**可跨虚拟通道共享使用。
  - **虚拟通道信用（VC Credit）：**绑定特定虚拟通道。

流程：

- **初始化：**发送端一开始无信用，接收端先发放初始信用，并通过 CreditInitDone 信号通知可以开始传输。
- **发送约束：**发送端必须同时拥有请求通道信用和数据通道信用，才能发起写操作。
- **信用返还：**接收端处理完一拍数据后，返还信用，并携带 VC 与 Pool 信息，发送端据此更新 VC 或 Pool 的可用信用。
- **端到端生效：**流量控制作用在整条路径的每一个 UPLI 接口，确保逐跳可靠、不拥塞、不丢包。

UPLI 事务与通道使用

UPLI 支持三类核心事务：

- 读事务
- 写事务
- 原子事务

所有事务都由 **UPLI 发起端发出，UPLI 完成端响应**。

1. 读事务

- 通过 **请求通道（Req）**发送一拍请求，包含：

  源 / 目的 ID、端口 ID、Tag、地址、长度、属性等。
- 完成端处理后，通过 **读响应 / 数据通道（RdRsp）**返回数据，支持 1~4 拍数据。
- 每拍携带：状态、偏移、总拍数、数据错误标志、最后一拍标志等。

2. 写事务

- 通过 **请求通道** 发送写请求。
- 同时通过 **发起端数据通道（OrigData）**发送 1~4 拍数据，包含：

  字节使能、偏移、数据错误标志、最后一拍标志。
- 完成端处理完毕后，通过 **写响应通道（WrRsp）**返回单拍响应，不带数据。

3. 原子事务

原子操作分为两类：

- AtomicR：先读出内存旧值返回，再原子修改内存。
- AtomicNR：只原子修改内存，不返回旧值。

原子操作支持常见语义，如加法、比较并交换等，通常需要 1~2 个操作数，由 UPLI 发起端随请求一同发送。

UPLI 的信号定义

1. UPLI 的信号组

UALink 协议层接口应按以下信号组进行组织：

- 公共信号 Common Signals
- UALink 协议层接口控制信号 UALink Protocol Level Interface Control signals
- 请求通道 Request Channel
- 读响应 / 数据通道 Read Response/Data Channel
- 写响应通道 Write Response Channel
- 发起端数据通道 Originator Data Channel

2. 公共信号 Common Signals

表18 Common Signals

Name	Size	描述
UPLIClk	1	接口时钟，由 SOC 通用逻辑提供。UALink 协议层接口信号与该时钟以及 UPLIReset_N、ClkReq 和ClkAck 信号同步。
UPLIReset_N	1	接口复位信号，低电平有效，其置位与撤销均与 UPLIClk 同步。

3. UPLI 事务 / 通道使用

事务可以是**读操作、写操作或原子操作**。这些事务从内存读取、向内存写入，或**以原子方式修改内存**，并可选择返回内存中的原始值。

一个 UPLI 事务由多个周期（也称为**拍 /beat**）组成。

根据 UPLI 事务类型（读、写、原子）的不同，这些拍会按照特定顺序，在 UALink 协议层接口的一组特定物理通道上传输：

- 请求通道

- 读响应 / 数据通道
- 写响应通道
- 发起端数据通道

事务由 **UPLI 发起端（Originator）**发出，并由 **UPLI 完成端（Completer）**处理完成。UPLI 完成端会通过**读响应 / 数据通道**或**写响应通道**其中之一向请求返回响应。具体使用哪个通道返回响应，取决于事务类型；对于原子操作，还额外取决于**原子操作是否返回修改前的内存初始值**。

读事务

- 每个 UPLI 事务由**请求通道**中的请求发起，通过请求通道上的有效信号（ReqVld）置高表示一个请求拍。
- 读请求拍包含与路由和请求标识相关的信息，例如：
  - 端口 ID（ReqPortID）
  - 源加速器与目的加速器物理 ID（ReqSrcPhysAcclD、ReqDstPhysAcclD）
  - 用于区分不同请求的标签（ReqTag）
  - 此外，请求拍还包含以下请求信息：
    - 地址（ReqAddr）
    - 命令类型（ReqCmd）
    - 待传输双字数量指示（ReqLen）
    - 请求属性（ReqAttr）及其他字段
- 读事务返回的数据通过**读响应 / 数据通道**，以数据拍的形式从完成端传送给发起端，同时携带读响应的附加信息，包括：
  - 读事务状态（RdRspStatus）
  - 当前返回的是第几个数据拍（RdRspOffset）
  - 事务中的总数据拍数（RdRspNumBeats）一个读响应可包含 1**~4 个数据拍**
  - 数据拍是否损坏（RdRspDataError）
  - 当前拍是否为最后一个传输拍（RdRspLast）

写事务

待写入的数据通过**发起端数据通道**，以 **1~4 个数据拍** 从发起端传送到完成端。每个数据拍包含：

- 端口 ID（OrigPortID），用于提供部分路由信息。
- **同一事务的第一个发起端数据拍必须与写请求出现在同一周期**。这使得发起端数据拍可以继承来自请求通道中写请求的其余路由与标识信息：
  - ReqSrcPhysAcclD
  - ReqDstPhysAcclD
  - ReqTag

- 以及事务中的数据拍总数（ReqNumBeats）

- 写请求拍还包含以下信息：
  - 字节使能，用于指示内存中需要更新的字节（OrigDataByteEn）
  - 当前发送的是第几个数据拍（OrigDataOffset）
  - 数据拍是否损坏（OrigDataError）
  - 当前拍是否为最后一个传输拍（OrigDataLast）

原子请求

原子操作分为两类：

- 返回**修改前内存初始值**并**原子修改内存**（称为 **AtomicR 命令**）
- 仅执行内存修改，不返回原值（称为 **AtomicNR 命令**）

所有原子命令（AtomicR 和 AtomicNR）都需要**1个或2个操作数**来控制内存的原子更新。例如：

- **原子加（Atomic Add）**需要1个参数，指定要原子加到内存位置的值
- **比较并交换（Compare-and-Swap）**需要 2 个参数：
  - 与内存位置初始比较的值
  - 若初始比较匹配，则写入内存位置的新值

4. Channel 的信号定义

UPLI 具体定义了请求通道 Request Channel、读响应 / 数据通道 Read Response/Data Channel、写响应通道 Write Response Channel、发起端数据通道 Originator Data Channel 的信号，以实现前述的读、写和原子事务。具体定义可以查询 UALink 规格书。下面具体使用 Request Channel 的定义，让大家理解一下定义的内容。

表19 请求通道信号

Name	Size (bits)	Driver	描述
请求通道信息与控制信号			
ReqVld	1	Originator	请求有效。该信号指示发起端正在该通道上呈现有效信息（一个 beat）。驱动请求通道的源加速器 ID 指示与该 TL Beat 关联的端口 ID，该 Beat 要在请求通道上呈现。
ReqPortID	2	Originator	请求地址空间标识符。对于所有发起端，ReqPortID 指示请求通道上的 TDM 周期。
ReqASI	2	Originator	请求地址空间标识符。对于不是 UPLI 写消息请求的任何请求，此字段指定特定地址空间的地址。对于 UPLI 写消息请求，此字段标识其他保留或特定于 UPLI 写消息的特定功能（UPLI 写消息的类型在 ReqMetadata 字段中指定）。
ReqAuthTag	64	Originator	请求授权标签。请求的授权标签。当授权处于活动状态时，发起端为此请求提供授权标签。当授权不活动时，物理加速器 ID 被置为无效。

Name	Size (bits)	Driver	描述
ReqSrcPhysAcclD	10	Originator	源加速器 ID。请求的源物理加速器 ID。源加速器是发起端（发起 UPLI 事务的加速器）内的物理加速器。
ReqDstPhysAcclD	10	Originator	请求的目的加速器 ID。UPLI 完成器的物理加速器 ID，该完成器将满足该请求。
ReqTag	11	Originator	请求标签。此标签用于唯一标识每个未完成的 UPLI 请求（在源加速器 UALink 端口内，所有与该请求关联的站都共享此标签）。
ReqNumBeats	2	Originator	事务中的数据拍数。在 UPLI 写消息请求中，发起端将此值设置为要传输的数据拍数。对于读、原子和 UPLI 写消息请求，传输的字节数是 (ReqNumBeats + 1)。此字段仅对原子和 UPLI 写消息请求有效。对于原子写请求（ReqCmd [5] = 1）和 UPLI 写消息请求，此字段用于确定要传输的数据拍数。对于 ReqCmd [5] = 0（原子读）的请求，此字段驱动为 0。
ReqAddr	57	Originator	对于不是 UPLI 写消息请求的任何请求，此字段指定请求的地址。对于 UPLI 写消息请求，此字段指定其他保留或特定于 UPLI 写消息的特定功能（UPLI 写消息的类型在 ReqMetadata 字段中指定）。除非 UPLI 写消息类型定义了此字段并在携带此消息的请求中指定，否则此字段为 0。
ReqCmd	6	Originator	请求命令。此字段指定此请求和此事务中要执行的操作的类型。
ReqLen	6	Originator	请求长度。对于不是 UPLI 写消息请求的任何请求，此字段指定要从 ReqAddr 地址开始读取的 64 位双字数量。对于 UPLI 写消息请求，传输器必须在一个事务中传输 4 字节的有效载荷，因此此字段驱动为 0。
ReqAttr	6	Originator	请求属性。对于不是 UPLI 写消息请求的任何请求，此字段指定与该请求关联的内存属性。对于 UPLI 写消息请求，此字段标识其他保留或特定于 UPLI 写消息的特定功能（UPLI 写消息的类型在 ReqMetadata 字段中指定）。
ReqMetadata	8	Originator	请求元数据。对于 UPLI 写消息请求，此字段指定特定 UPLI 写消息的类型。对于不是 UPLI 写消息请求的任何请求，此字段驱动为 0。
ReqPC	2	Originator	请求处理类别。此字段指定请求的处理类别。此值在交换机中用于路由。除了在源加速器 UALink 端口中驱动此值外，发起端在整个交换机中都不解释此值。
ReqPool	1	Originator	请求池。指定请求是使用池信用还是虚拟信道资源（0 = 池，1 = 虚拟信道）。
请求通道信用返回信号			
ReqCredVld	4	Completer	信用有效。当且仅当相应端口的信用被返回时，此信号置为高。信用返回信号不是时分复用（TDM）的；任何端口的信用都可以在任意周期返回。
ReqCredPool	2	Completer	信用池。指示是池信用还是虚拟信道（VC）信用被返回。如果 “0”，则返回的是池信用；如果 “1”，则返回的是虚拟信道信用。
ReqCredPortVC	2	Completer	指示为此请求返回的虚拟信道（VC）或池信用。如果 ReqCredPool 为 0，则此字段指示为此请求返回的池信用；如果 ReqCredPool 为 1，则此字段指示为此请求返回的虚拟信道（VC）。

Name	Size (bits)	Driver	描述
ReqCredPortVC2Num	3	Completer	指示为此请求返回的虚拟信道（VC）或池信用。如果 ReqCredPool 为 0，则此字段指示为此请求返回的池信用；如果 ReqCredPool 为 1，则此字段指示为此请求返回的虚拟信道（VC）。
ReqCredPortVC3Num	2	Completer	指示为此请求返回的虚拟信道（VC）或池信用。如果 ReqCredPool 为 0，则此字段指示为此请求返回的池信用；如果 ReqCredPool 为 1，则此字段指示为此请求返回的虚拟信道（VC）。
ReqCredPortVC	2	Completer	指示为此请求返回的虚拟信道（VC）或池信用。如果 ReqCredPool 为 0，则此字段指示为此请求返回的池信用；如果 ReqCredPool 为 1，则此字段指示为此请求返回的虚拟信道（VC）。
ReqCredPortNum	2	Completer	指示为此请求返回信用的端口号（最多 3 个）。仅当 ReqCredVld 指示的端口有效时，此字段才有效。
ReqCredPort2Num	3	Completer	指示为此请求返回信用的端口号（最多 3 个）。仅当 ReqCredVld 指示的端口有效时，此字段才有效。
ReqCredPort3Num	1	Completer	指示为此请求返回信用的端口号（最多 3 个）。仅当 ReqCredVld 指示的端口有效时，此字段才有效。
请求通道奇偶校验信号			
ReqVldParity	1	Originator	对每个请求有效信号 ReqVld 进行奇偶校验，每周期检查一次。
ReqAddrParity	1	Originator	对请求地址 ReqAddr 进行奇偶校验。仅当 ReqVld 置为有效时检查。
ReqParity	1	Originator	对以下信号进行奇偶校验：ReqASI、ReqAuthTag、ReqCmd、ReqLen、ReqAttr、ReqTag、ReqNumBeats、ReqVld、ReqSrcPhysAcclD、ReqDstPhysAcclD、ReqPortlD。仅当 ReqVld 置为有效时检查。
ReqCredVldParity	1	Completer	对每个信用有效信号 ReqCredVld 进行奇偶校验，每周期检查一次。
ReqCredParity	1	Completer	对以下信号进行奇偶校验：ReqCredVld、ReqCredPortNum、ReqCredPort2Num、ReqCredPort3Num、ReqCredPortVC、ReqCredPortVC2Num、ReqCredPortVC3Num、ReqCredPool。仅当 ReqCredVld 非零时检查。

本章小结

本章重点讲解了 UALink 最顶层的 **UPLI 接口**，它是整个协议的 “用户入口”，核心内容可总结为：

- **UPLI 采用主从模式：**每个 GPU 同时具备发起端与完成端。
- **四条通道分工明确：**Req、OrigData、RdRsp、WrRsp，与 AXI 思想相似。
- **请求与响应路径对称**，经过交换机逐跳转发。
- **路由依靠源 / 目的加速器 ID + 端口 ID，ID 全局不变**，PortID 本地有效。

- **基于信用的流量控制**，保证可靠无拥塞传输。
- **支持读、写、原子三类事务**，覆盖加速器间内存交互全部需求。

事务层

UALink 事务层（TL 层）主要完成两类数据转换工作：

- **发送方向（Tx）**

将两个 UPLI 接口传入事务层的 UPLI 拍（Beat），进行格式转换，封装成事务层微片（TL Flit），并从发送方向的 TL Flit 通道输出。

- **接收方向（Rx）**

将从接收方向的 TL Flit 通道收到的事务层微片（TL Flit），解析还原为 UPLI 拍（Beat），并输出到两个北向 UPLI 接口，供上层模块使用。

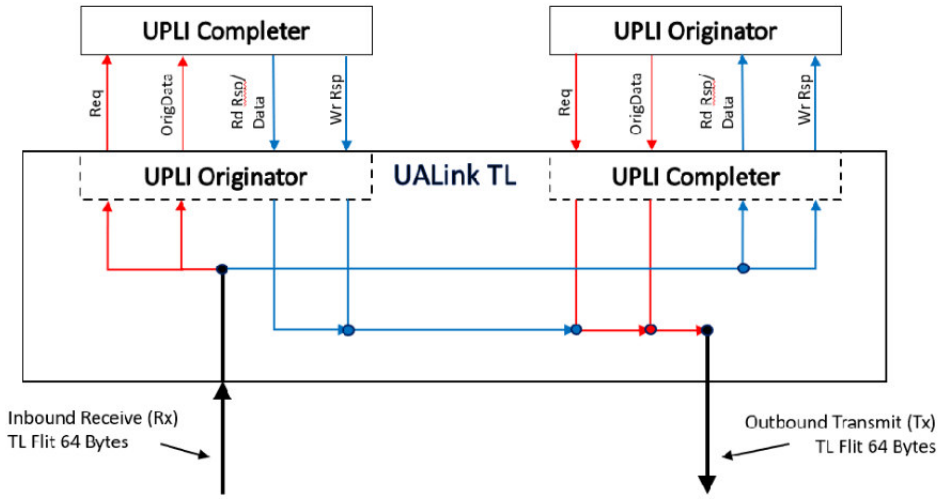


图146 TL FLIT 与 UPLI 的连接

事务层 FLIT（TL Flit）格式与时序

每个 64 字节的事务层 FLIT（TL Flit）被划分为高 32 字节与低 32 字节两个半 FLIT（Half Flit）；同时，该 64 字节事务层 FLIT 也可划分为 16 个 4 字节扇区（Sector）。

64-byte TL Flit															
Upper TL Half-Flit								Lower TL Half-Flit							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

图147 TL Flit 组织架构

事务层 Half Flit 分为四种类型：**控制 Half Flit、数据 Half Flit、消息 Half Flit、认证标签（AuthTags）Half Flit**。本文重点介绍最常用的**控制 Half Flit 和数据 Half Flit**，消息与认证标签相关定义可查阅 UALink 规格书。

控制 Half Flit 用于编码以下信息：

- 请求：读、写、原子读（AtomicR）、原子非读（AtomicNR）
- 读响应：不含关联数据（注：AtomicR 请求会产生读响应）
- 写响应：注：AtomicNR 请求会产生写响应
- 流量控制 / 空操作（NOP）信息

请求可使用 **4 个或 2 个扇区字段** 进行编码。

响应可使用 **2 个或 1 个扇区字段** 进行编码。

流量控制（FC）信息与 NOP 指示各使用 **1 个扇区** 进行编码。

控制 Half-Flit 中所有未使用的扇区必须为 NOP 字段。

数据 Half-Flit 用于对以下信息进行编码：

- 读响应数据（读与 AtomicR 操作）
- 写数据（写、全写、UPLI 写消息、AtomicNR 操作）
- 字节使能（写与 AtomicR/AtomicNR 操作）
- 原子操作数（AtomicR 与 AtomicNR 操作）

UPLI 接口支持 1/2/3/4 拍 64 字节数据传输，分别对应 2/4/6/8 个 32 字节事务层数据 Half-Flit。

某个 Half-Flit 是控制 Half-Flit、还是数据 Half-Flit，**并不在 Half-Flit 内部携带标识**，而是通过接口上 Half-Flit 的**时序与顺序**推导得出。

若某控制 Half-Flit 要求携带数据 Half-Flit（包含读响应字段、写请求字段或 AtomicR/AtomicNR 请求字段，下文统称为 “数据请求字段”），则后续 Half-Flit 将被解析为数据 Half-Flit，直至满足数据请求字段所需的 Half-Flit 数量。

数据 Half-Flit 的顺序与控制 Half-Flit 中数据请求字段的顺序一致，从低位开始。

一个特殊处理：若控制 Half-Flit 所对应的最后一个数据 Half-Flit 本应出现在事务层 FLIT 的低 Half-Flit 位置，则该最后一个数据 Half-Flit 将交换到该事务层微片的高 Half-Flit 位置，而低 Half-Flit 则被解析为下一个控制 Half-Flit。

下面是一些 TL-FLIT 报文的举例，用来理解事务层的封装格式，并且这个格式需要包含 UIPLI 指令中所有的控制和数据信号。

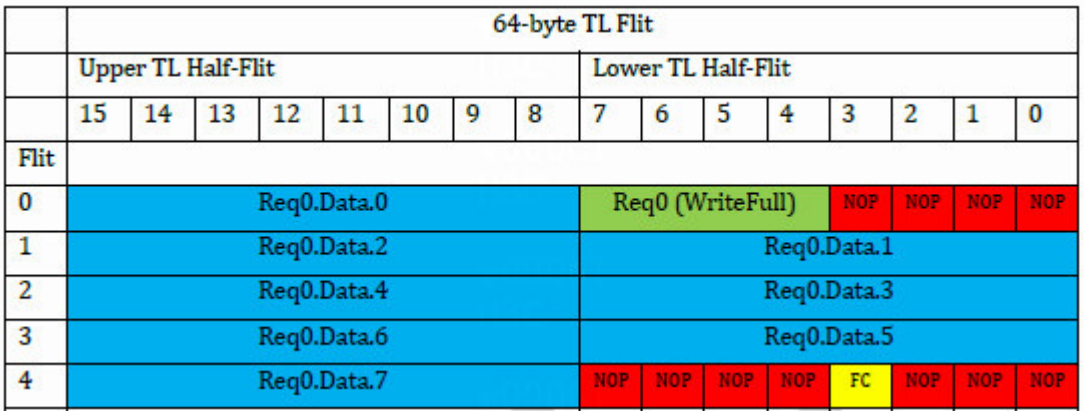


图148 一个 Write Full 256 字节请求和一个流控控制 Half FLIT

地址缓存技术 —— 发送 / 接收数据流与压缩缓存

地址缓存是 UALink 事务层用于**提升传输效率、降低链路带宽开销**的关键优化机制，通过缓存通信地址实现请求报文压缩，分为**发送端地址缓存（Tx Address Cache）和接收端地址缓存（Rx Address Cache）**，二者成对工作、严格同步。

1. 基本工作原理

发送方向的请求会先进入请求队列，然后查询**发送端地址缓存**，判断该地址是否已经发送过，并且已经缓存在对端 UALink 事务层的**接收端缓存**中。

地址缓存属于**可选功能**：

- 如果不实现该功能，链路只使用非压缩请求，并标识不加载接收端缓存。
- 如果实现地址缓存，则可以使用压缩请求，大幅减少链路上传输的地址位数与冗余信息。

工作流程：

- **缓存命中**

  发送端缓存查询命中 → 直接使用**压缩请求**，省略大量地址位和其他信息。

- **缓存未命中但地址可缓存**

  先将地址加载到发送端缓存，同时发送**非压缩请求**，并通知对端将地址加载到接收端缓存。加载完成后，后续相同地址的请求即可使用压缩请求。

- **不加载缓存**

  若发送端缓存无法加载，或设计上选择不缓存该地址，则直接发送**非压缩请求**，并指示对端不加载接收端缓存。

收发缓存同步关系

- 接收端缓存**完全由发送端控制**，与发送端缓存严格同步。

- 发送端缓存可随时清空或失效；清空后，在重新加载前不再发送压缩请求，接收端缓存也会同步重新加载。

2. 发送端与接收端地址缓存特性

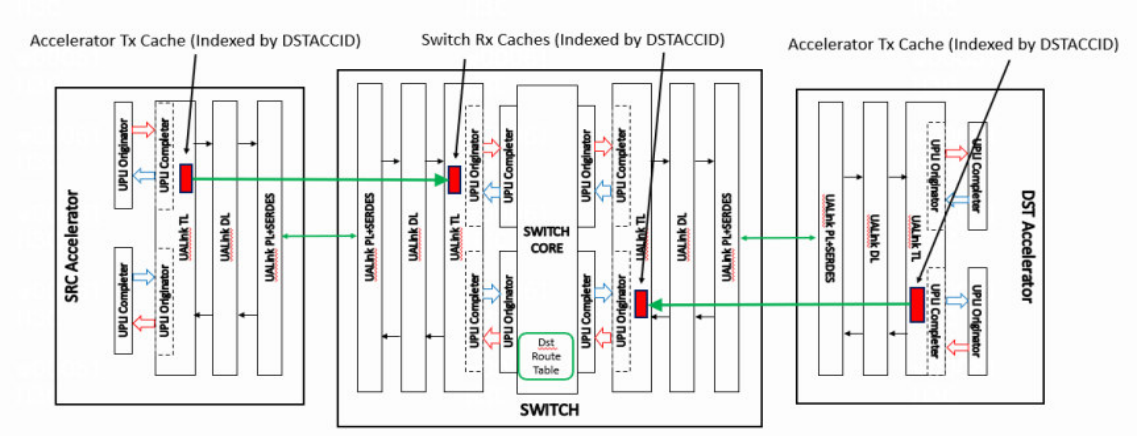


图149 加速器发送和交换机接收地址 Cache

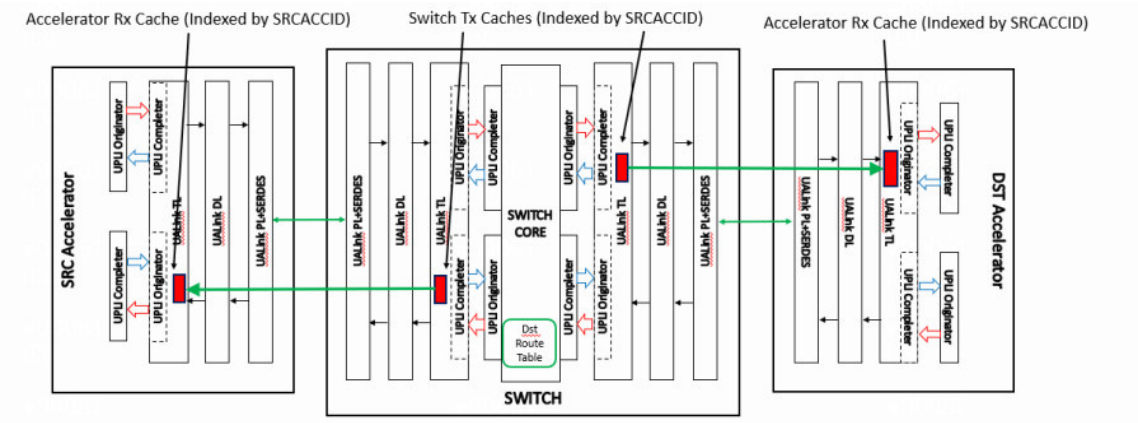


图150 交换机发送和加速器接收地址 Cache

收发端缓存支持两个加速器之间**同时 4 条活跃流，源加速器到交换机接收，交换机发送到目的加速器，目的加速器到交换机接收，交换机发送到源交换机**，具备以下特性：

- 一对加速器之间的缓存器最多为（且典型为）**4 组**，如图中绿线与绿线连接的两个红色矩形，标识一对同步的缓存器组，同步方向同绿线箭头方向。
- 缓存规模：一个缓存器最多可管理 **1024 个缓存行**。
- 缓存处理性能：缓存每个周期至少可处理**一次表项查询或更新**。
- 缓存索引方法：发送端与接收端缓存行，根据 UPLI 请求通道中的 **SRCACCID（源加速器 ID）与 DSTACCID（目的加速器 ID）**字段，确定缓存器，以及内存地址的余数计算的结果找到缓存行号（如 HASH 计算的余数）。
- 缓存索引同步方法：一对收发缓存（Tx 缓存 / Rx 缓存）的索引（缓存行号）由 **Tx 缓存** 控制。

3. 收发地址缓存同步规则（时序与约束）

为保证发送端与接收端缓存状态完全一致，必须遵循以下时序规则：

- **非压缩请求可随时发送**

  无论缓存是否有效，都允许发送 CLOAD=0 的非压缩请求。

- **加载类请求必须确保缓存可用**

  发送 CLOAD=1（加载对端缓存）的非压缩请求前，对应索引（缓存行）可用，即 Cache 路（CWAY）的表项必须已加载，且在请求“已发出”前不能被其他请求覆盖。

- **压缩请求只能在缓存命中时发送**

  只有满足以下两个条件，才能发送压缩请求：

  - 地址在发送端缓存**命中**；
  - 对应缓存表项在“已发出”前**不会被其他请求覆盖**。

- **请求按地址低位到高位顺序排列**

  控制 Half-Flit 内的请求根据请求的到达顺序，按**低字节到高字节**排列，保证接收端与发送端对缓存按正确的顺序处理。

- **请求 “已发出” 的定义**

  当请求被放入控制 Half-Flit，并确定在发送通道的最终位置时，视为**已发出（issued）**。

- **接收端缓存更新规则**

  - 收到 CLOAD=1 的非压缩请求时，根据 **CWAY** 更新对应表项，即非压缩请求中的内存地址。
  - 更新结果必须**立即对后续请求可见**（要么写操作下一周期直接生效，要么通过 CAM 逻辑提前提供新值）。
  - 接收端缓存表项**不能主动失效**，只能被新的 CLOAD=1 请求覆盖替换。

4. 缓存刷新机制

发送端与接收端缓存的刷新逻辑保持一致，由 CLOAD=1 与 **CWAY** 共同控制：

- **发送端缓存（Tx Cache）**

  发送带 CWAY 的 CLOAD=1 非压缩请求时，刷新该 CWAY 指定的表项；

  该表项保持有效，直到下一次相同 CWAY、CLOAD=1 的请求到来。

- **接收端缓存（Rx Cache）**

  收到带 CWAY 的 CLOAD=1 非压缩请求时，刷新该 CWAY 指定的表项；

  表项保持有效，直到被相同 CWAY 的新请求覆盖。

事务层流量控制（TL Flow Control）

UALink 事务层（TL）之间的流量控制，通过以下两类信用（Credit）机制进行管理：

- 针对**控制 Half Flit** 中请求字段与响应字段的信用；
- 针对**成对携带数据的数据 Half Flit** 的信用（包括读响应数据、写数据、原子操作数）。

一个数据信用（Data Credit）会预留一个 64 字节的数据缓冲区。

信用分为两种类型：

- **池信用（Pool Credit）：** 可用于**任意虚拟通道**上的请求、响应或数据。
- **虚拟通道信用（Virtual Channel Credit）：** 仅用于与 4 个虚拟通道中某一特定通道绑定的请求、响应或数据。

1. 流量控制字段信号

流量控制字段包含四类信用信号，如下表所示：

信号名称	描述
Request CMD	用于未压缩或压缩的 TL 控制 Half-Flit 中请求的信用。
Response CMD	用于未压缩或压缩的读响应（不含数据）或压缩写响应的信用。
Request Data	用于 64 字节数据缓冲区的信用，用于存放写、全写（WriteFull）请求、原子读 / 原子非写（AtomicR/AtomicNR）命令及 UPLI 写消息的数据。字节使能由关联的请求或数据缓冲区缓冲，因此不占用独立的请求或数据信用。
Response Data	用于 64 字节数据缓冲区的信用，用于存放读响应数据（不含字节使能）。

2. 信用容量与字段位宽

为了支持最坏情况下的写例程（Write Example），流量控制字段必须至少能返回 20 个请求数据信用和 5 个 CMD 信用。因此：

- **Request Data 和 Response Data** 信用信号各为 **8 位**：
  - 5 位用于返回信用数量（最多 31 个，满足至少 20 个的要求）；
  - 2 位用于标识虚拟通道；
  - 1 位用于标识是池信用还是虚拟通道信用。
- **Request CMD 和 Response CMD** 信用信号各为 **6 位**：
  - 3 位用于返回信用数量（最多 7 个，满足至少 5 个的要求）；
  - 2 位用于标识虚拟通道；
  - 1 位用于标识是池信用还是虚拟通道信用。

四类信号共占用 28 位，加上高 4 位的字段类型（FTYPE）标识，共同组成一个 32 位的单扇区流量控制字段。

表20 流量控制 / 空操作字段（Flow Control/NOP Field）

名称	大小（位）	位置	描述（中文）
FTYPE	4	[31:28]	字段类型。该信号值为 0 时，表示此字段为流量控制 / 空操作（Flow Control/NOP）字段。
ReqCmd	6	[27:22]	请求命令。返回请求命令信用（即来自 UPLI 读 / 写 / 原子请求通道的请求信用）。字段格式为 “tvvccc”，其中： • t = 0 表示池信用，t = 1 表示虚拟通道信用（仅当 t=1 时，vv 有效） • vv = 信用所属的虚拟通道 • ccc = 待返回的信用数量（0 到 7）。
RspCmd	6	[21:16]	响应命令。返回响应命令信用（即来自 UPLI 读响应（不含数据）和写响应通道的响应信用）。字段格式为 “tvvccc”，其中： • t = 0 表示池信用，t = 1 表示虚拟通道信用（仅当 t=1 时，vv 有效） • vv = 信用所属的虚拟通道 • ccc = 待返回的信用数量（0 到 7）。
ReqData	8	[15:8]	请求数据。返回请求数据信用（即来自 UPLI 操作数 / 数据通道的信用）。字段格式为 “tvvcccc”，其中： • t = 0 表示池信用，t = 1 表示虚拟通道信用（仅当 t=1 时，vv 有效） • vv = 信用所属的虚拟通道 • cccc = 待返回的信用数量（0 到 31）。
RspData	8	[7:0]	响应数据。返回响应数据信用（即来自 UPLI 读响应 / 数据通道的信用）。字段格式为 “tvvcccc”，其中： • t = 0 表示池信用，t = 1 表示虚拟通道信用（仅当 t=1 时，vv 有效） • vv = 信用所属的虚拟通道 • cccc = 待返回的信用数量（0 到 31）。
TOTAL BITS	32	1 扇区	—

四类信用信号可独立选择表示池信用或虚拟通道信用。

3. 信用初始化与分配

在初始化阶段，事务层（TL）会为每类信用分配初始信用，这些信用可以是：

- **池信用（Pool Credits）：** 可用于任意请求、响应或数据。
- **虚拟通道信用（Virtual Channel Credits）：** 仅用于与该信用关联的虚拟通道。

一旦初始信用完全释放，TL 即可开始发送请求、响应或数据。每个请求或响应都带有一个 “池（Pool）” 信号：

- 当 Pool 信号为 1 时，表示使用池信用；
- 当 Pool 信号为 0 时，表示使用虚拟通道信用，同时在请求或响应中携带虚拟通道字段（VCHAN）。

接收端 TL 会记录请求或响应中的虚拟通道和池信号，并在流量控制字段中使用这些值，将信用归还给发起端。

数据或原子操作数的信用会直接继承对应请求或响应的池和虚拟通道值，这些值在数据 Half Flit 中不会单独指示。

4. 空操作字段（NOP Field）

当流量控制字段返回四类信用的数量均为 0 时，该字段即为 **空操作字段（NOP Field）**，其值为 x00000000。

事务层控制字段比特定义

1. TL 控制 Half-Flit（TL Control Half-Flit）字段类型

在一个 TL 控制 Half-Flit（Control Half-Flit）中，可包含以下几种字段类型，每种类型占用固定数量的 4 字节扇区（Sector）：

字段类型	占用扇区数
未压缩请求（Uncompressed Requests）	4 个扇区
未压缩响应（Uncompressed Responses）	2 个扇区
压缩请求（Compressed Request）	2 个扇区
单拍读响应的压缩响应（Compressed Response for Single-Beat Read）	1 个扇区
写或多拍读响应的压缩响应（Compressed Response for Write or Multi-Beat Read Response）	1 个扇区
流量控制 / 空操作指示（Flow Control/NOP Indication）	1 个扇区

每个字段的高 4 位用于标识**字段类型（Field Type）**，共支持 16 种可能的字段类型，同时也隐含了该字段内部包含的扇区数量。

2. TL 控制 Half-Flit 字段类型编码表

下表列出了控制 Half-Flit 字段类型高 4 位的合法编码：

表21 TL 控制 Half-Flit 消息类型值

字段类型高 4 位值	消息类型（Message Type）
0x0	流量控制池 / 空操作指示（Flow Control Pool/NOP Indication）
0x1	未压缩请求（Uncompressed Request）
0x2	未压缩响应（Uncompressed Response）
0x3	压缩请求（Compressed Request）
0x4	单拍读响应的压缩响应（Compressed Response for Single-Beat Read Response）
0x5	写或多拍读响应的压缩响应（Compressed Response for Write or Multi-Beat Read Response）

下面重点介绍 0x1 未压缩请求和 0x2 未压缩响应的报文格式定义，便于理解事务层报文的格式定义。其他事务层报文的格式定义，请参考协议的具体介绍。从 TL 控制 Half FLIT 的格式定义中，可以看出其 4 或 2 个 Sector（16-Byte 或 8-Byte）中包含了对应 UPLI 指令中所有的内容。

3. 非压缩请求字段

Name	Size (bits)	Position	Description
FTYPE	4	[127:124]	字段类型。该信号值为 0x01，表示该字段是一个非压缩请求字段（Uncompressed Request Field）。
CMD	6	[123:118]	命令。该信号携带此请求对应的 UPLI ReqCmd 信号值。
VCHAN	2	[117:116]	虚拟通道。该信号携带此请求对应的 UPLI ReqVC 信号值。
ASI	2	[115:114]	地址空间标识符。该信号携带此请求对应的 UPLI ReqASI 信号值。
TAG	11	[113:103]	标签。该信号携带此请求对应的 UPLI ReqTag 信号值。
POOL	1	[102]	池。该信号指示事务层（TL）是使用池信用（Pool Credit）还是虚拟通道信用（Virtual Channel Credit）来发送此请求。该信号的值与 UPLI 接口上的 ReqPool 信号无关（TL 信用管理独立于 UPLI 信用管理）。
ATTR	8	[101:94]	属性。该信号携带此请求对应的 UPLI ReqAttr 信号值。
LEN	6	[93:88]	长度。该信号携带此请求对应的 UPLI ReqLen 信号值。
METADATA	8	[87:80]	元数据。该信号携带此请求对应的 UPLI ReqMetadata 信号值。
ADDR	55	[79:25]	地址。该信号携带此请求对应的 UPLI ReqAddr [56:2] 信号值。这是一个双字对齐的地址。

Name	Size (bits)	Position	Description
SRCACCID	10	[24:15]	源加速器 ID。该信号携带此请求对应的 UPLI ReqSrcPhysAccID 信号值。
DSTACCID	10	[14:5]	目的加速器 ID。该信号携带此请求对应的 UPLI ReqDstPhysAccID 信号值。
CLOAD	1	[4]	缓存加载。当该信号为 '1' 时，表示接收端事务层（Rx TL）应将 ReqAddr [56:20]（ADDR [54:18]）加载到 Rx 地址缓存中，该缓存位于由 CWAY 指定的路（Way），以及由 SRCACCID [9:0]（用于加速器上的 Rx 缓存）或 DSTACCID [9:0]（用于交换机上的 Rx 地址缓存）指定的行（Row）。当该信号为 '0' 时，Rx 地址缓存不加载。
CWAY	2	[3:2]	缓存路。该信号指定要加载的地址缓存中的路（Way），仅当 CLOAD='1' 时有效。
NUMBEATS	2	[1:0]	拍数。该信号指示为此请求传输的数据拍数（Beats）。当 CMD [5] 为 '1' 时，用于传输原子操作数、写数据或全写（Write Full）数据，或用于任何厂商自定义命令。
TOTAL BITS:	128	4 Sectors	

4. 非压缩响应字段

Name	Size (bits)	Position	Description
FTYPE	4	[63:60]	字段类型。该信号值为 0x02，表示该字段是一个非压缩响应字段（Uncompressed Response Field）。
VCHAN	2	[59:58]	虚拟通道。该信号携带此响应对应的 UPLI RdRspVC 或 WrRspVC 信号值（由读 / 写信号决定）。
TAG	11	[57:47]	标签。该信号携带此响应对应的 UPLI RdRspTag 或 WrRspTag 信号值（由读 / 写信号决定）。
POOL	1	[46]	池。该信号指示事务层（TL）是使用池信用（Pool Credit）还是虚拟通道信用（Virtual Channel Credit）来发送此响应。该信号的值与 UPLI 接口上的 ReqPool 信号无关（TL 信用管理独立于 UPLI 信用管理）。
LEN	2	[45:44]	长度。该信号携带 UPLI RdRspNumBeats 信号值（由 UPLI 读响应管理器决定）。该字段仅对多拍读响应有意义；单拍读响应和所有写响应的长度始终为 1 拍，因此这些字段应设置为 b'00'。

Name	Size (bits)	Position	Description
OFFSET	2	[43:42]	偏移。该信号携带 UPLI RdRspOffset 信号值，用于单拍读响应。对于无数据返回的写响应（写响应）和多拍读响应（TL 负责重建总线传输），该信号应设置为 b'00'。
STATUS	4	[41:38]	状态。该信号携带 UPLI RdRspStatus 或 WrRspStatus 信号值（由读 / 写信号决定）。
RD/WR	1	[37]	读 / 写指示符。该信号指示该响应是读响应（读、AtomicNR 请求导致写响应，注意：AtomicR 请求导致读响应）还是写响应。
LAST	1	[36]	最后。该信号仅对读响应有效，携带 UPLI RdRspLast 信号值。对于写响应，该信号应设置为 b'0'。
SRCACCID	10	[35:26]	源加速器 ID。该字段携带 UPLI RdRspSrcPhysAccID 或 WrRspSrcPhysAccID 信号值，具体取决于响应是读、写、全写，还是 AtomicR/AtomicNR 请求（由 TAG 标识）。注意：响应中的 SRCACCID 值（如果携带）应等于请求中的 DSTACCID 值，以确保响应被路由到正确的请求方。SRCACCID 字段在功能上不是必需的，实现可以选择不在 TL 和 / 或交换机上携带该字段。当信号不包含准确值时，应在相应的 UPLI 接口上驱动为零。
DSTACCID	10	[25:16]	目的加速器 ID。该字段携带 UPLI RdRspDstPhysAccID 或 WrRspDstPhysAccID 信号值，具体取决于响应是读、写、全写，还是 AtomicR/AtomicNR 请求（由 TAG 标识）。注意：响应中的 DSTACCID 值应等于请求中的 SRCACCID 值，这用于确保响应被路由回最初发出请求的加速器。
SPARE(S)	16	[15:0]	该字段格式中有 16 位未分配。
TOTAL BITS:	64	2 Sectors	

数据链路层

如图所示，数据链路层位于事务层与物理层之间，下行可以将来自事务层的 TL Flit 打包为 640 字节数据链路层 DL FLIT，供物理层使用。同理，上行可以将物理层的 640 字节 DL FLIT 解包为事务层 TL FLIT。此外，数据链路层还在链路伙伴之间提供消息服务，此类消息在数据链路层发起并终结，不需要传递给事务层。

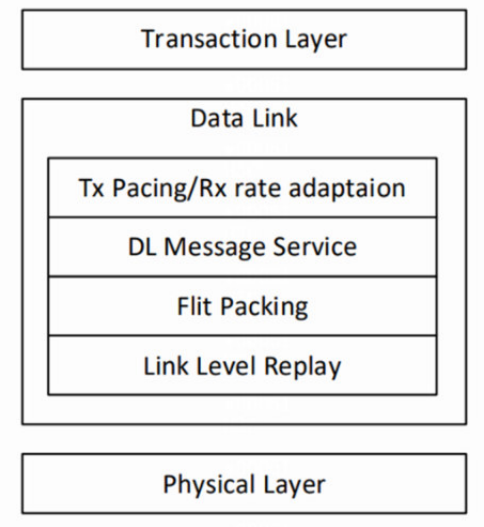


图151 数据链路层架构

DL FLIT 的格式

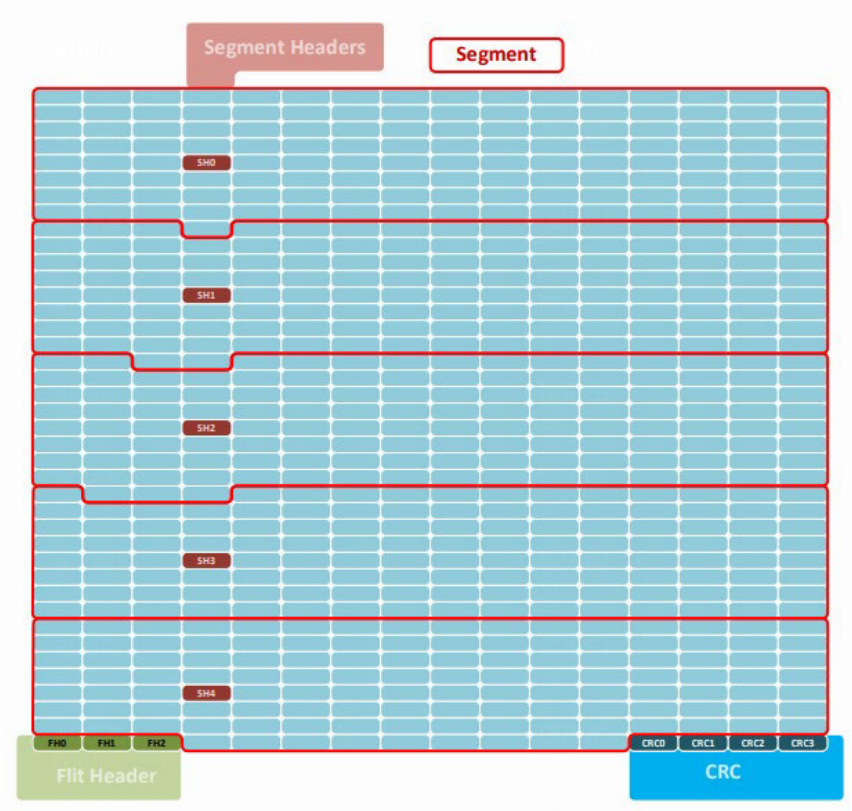


图152 数据链路层 640 字节 DL FLIT 格式定义

如图所示，共 640 字节，其中：

- 绿色 FH[2:0]：为 FLIT Header，共 3 字节，定义如下：

显式序号 FLIT 头（Explicit Sequence Number Flit Header）：显式序号 FLIT（简称 “显式 FLIT”）的头格式如下所示。其中包含完整的 9 位序号，不包含应答（Ack）或重传请求（Replay Request）指示信息。

Field	Bit	Description
Op	23:21	当 payload = 0（NOP）时： 0b000：空操作FLIT（NOP Flit） 其他值：保留 当 payload = 1 时： 0b000：净荷FLIT的原始传输（"org"） 0b001：净荷FLIT的重传（"rpy"） 其他值：保留
payload	20	1：净荷FLIT（payload Flit） 0：空操作FLIT（NOP Flit）
reserved	19:17	保留（Reserved）
flitSeqNo	16:8	FLIT序号（Sequence number of Flit），为完整的 9 位值
reserved	7:0	保留（Reserved）

命令 FLIT 头（Command Flit Header），应答（Ack）或重传请求（Replay Request）。其中包含待应答或未应答 FLIT 所对应的完整 9 位序号，同时还包含 FLIT 序号 flitSeqNo 的低 3 位，记为 flitSeqLo。

Field	Bit	Description
op	23:21	0b010：应答（Ack） 0b011：标准重传请求（Standard Replay Request "rpy"） 其他值：保留（Reserved）
payload	20	1：净荷FLIT（payload Flit） 0：空操作FLIT（NOP Flit）
ackReqSeq	19:11	待应答的应答FLIT的完整序号，或重传请求的序号。该值为完整的 9 位值。
flitSeqLo	10:8	FLIT序号的低 3 位。 注：由于Header空间有限，无法同时表示确认接收或需要重传的SeqNo，以及正在发送的SeqNo，所以对正在发送的SeqNo使用SeqLo来表示。
reserved	7:0	保留（Reserved）

- 红色 SH[4:0]：为 Segment Header，共 5 个，每个 Header 1 个字节。SH 用于描述从本段起始的事务层（TL）FLITS 序列。一个事务层 FLIT 序列中最多可包含两个 64 字节的 TL FLIT，Segment 净荷

长度可小于 128 字节，且部分段可包含备选扇区，因此部分事务层 FLIT 序列会跨段延续至下一段。定义如下

Field Name	Position	Description
DLAltSector	[0]	DL 备选扇区（DL Alternative sector） 0b – 段中无备选扇区 1b – 段中存在 DL 备选扇区
Reserved	[1]	保留（Reserved）
Message[0]	[3:2]	事务层FLIT 0（TL Flit [0]）消息位指示符 若事务层FLIT 0（TL Flit [0]）不存在，则为保留位；否则为消息位指示符
TL Flit[0]	[4]	事务层FLIT 0（TL Flit [0]）存在标识 0b – 事务层FLIT 0（TL Flit [0]）不存在 1b – 事务层FLIT 0（TL Flit [0]）存在
Message[1]	[6:5]	事务层FLIT 1（TL Flit [1]）消息位指示符 若事务层FLIT 1（TL Flit [1]）不存在，则为保留位；否则为消息位指示符
TL Flit[1]	[7]	事务层FLIT存在标识 0b – 事务层FLIT 1（TL [1]Flit）不存在 1b – 事务层FLIT 1（TL [1]Flit）存在

- CRC：循环冗余校验，共 4 个字节
- Segment payload：净荷，共 628 字节，每个端的净载荷字节数如下：

Segment Header	Number of payload Sectors	Number of payload bytes	Half Segment Sector ranges
SH0	32 Sectors	128–bytes	[0:15], [16:31]
SH1	32 Sectors	128–bytes	[32:47], [48:63]
SH2	32 Sectors	128–bytes	[64:79], [80:95]
SH3	31 Sectors	124–bytes	[96:111], [112:126]
SH4	30 Sectors	120–bytes	[127:142], [143:156]

DL 层有几个重要的机制，包括发送端速率调节、LLR 下面重点介绍一下：

链路层重传

链路层重传机制用于确保：在出现物理层 FEC 无法纠正的比特错误时，仍能保证 DL 微片按序可靠投递。发送端会保留所有净荷 FLIT（Payload Flit）（不含空操作微片 NOP）的副本，直至接收端给出肯定应答。未应答的 FLIT 存储在 TxReplay 缓冲区（发送重传缓冲区）中。

TxReplay 缓冲区的大小必须足够覆盖链路的往返时延（RTT），否则链路无法满带宽运行。

若 TxReplay 缓冲区已满（正在等待肯定应答 ACK），则不得发送新的 DL 净荷 FLIT，转而发送 NOP FLIT 替代。

若 TxReplay 缓冲区已满或处于重传状态，数据链路层（DL）需向事务层（TL）施加反压（back pressure），不再接收任何新的 TL FLIT，并确保已接收的 TL FLIT 不丢失。

PCS 接收端在将 FLIT 转发至 DL 之前先执行 FEC 纠错，仅通过 FEC 纠错的 FLIT 才会转发至 DL。

CRC 校验在 DL 层执行：若 CRC 校验失败，则该 DL 微片被判定为无效并丢弃。若 CRC 校验正确，则执行以下操作之一：

- 当接收端判定接收序号失序时，通过重传请求（Replay Request）起一次标准重传；
- 当接收端判定接收序号有序时，安排发送应答（Ack）。
- 发送端收到 Ack 后，TxReplay 缓冲区将删除直至 Ack 中 ackReqSeq 字段所指示序号（含该序号）的所有表项。
- 发送端收到标准重传请求后，将从重传请求中 ackReqSeq 字段指示的序号开始，重传当前存放在发送重传缓冲区中的所有 DL 净荷 FLIT。
- 重传请求未应答前，发送端不会从 TxReplay 缓冲区删除任何表项。
- 发送端重传时，为提高可靠性，应连续发送 3 个重传请求，且均请求相同序号。
- 接收端收到新的重传后，如果重传序列号在重传请求忽略窗口内，则忽略后续的重传请求。
- 3 份重传请求应尽可能快地发送，确保每个 FEC 纠错段最多只发送 1 份重传请求。注：该机制可确保任意一个 FEC 纠错段丢失时，最多仅造成 1 份重传请求丢失。

FLIT 序号规则：有效 FLIT 序号范围为 1~511，0 保留供未来使用。序号到达 511 后回绕至 1。

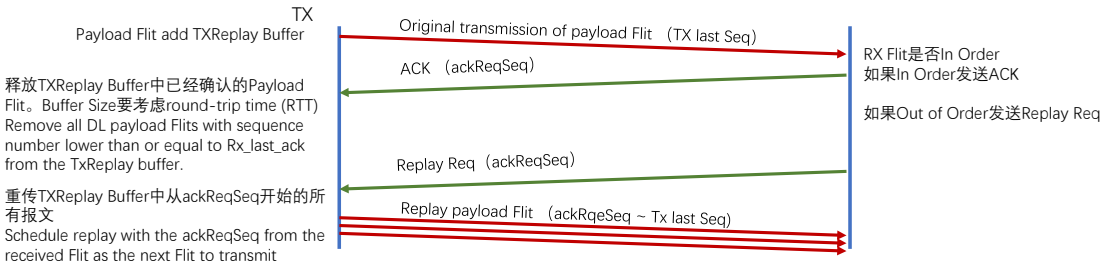


图153 DL LLR 流程

小结

本章介绍的**数据链路层（DL）处于事务层与物理层之间，主要完成 TL Flit 与 640 字节 DL FLIT 之间的打包与解包**，并提供链路内部消息处理功能；其 DL FLIT 采用标准化 640 字节格式，由 3 字节 FLIT 头、5 个字节段头、4 字节 CRC 及 628 字节净荷构成，支持显式序号、应答、重传请求等控制功能。

同时，数据链路层**链路层重传机制**，结合 FEC 纠错、CRC 校验、FLIT 序号、TxReplay 缓存、ACK 及重传请求，实现物理层错误无法纠正时的可靠数据投递，确保 FLIT 有序、不丢失、不重复，最终为上层提供高效、稳定、可靠的链路传输服务。

UALink 协议总结

UALink 协议与 OISA、ETH-X、SUE 协议的核心目标完全一致，均是实现 UPLI 指令 或 AXI 指令在各节点间的正常传递，二者的核心差异仅在于多消息聚合方式不同：

- UALink 采用基于端口的聚合方式；
- OISA、ETH-X、SUE 三款协议则采用基于目的 GPU ID 的聚合方式。

这种聚合方式的差异，直接导致了交换机侧转发逻辑的不同：

- UALink 交换机需要先对聚合的消息进行解聚合，再根据每个 UPLI 消息中携带的目的 GPU ID 完成转发；
- OISA、ETH-X、SUE 交换机无需解聚合，直接基于数据链路层报文中的目的 GPU ID 即可完成转发。

综合来看，这种设计 trade-off（权衡）的结果是：UALink 降低了 GPU 侧的实现复杂度，但相应提升了交换机侧的实现复杂度；同时，基于端口的聚合方式也提高了 UPLI 消息的聚合概率，进而提升了链路传递效率。

第二，UALink 的设计核心之一是进一步降低 GPU 侧设计复杂度，将更多硬件资源留给 GPU 核心计算。为此，UALink 去掉了可靠传输层协议，这也导致其在复杂拓扑（如多跳转发）场景中，在 CBFC（Credit-based Flow Control）流量控制、数据保序这两方面做出了一定舍弃。因此，UALink 更适合简单拓扑的 Scale UP 网络，其设计理念就是最大限度减少 GPU 芯片中通信引擎占用的 DIE Size（芯片面积）。

第三，UALink 的多个 Station（站点）、多个端口，在 UPLI 接口层面是可见的，这也就意味着这些资源对 GPU 完全可见。这就要求 GPU 在设计时，需额外考虑如何合理使用 UALink 的多个 Station 和端口、如何实现负载均衡等问题，且这些逻辑需要在 GPU 通信库的设计中予以体现。相比之下，OISA、ETH-X、SUE 三款协议在端口使用上更为灵活：既可以将多个端口聚合使用，由通信引擎自动实现负载均衡；也可以让多个端口独立工作，由 GPU 自身实现负载均衡，无需在通信库设计中额外增加过多复杂逻辑。

最后，UALink 对 UPLI 接口进行了详尽、统一的规范定义，完整规定了各类 UPLI 信号在事务层 TL FLIT 的传递方式，最大程度提升了不同厂商 GPU 之间通过 UPLI 接口实现互通的兼容性。而很多协议对 AXI（与 UPLI 功能类似的接口）的定义则较为宽泛：虽然 AXI 本身有基础标准，但对于扩展字段（如负载均衡用的 Entropy 字段等）的实现方式缺乏统一细化要求；同时，AXI 信号到事务层（TL）的封装、解封装逻辑也没有明确规范，需要各厂商自行定义。这种缺乏统一事务层报文标准的情况，导致不同厂商的 GPU 在尝试跨厂商通信时，面临显著的兼容性困难。但是也由于 UPLI 的定义，虽增加了互通性，但是也限制了 NOC 总线对外设的接口能力，也限制了很多 GPU 厂商对 UPLI 的私有化扩展。

3.7 参考文献

[1] UALink 200G 1.0 Scale-Up 互联技术白皮书

[2] Scale-Up Ethernet Framework/ Scale-Up Ethernet Framework Specification, Broadcom

[3] Enabling an Open Ecosystem for 100% AI Infrastructure Testing, OCP Summit 2025

[4] Enabling an Open Ecosystem for 100% AI Infrastructure Testing, OCP Summit 2025)

[5] Leading the Future of AI, OCP Summit 2025

[6] OAI Overview: An Open Accelerator Infrastructure Project for OCP Accelerator Module (OAM), OCP

[7] OCP Accelerator Module (OAM) Design Specification v1.5, OCP

[8] Open Accelerator Infrastructure (OAI) – OCP Accelerator Module (OAM)Base Specification r2.0 v1.0

[9] Open Accelerator Infrastructure (OAI) Universal Baseboard (UBB) Base Specification r2.0 v1.0

[10] NVIDIA GH200 Grace Hopper Superchip Architecture

[11] AMD CDNA2 Architecture

[12] 海光系统互联总线接口(HSL)规范

[13] UB-Mesh:a Hierarchically Localized nD-FullMesh Datacenter Network Architecture

[14] Sailing into the Future of the Semiconductor Industry, TMSC

[15] Sailing into the Future of the Semiconductor Industry, TMSC

[16] NVIDIA Tesla P100 White Paper

[17] NVIDIA TESLA V100 GPU Architecture

[18] NVIDIA A100 TENSOR CORE GPU Datasheet

[19] NVIDIA H100 TENSOR CORE GPU Datasheet

[20] AMD INSTINCT™ MI325X ACCELERATOR Datasheet

[21] AMD INSTINCT™ MI325X ACCELERATOR Datasheet

[22] NVIDIA Blackwell Datasheet

[23] HBM Roadmap Ver 1.7, KAIST TERALAB

[24] HBM Roadmap Ver 1.7, KAIST TERALAB

[25] The Future Of Memory Architecture For Ai Introducing: High Bandwidth Flash, Sandisk

[26] High Bandwidth Flash (HBF) Overview, Emergent Mind

[27] Hybrid Architecture Using High Bandwidth Memory and High Bandwidth Flash for Cost-Efficient LLM Inference, SK Hynix

[28] Charles Clos, "A Study of Non-blocking Switching Networks," Bell System Technical Journal , vol. 32, no. 2, pp. 406–424, 1953

[29] William J. Dally and Charles L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," IEEE Transactions on Computers, 1985

[30] R. Urata et al, "Mission Apollo: Landing optical circuit switching at datacenter scale," arXiv:2208.10041 (2022).

[31] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technologydriven, highly-scalable dragonfly topology," Comput. Archit. News , vol. 36, no. 3, pp. 77–88, 2008

[32] 灵衢超节点参考架构白皮书

[33] Enhancing distributed computing with optical interconnects in data centers and high-performance computing (HPC) systems

[34] UPN512 技术架构白皮书, 阿里云智能集团 网络研发

[35] 开放解构超节点(ODS)系统架构技术白皮书, 中国移动

[36] PCI Express® Base Specification Revision 5.0 Version 1.0

[37] https://www.ieee802.org/3/bm/public/may13/farhood_01a_0513_optx.pdf

IEEE PAM8 Draft v0 1 Summary PAM8 Draft v0.1 Summar

[38] https://semiwiki.com/wikis/semiconductor-ip-wikis/serdes-serializer-deserializer-wiki/

[39] 维基百科

[40] A new time-based architecture for serial communication links

2009 16th IEEE International Conference on Electronics, Circuits and Systems - (ICECS 2009)，DOI: 10.1109/ICECS.2009.5410875

[41] https://e2echina.ti.com/blogs_/b/analogwire/posts/51830

[42] Understanding the high speed SerDes solution space 10G-112G

https://www.chipestimate.com/Understanding-the-high-speed-SerDes-solution-space-10G-112G/Cadence/Technical-Article/2020/06/30

[43] IEEE PAM8 & FEC Options，https://www.ieee802.org/3/bm/public/nov12/bhoja_01a_1112_optx.pdf

[44] OIF 组织，OIF-FD-CEI-448G-01.0，Next Generation CEI-448G Framework

[45] OIF-FD-CEI-448G-01.0， Next Generation CEI-448G Framework

[46] SerDes (Serializer/Deserializer) Wiki

https://semiwiki.com/wikis/semiconductor-ip-wikis/serdes-serializer-deserializer-wiki/

[47] PCI Express® Base Specification Revision 5.0 Version 1.0

[48] 灵衢基础规范 2.0

[49] PCI Express Base Specification Revision 5.0 Version 1.0

[50] Cadence，Understanding PCIe 6.0 Shared Flow Control，

https://community.cadence.com/cadence_blogs_8/b/fv/posts/understanding-pcie-6-0-shared-flow-control

[51] UALink 1.0 规格

[52]灵衢基础规范 2.0

[53] PCI Express Base Specification

https://pcisig.com/specifications

Compute Express Link Specification

https://computeexpresslink.org/cxl-specification/

InfiniBand Architecture Specification

https://www.infinibandta.org/ibta-specification/

RDMA over Converged Ethernet (RoCE)

https://www.infinibandta.org/ibta-specification/

MPI: A Message-Passing Interface Standard

https://www.mpi-forum.org/docs/

UALink200_ Specification v1.0 Final

https://ualinkconsortium.org/specifications

[54] PCI Express® Base Specification Revision 5.0 Version 1.0

[55] CXL_4.0-Webinar_December-2025_FINAL

[56] CXL_4.0-Webinar_December-2025_FINAL

[57] CXL-Specification_rev4p0_ver1p0_2025November17

[58] CXL-Specification_rev4p0_ver1p0_2025November17

[59] Introducing the CXL 4.0 Specification

[60] Introducing the CXL 4.0 Specification

[61] CXL-Specification_rev4p0_ver1p0_2025November17

[62] CXL-Specification_rev4p0_ver1p0_2025November17

[63] ARM AMBA《Introduction to AMBA AXI4》:

https://www.amd.com/en/products/adaptive-socs-and-fpgas/intellectual-property/axi.html

《AMBA® APB Protocol Specification》:

https://documentation-service.arm.com/static/60d5b505677cf7536a55c245?token=

《AMBA® AXI™ and ACE™ Protocol Specification》:

https://documentation-service.arm.com/static/5f915b62f86e16515cdc3b1c

[64]《Introduction to AMBA AXI4》

[65]https://support.huawei.com/enterprise/zh/doc/EDOC1100410569/78b7de66?idPath=24030814|21782164|21782186|259602655

AMD 文档，https://docs.amd.com/r/en-US/pg051-tri-mode-eth-mac/Protocol-Description

https://www.ieee802.org/3/as/public/0503/4d0_1_CMP.pdf

[66] https://support.huawei.com/enterprise/zh/doc/EDOC1100088136/34cc009b

华为《IEEE 802.1Q 封装的 VLAN 数据帧格式》

[67] 2025 年 9 月 ODCC《ETH-XScale Up 互联协议白皮书 V1.0》

https://www.odcc.org.cn/news/p-1968136009734217730.html

[68]《ETH-XScale Up 互联协议白皮书 V1.0》

[69]《ETH-XScale Up 互联协议白皮书 V1.0》

[70]《ETH-XScale Up 互联协议白皮书 V1.0》

[71]《ETH-XScale Up 互联协议白皮书 V1.0》

[72]《ETH-XScale Up 互联协议白皮书 V1.0》

[73]《ETH-XScale Up 互联协议白皮书 V1.0》

[74]《ETH-XScale Up 互联协议白皮书 V1.0》

[75]《ETH-XScale Up 互联协议白皮书 V1.0》

[76] https://www.odcc.org.cn/news/p-1966330495924826114.html

https://www.odcc.org.cn/news/p-1995728064631037953.html

[77] 2025 年 9 月 5 日 OCP《Scale Up Ethernet (SUE) Version 1.0.0》,

https://www.opencompute.org/documents/ocp-sue-spec-final-pdf-1

[78] OCP Blog：Introducing ESUN: Advancing Ethernet for Scale-Up AI Infrastructure at OCP

https://www.opencompute.org/blog/introducing-esun-advancing-ethernet-for-scale-up-ai-infrastructure-at-ocp

[79] OCP《Scale Up Ethernet (SUE) Version 1.0.0》

[80] OCP《Scale Up Ethernet (SUE) Version 1.0.0》

[81] OCP《Scale Up Ethernet (SUE) Version 1.0.0》

[82] OCP《Scale Up Ethernet (SUE) Version 1.0.0》

[83]OCP《Scale Up Ethernet (SUE) Version 1.0.0》

[84] OCP《Scale Up Ethernet (SUE) Version 1.0.0》

[85] OCP《Scale Up Ethernet (SUE) Version 1.0.0》

[86] OCP《Scale Up Ethernet (SUE) Version 1.0.0》

[87] OCP《Scale Up Ethernet (SUE) Version 1.0.0》

[88] https://blog.apnic.net/2025/06/03/scale-up-fabrics/

[89] OCP《Scale Up Ethernet (SUE) Version 1.0.0》整理

[90] OCP《Scale Up Ethernet (SUE) Version 1.0.0》

[91] 2026 年 SPC 超节点大会技术洞察

https://www.ncsti.gov.cn/kjdt/ztbd/2026spcf/202601/t20260123_236345.html

[92] 基于 OISA 架构的 GPU 共享内存访问技术》作者：李锴 钟旭霞

《OISA Technology Specification v2.0.3 》作者：中国移动

[93] NVIDIA NVLink & NVSwitch :https://www.nvidia.com/en-us/data-center/nvlink/

NVIDIA® NVLink TM High-Speed Interconnect: Application

Performance:https://info.nvidianews.com/rs/nvidia/images/NVIDIA%20NVLink%20High-Speed%20Interconnect%20Application%20Performance%20Brief.pdf

NVLink 简介：https://blogs.nvidia.cn/blog/what-is-nvidia-nvlink/

NVIDIA Grace CPU Architecture：https://docs.nvidia.com/clara/parabricks/4.5.1/GettingStarted/GraceHopper.html

NVIDIA GH200 Grace Hopper Superchip Architecture:

https://nvdam.widen.net/s/c9lts6msjj/nvidia-grace-hopper-superchip-architecture-whitepaper%20

NVIDIA NVLink：https://blogs.nvidia.cn/blog/what-is-nvidia-nvlink/

[94] The Evolution of NVLink:

https://www.fibermall.com/blog/evolution-of-nvlink.htm?srsltid=AfmBOoppaZKus5xxbpsuiNo-OFFnOlwR4wet1-JvYC3hfkiUdmXVMwGz

[95] NVIDIA GB200 Interconnect Architecture Analysis-NVLink, InfiniBand, and Future Trends

https://naddod.medium.com/nvidia-gb200-interconnect-architecture-analysis-nvlink-infiniband-and-future-trends-91dc6ba49bf3

[96] NCCL Documentation：https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/

Demystifying NCCL: An In-depth Analysis of GPU Communication Protocols and Algorithms:

https://arxiv.org/pdf/2507.04786

Scaling Deep Learning Training with NCCL：https://developer.nvidia.com/blog/scaling-deep-learning-training-nccl/

Massively Scale Your Deep Learning Training with NCCL 2.4:

https://developer.nvidia.com/blog/massively-scale-deep-learning-training-nccl-2-4/

[97] Demystifying NCCL: An In-depth Analysis of GPU Communication Protocols and Algorithms:

https://arxiv.org/pdf/2507.04786

[98] Demystifying NCCL: An In-depth Analysis of GPU Communication Protocols and Algorithms: https://arxiv.org/pdf/2507.04786

[99] Massively Scale Your Deep Learning Training with NCCL 2.4: https://developer.nvidia.com/blog/massively-scale-deep-learning-training-nccl-2-4/

[100] NVIDIA Scalable Hierarchical Aggregation and Reduction Protocol (SHARP™): https://docs.nvidia.com/networking/display/SHARPv320

NVIDIA SHARP Technology: https://docs.nvidia.com/networking/display/sharpv390

[101] UALink，Ultra Accelerator Link Consortium, Inc. Specification UALink_200 Rev 1.0，Version 1.0，Issue Date 4/1/2025)

[102] ARM AMBA《Introduction to AMBA AXI4》: https://www.amd.com/en/products/adaptive-socs-and-fpgas/intellectual-property/axi.html

《AMBA® APB Protocol Specification》: https://documentation-service.arm.com/static/60d5b505677cf7536a55c245?token=

《AMBA® AXI™ and ACE™ Protocol Specification》: https://documentation-service.arm.com/static/5f915b62f86e16515cdc3b1c

[103] ARM AMBA《Introduction to AMBA AXI4》: https://www.amd.com/en/products/adaptive-socs-and-fpgas/intellectual-property/axi.html

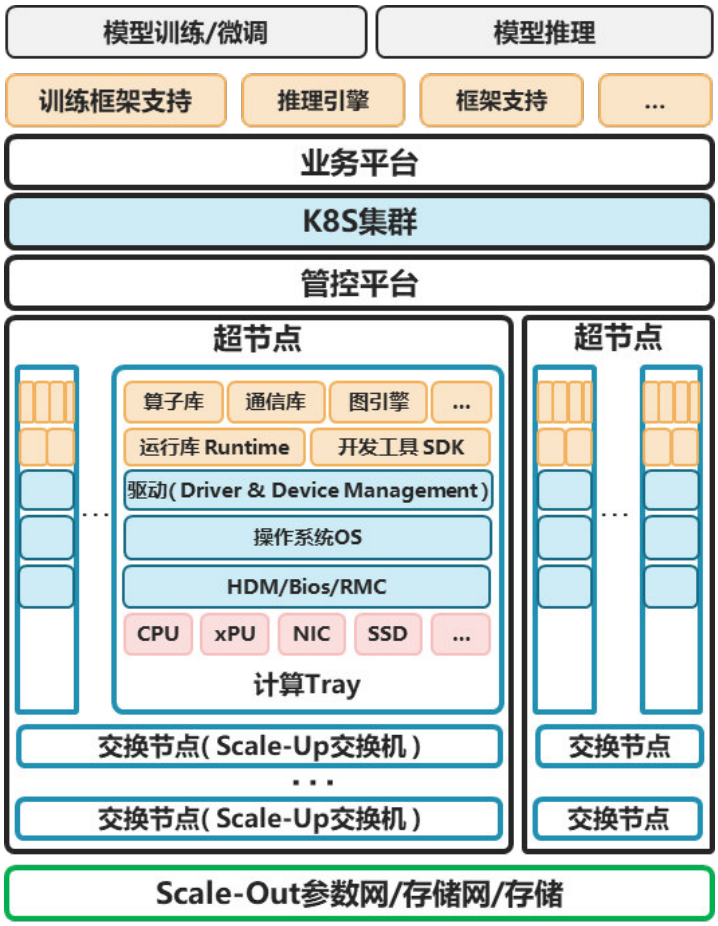
《AMBA® APB Protocol Specification》: https://documentation-service.arm.com/static/60d5b505677cf7536a55c245?token=

《AMBA® AXI™ and ACE™ Protocol Specification》: https://documentation-service.arm.com/static/5f915b62f86e16515cdc3b1c

软件栈与协同优化

1 超节点软件栈

超节点并不是一个硬件的堆砌，而是整合软硬件及平台的完整系统和方案。软件对于超节点的性能、易用性、可维护性、长任务运行稳定性至关重要。一般来讲软件包含：



- **机上软件**，即计算 Tray 上的软件。这部分软件栈与现有的算力服务器软件基本一致，一般包括运行在主机上的操作系统、驱动、设备管理组件、集群组件、运维工具及探针等，以及运行于容器中的加速卡的运行时库，开发工具、开发库、算子库、加速库、集合通信库等。其中集合通信库是超节点与现有算力服务器在软件部分最大区别之一。

- **交换节点软件**，这部分软件为交换节点嵌入式软件，与 GPU 南向互联协议强相关，实现南向交换机的管理、运行、监控是超节点不同于单机的最大区别之一，发挥超节点性能的关键之一。
- **管控平台**，负责资源管理和控制，包括资源监控，运维，故障快速感及自愈，故障分析及上报。
- **集群软件**，负责资源的集群管理、资源分配等。
- **业务平台**，负责资源管理、调度，训练、推理任务生命周期管理、状态监控，通过任务资源亲和调度，故障恢复提升性能及资源利用率。
- **AI 框架及引擎**，负责具体模型的训练及推理，抽象不同硬件的软件接口，在不修改模型的情况下，可以通过优化框架算法提升性能。

AI 框架及引擎

1. 定位与作用

AI 框架及引擎位于超节点软件栈的中间层，承担从模型表达到计算执行的核心职责，是连接算法与硬件的关键执行与调度层。其向上提供统一的模型开发与运行接口（如模型训练与推理业务），向下对接通信库与算子库，在资源调度系统分配的资源范围内完成计算任务的组织与执行。

其核心作用主要在三个方面：

- 统一抽象与开发接口：为模型提供一致的编程与执行环境，在屏蔽底层差异的同时逐步引入对硬件特性的感知能力，降低开发复杂度；
- 计算与数据流调度：对计算任务、数据依赖及跨设备执行进行统一编排，以在分布式环境下高效执行；
- 计算与通信协同：通过与通信库协同，将数据交换操作嵌入计算流程，实现计算与通信的有序衔接。

以大规模模型训练为例：模型在多个计算节点上并行执行，当完成一次前向与反向计算后，需要对各节点上的梯度进行同步。此时，AI 框架会根据并行策略决定哪些设备参与同步、在何时触发通信操作，并将同步任务编排进整体执行流程；随后，具体的数据交换由通信库完成，底层高速互联负责实际数据传输。整个训练任务运行在集群调度系统分配的计算资源之上。通过这种分层协同，AI 框架实现了计算与通信的有序衔接，从而提升整体执行效率。

在超节点架构中，面对大规模 GPU 与高速互联构成的复杂系统，AI 框架的执行调度与协同能力直接决定计算效率与资源利用率，成为系统性能发挥的关键。

2. 体系结构与核心能力

AI 框架及引擎围绕模型开发、训练与部署过程形成分层体系，主要从职责层面可以包括基础 AI 框架、训练扩展框架与推理引擎三个部分，共同构成从模型表达到高效执行的完整技术链路。

- **基础 AI 框架：**

  基础 AI 框架（如 PyTorch、TensorFlow）提供模型定义、自动求导与基础算子能力，并对底层算力资源进行统一抽象，是模型开发与执行的基础入口。其核心作用在于构建统一的计算表达与执行环境，使模型能够在不同硬件平台上运行，同时为上层扩展能力提供基础。

- **训练扩展框架：**

  训练扩展框架构建在基础 AI 框架之上，面向大规模模型训练场景，通过数据并行、模型并行、流水线并行及混合并行等策略，将计算任务扩展至多节点环境。其核心职责包括并行策略实现、梯度同步机制、任务编排与通信调度。

  在大模型训练中，训练框架不仅决定计算任务的切分方式，也直接影响通信模式与系统扩展效率。例如，通过分片参数（FSDP）或训练框架（Megatron-LM）等机制，可以在提升显存利用率的同时降低通信开销，从而实现更高效的规模扩展。典型实现包括 DeepSpeed、Megatron-LM、PyTorch FSDP 等。

- **推理引擎：**

  推理引擎面向模型部署与服务化场景，通过对计算图、执行路径及硬件特性的深度优化，实现低延迟、高吞吐的推理能力。在大模型场景下，推理引擎还负责中间状态管理（如 KV Cache）与请求调度，以减少重复计算并提升长上下文处理效率。

  例如，通过 KV Cache 复用机制，可以避免重复计算历史上下文，从而显著降低长序列推理的计算开销；通过动态批处理与请求调度机制，可以在多请求并发场景下提升整体吞吐能力。代表性推理引擎包括 vLLM、TensorRT-LLM 及 HuggingFace TGI 等。

在上述分层体系之上，AI 框架及引擎的核心能力集中体现在以下三个方面：

- 计算与执行编排能力：提供模型计算任务的统一表达与执行编排。在训练场景下，支持自动求导、多维并行策略（数据并行、模型并行、流水线并行等）的编排；在推理场景下，支持计算图执行、算子高效执行、请求调度及动态批处理，实现计算任务的高效组织与执行；
- 通信协同能力：管理跨设备的数据交换与同步机制，根据并行策略和底层互联拓扑编排通信任务，并与通信库协同实现计算与通信的重叠执行，提升效率；
- 数据与内存管理能力：统一管理模型参数、中间状态及运行时数据。在训练场景下，管理梯度、优化器状态与激活值，支持显存复用与分片；在推理场景下，管理 KV Cache 与请求上下文，支持内存复用与动态调度，提升显存利用效率，支撑更大规模模型运行。

因此，AI 框架及引擎不仅是模型开发与执行平台，更是面向大规模算力系统的统一调度与运行基础。

3. 超节点协同与发展趋势

在超节点架构下，AI 框架及引擎从单机优化工具演进为系统级协同优化核心。超节点由大规模 GPU 与高带宽、低延迟互联网络构成，这种高密度算力结构对框架提出了更高要求，主要体现在以下方面：

- 更大规模的并行能力：支持更细粒度的计算切分与多维并行策略组合，实现跨数十到数百设备的高效扩展。
- 拓扑感知的通信调度：感知底层互联拓扑结构，将通信任务优先调度到同一超节点，降低跨域通信开销。
- 计算与通信的深度融合：通过精细化执行编排，实现计算与通信的高度重叠，减少同步等待带来的性能损失。

在此基础上，AI 框架及引擎的发展呈现出两大趋势：

第一，软硬协同持续深化。

框架将从“统一抽象硬件”进一步演进为“感知并利用硬件”，针对超节点的高速互联与内存体系优化通信调度与数据访问路径，充分释放底层硬件能力。

第二，统一执行平台演进。

框架逐步从训练或推理工具演进为统一的算力执行平台，在同一系统中支持多任务并行运行与动态资源调度，实现训练与推理任务的协同执行与资源共享。

总体来看，AI 框架及引擎正从以开发效率为核心的软件工具，演进为超节点算力系统的核心调度与优化中枢。一方面，其通过高效的执行调度与协同机制，将底层硬件能力转化为可用算力；另一方面，通过持续的软硬协同优化，使上层应用能够高效利用超节点的计算与互联资源。作为连接算法与硬件的关键纽带，AI 框架及引擎是实现超节点算力高效释放的核心基础设施。

2 软硬协同优化

无论训练还是推理，性能的极致优化都依赖于软硬件的紧密协同。其核心目标在于最大化算力效力，竭力消除重复计算与数据等待的无效时间。常见的优化手段包括：提升通信带宽、优化通信算法、实现计算与通信的重叠执行以减少等待；通过缓存 KV Cache 到加速卡之外，计算 KV Cache 的 P 节点与使用 KV Cache 的 D 节点分离来规避因显存不足导致的重计算。

在此基础上，超节点通过构建更大规模的 Scale-Up 域，提供了更高的互联带宽与更低的时延，从而奠定了卓越的通信性能。再辅以集合通信优化、推理引擎调优等软件层面的加持，便能充分释放超节点硬件架构所带来的全部潜能。

大模型推理性能优化：Prefill-Decoding 阶段分离（PD 分离）

在大语言模型（LLM）推理中，一次完整的推理请求包含两个计算特征截然不同的阶段：

- **Prefill（预填充）阶段：**处理用户输入的完整 Prompt，对所有输入 Token 进行并行前向计算，生成初始 KV Cache。此阶段为计算密集型（Compute-Bound），计算量大但延迟要求相对宽松，GPU 算力利用率高。
- **Decoding（解码）阶段：**逐 Token 自回归生成输出序列，每一步仅计算一个新 Token。此阶段为访存密集型（Memory-Bound），计算量小但对延迟极为敏感，GPU 算力利用率低，主要瓶颈在于 KV Cache 的读取与注意力计算中的访存操作。

在传统的 LLM 推理系统中，Prefill 和 Decoding 在同一组 GPU 上交替执行。这带来严重的资源效率问题：Prefill 的大批量计算会抢占 Decoding 的 GPU 时间片，导致 Decoding 延迟（即用户感知的 Token 间延迟，TPOT）大幅波动；Decoding 阶段 GPU 算力大量空闲，但因需维持 KV Cache 而无法释放给 Prefill 使用；两个阶段的 Batch Size 优化目标相互冲突：Prefill 倾向小 Batch 低延迟，Decoding 倾向大 Batch 高吞吐。

PD 分离（也称为 Prefill-Decoding Disaggregation）通过将两个阶段解耦到独立的硬件资源池中来解决上述矛盾：

- **Prefill 资源池（P-Pool）：**配置高算力加速器，针对计算密集特征进行优化（如增大 Batch Size、启用 Flash Attention 的 Prefill 模式等），最大化 TTFT（Time To First Token）吞吐量。
- **Decoding 资源池（D-Pool）：**配置大显存、高带宽加速器，针对访存密集特征进行优化（如连续批处理 Continuous Batching、KV Cache 内存池化等），最大化 Token 生成吞吐量并保障 TPOT 稳定性。

- **KV Cache 传输通道：**Prefill 阶段完成后，生成的 KV Cache 需从 P-Pool 传输至 D-Pool。这一传输过程是 PD 分离架构的关键通信挑战。

通过该架构，系统实现了计算路径与访存路径的解耦，使不同类型负载能够在各自最优资源环境中独立优化，从而提升整体资源利用率与服务稳定性。

而超节点的高带宽互联为 PD 分离架构又提供了独特优势——KV Cache 的跨 Pool 传输瓶颈得到显著缓解，从而支持更深入的系统级优化：

- **动态 P/D 资源配比调节：**基于系统 Token 吞吐等反馈指标，按实时请求特征（长 Prompt/短输出 vs. 短 Prompt/长输出）自适应调整 P-Pool 与 D-Pool 占比，并通过软件虚拟化实现 GPU 在两类资源池间的秒级弹性切换（“借还”机制），提升整体资源利用率。
- **KV Cache 高速传输与压缩：**围绕 P→D 关键路径，通过 GPU 直连（P2P）实现显存级数据传输以规避 PCIe 与 CPU 中转开销，同时结合量化压缩（FP16→INT8/INT4）与 Token 剪枝降低数据规模，并按 Transformer 层分块传输以降低时延并提升传输效率。
- **分级缓存管理：**构建 L1（本地 HBM）、L2（本地 Memory）、L3（本地/共享 SSD 存储）的多级 KV Cache 体系，在容量与时延之间取得平衡，并基于请求活跃度与 Attention 特征进行数据分层与迁移，结合 LRU 与 Attention Score 实现缓存淘汰优化。
- **全局请求调度与负载均衡：**通过全局请求路由器实现跨 P/D 节点的统一调度，在请求级、Prefill 侧与 Decoding 侧分别进行多级负载均衡，并结合预测性资源预分配减少冷启动开销，同时在极低时延场景下支持请求级冗余执行以优化响应延迟。

总体而言，PD 分离通过将 Prefill 与 Decoding 两类异构负载进行资源级解耦，从根本上解决了传统混合部署下算力利用率与时延稳定性之间的冲突。在此基础上，结合超节点架构提供的高带宽互联与统一调度能力，系统能够围绕资源配比、数据传输、缓存管理与全局调度等关键路径进行协同优化，进一步释放性能潜力。

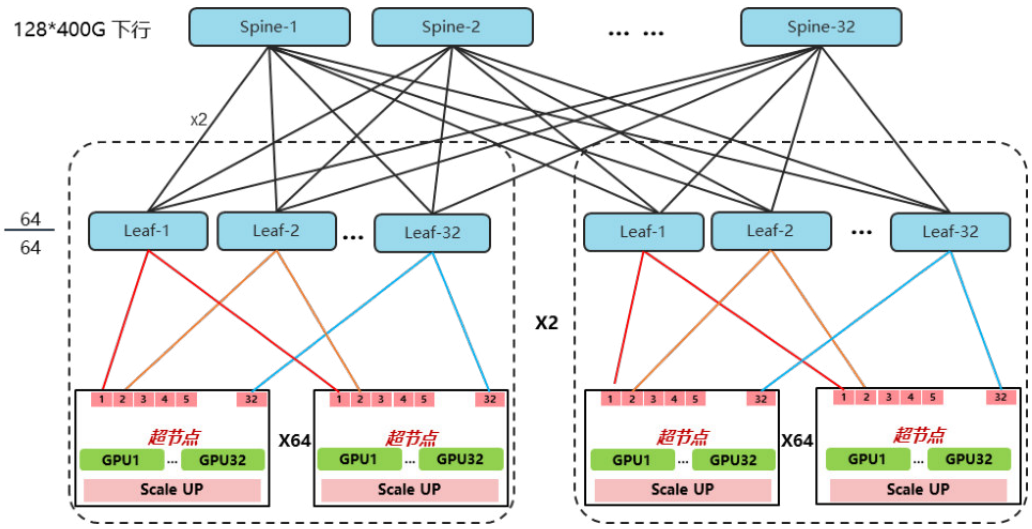
集群通信优化

单个超节点虽然极大的提升了单节点的算力，但是为了支撑大模型训练和高性能推理，仍然需要将大量的超节点组成算力集群。集群内通信的核心意义在于突破单超节点的扩展瓶颈，使聚合更大算力。因此集群性能优化对发挥整个集群的算力效率，有着至关重要的作用。集群性能优化需要从 Scale-Out/Scale-Up 融合的网络设计、模型并行策略、集合通信软件等角度进行考虑。

1. Scale-Out 网络的收敛比设计与挑战

在 AI 训练集群中，模型训练和推理过程会产生大量微突发流量。为避免 Spine-Leaf 网络架构中因带宽不足造成网络瓶颈，当前主流设计采用收敛比为 1:1 的架构，即 Leaf 交换机的上行与下行端口数量相等。这意味着每个 Leaf 交换机需配置一半端口下行连接 GPU 服务器/超节点，另一半端口上行连接 Spine 交换机。

以 H3C S9827 交换机为例，其最大支持 128 个 400Gbps 端口，作为 Leaf 交换机时，64 个端口用于下行服务器接入，64 个端口上行连接 Spine 交换机。这种对称设计确保无阻塞转发能力，但也带来了流量分发策略的挑战——当报文从 Leaf 转发至 Spine 时，如何选择最优的上行端口成为关键问题。超节点典型组网如下：



2. Scale-Out 网络负载不均问题

最直接的选路方式是等价路由（ECMP，Equal-Cost Multi-Path routing）。Leaf 交换机根据报文特征（如源目的地址、端口号、协议类型等）计算哈希值，并根据哈希值选择上行端口。该方法的优势在于保证相同流走相同路径，避免报文乱序问题。

然而，ECMP 存在显著的负载不均缺陷。在 640 卡 GPU 进行 all reduce 测试中，不同上行端口的报文发送分布呈现极不均匀状态。这种不均匀导致虽然总体带宽充足，但部分链路因拥塞造成整体网络性能下降，严重时网络性能仅为理想情况的 60%左右。解决哈希不均带来的网络性能下降，成为大规模 Scale-Out 网络的关键挑战之一。

3. 负载均衡优化技术

SDN 路径导航利用集合通信的周期性特征，通过控制器进行全局路径规划，需 CCL 深度适配，可大幅提升带宽利用率；逐包负载技术利用 RDMA 块协议特性，让交换机按实时负载分发报文；DDC 架构则将传统机框解耦为分布式盒式设备，通过信元切片与重组技术，在网络侧实现精细化负载均衡。三者各有侧重：SDN 通过软件实现，只依赖交换机通用的能力硬件依赖小；逐包负载胜在透明性，相对成熟，但是需要网卡与网络设备配合使用，需要依赖网卡恢复网络逐包负载均衡丢失的顺序；DDC 则以网络架构创新实现了端测网卡网侧交换机的解耦与极致均衡，需要使用支持 DDC 的接入及核心交换机进行组网。

4. Scale-Out 网络服务器接入拓扑优化

- 服务器内部拓扑结构

  典型的 8 卡算力服务器中，多块加速卡通过 Scale-Up 方式实现节点内算力扩展。为保证每块加速卡的节点间通信性能，服务器等量配置高性能 IB 或 RoCE 网卡用于 Scale-Out 扩展，当前网卡带宽已达 400Gbps 甚至 800Gbps。

  服务器内部采用 PCIe Switch 将加速卡与 Scale-Up 网卡互联。集合通信库进行跨节点通信时，基于内部拓扑感知，优先使用同一 PCIe Switch 的 Scale-Up 网卡进行通信。通过 RDMA、GDR（GPUDirect

RDMA）等技术，实现跨节点加速卡之间通信数据，直接从加速卡经过网卡传输，完全绕过主机内存和CPU，最大限度降低时延和性能损耗。

- 单轨接入与多轨接入对比
  - **单轨接入：**将一台服务器的所有网卡接入同一台 Leaf 交换机。优势在于组网连线短，可采用低成本电缆接入，有效降低组网成本。同节点加速卡间通信仅需经过 Leaf 一跳即可达。但该接入方式存在根本性问题——Scale-Out 最优流量路径与 Scale-Up 的 HBD（高带宽域）重叠。由于 Scale-Up 网络带宽远高于 Scale-Out 网络，时延也更低，集合通信库选路时会优先选择 Scale-Up 网络，导致 Scale-Out 的同 Leaf 最优路径被浪费。
  - **多轨接入：**将若干服务器的同编号网卡接入同一台 Leaf 交换机。这种拓扑连线较长，通常需采用光纤接入，组网成本较高。但其核心优势在于 Scale-Out 的最优路径与 Scale-Up 的 HBD 正交，能够充分发挥两种网络的协同性能。
- 基于流量模型的接入策略分析

  从并行策略角度分析模型训练流量特征，可深入理解接入策略选择：
  - **张量并行（TP）：**将模型大矩阵拆分到服务器内不同 GPU 运行，通过 AllReduce 同步计算结果。单次通信量大，通信频率极高，对吞吐和时延要求极为严苛。基本通过 Scale-Up 网络通信，这也是 Scale-Up 网络需具备超高吞吐和超低时延的根本原因。
  - **流水并行（PP）：**将模型中不同层放置在不同服务器计算，通过 P2P 同步计算结果。单次数据量小，通信频率较高，吞吐要求相对较低。可能存在跨节点通信场景，主要为流水线前一个节点最后一块加速卡与后一个节点第一块加速卡之间通信。
  - **数据并行（DP）：**将样本数据拆分到不同模型副本计算，通过 AllReduce 同步计算结果。单次通信量极大，但每个迭代周期仅完成一次通信，频率较低，对通信吞吐要求极高。跨节点部署 DP 时，通常为不同节点的同编号加速卡之间通信。

  模型训练流量具有周期性和强同步特征。周期性指每个迭代周期产生相同特征流量；强同步指同一阶段通信为整个集群相关加速卡同时启动和结束，通信效率取决于性能最差的一个。考虑到 DP 流量的极大通信量特点，需最大限度保证通信吞吐，避免不同加速卡通信路径相互干扰。在 Spine-Leaf 组网拓扑下，DP 流量能在同一 Leaf 下完成最为理想。由于 DP 通信为同编号加速卡之间，将同编号加速卡接入相同 Leaf 对 DP 流量最优——这正是多轨接入的核心优势。

集合通信拥塞控制

1. AllReduce 拥塞问题分析

AllReduce 算子将不同加速卡数据进行汇总并在所有卡间同步，常用操作包括 add、max、min、avg 等。跨节点 AllReduce 最常用场景为 DP 训练中的梯度同步，一般为同轨通信。

NVIDIA NCCL 实现了多种 AllReduce 算法，包括 Tree、Ring、NVLS、NVLS Tree、CollNet Direct、CollNet Chain 等。不同算法对网络流量模型产生显著影响：

Tree 算法分为两阶段：首先通过 Reduce 算子将所有数据汇总到根节点，再通过 Broadcast 将汇总数据同步至整个通信域。在 Reduce 阶段，除最后一次外，每次均为两个加速卡向一个加速卡汇总数据，形成明显的"二打一"流量模式。这种模式在加速卡入方向产生严重拥塞，且集群规模越大、数据量越大，拥塞越严重。

通过链路中 ECN 报文数量可定性判断拥塞程度。480 卡 NCCL Test 采用 Tree 算法进行 AllReduce 时，单台 Leaf 交换机产生大量 ECN 报文；640 卡测试时 ECN 数量显著增加。对应地，640 卡测试中 4GB 数据量的 busbw 带宽仅为 480 卡的 60%左右，充分说明大数据量操作时，更大规模集群使用 Tree 算法会在加速卡接入端产生严重拥塞，对 AllReduce 性能造成极大影响。

Ring 算法将 AllReduce 分为 ReduceScatter 和 AllGather 两阶段。每个阶段将通信域加速卡串接成环，从根本上避免了 Tree 算法的"二打一"拥塞。实测 ECN 数量极少甚至完全消除，意味着通过算法优化消除了拥塞问题。640 卡集群测试中，4GB 数据量下 busbw 可达物理带宽的 95%。

- 算法选择策略

Ring 算法并非完全优于 Tree 算法。相同规模集群下，Ring 算法操作步骤为 2N-1，显著多于 Tree 算法的 $2(\log_2 N)$。因此，小数据量场景下，Tree 算法时延明显小于 Ring 算法；随着数据量增加，Ring 算法的高吞吐优势逐渐显现。

480 卡实测数据显示，小数据量时 Tree 算法时延明显低于 Ring 算法。这表明不同算法适用不同场景——小数据量场景宜选用 Tree 算法降低时延，大数据量场景宜选用 Ring 算法提升吞吐，具体选择需结合实际场景权衡。

2. Sacle-Up 与 Scale-Out 网络的协同优化

随着 DeepSeek V3 将 MoE（混合专家）模型工程化落地，越来越多模型采用 MoE 架构。MoE 模型的训练和推理产生大量 All-to-All 流量，具有两大特点：大量跨轨加速卡通信和大量多打一流量模式。

为优化此类流量路径，NVIDIA 引入 PXN（PCIe x NVLink）技术。该技术先将跨轨流量通过高性能 Scale-Up 网络转发至同节点同轨加速卡，再通过同轨路径进行跨节点转发。据 NVIDIA 数据，采用 PXN 技术可减少 All-to-All 通信时延约 50%。同样，PXN 技术需在多轨接入组网下才能充分发挥效能。

从 AllReduce 和 All-to-All 通信算法的优化实践可以看出，优化集合通信算法能够有效改善流量模型，减少甚至避免网络流量"多打一"带来的性能下降和不确定性，是构建高性能 AI 集群的关键技术手段。

软件优化

在超节点（Super Pod）架构中，硬件性能决定了算力的“天花板”，而软件优化则决定了实际运行中的“有效利用率”。即便拥有最顶级的互联拓扑，如果软件栈未能针对超节点的大带宽、低延迟特性进行深度适配，系统依然会面临严重的通信开销和计算闲置。

本节将从底层库适配、内存管理、并行策略及监控诊断四个维度，探讨如何在软件层面榨取超节点的极致性能。

1. 通信库的深度定制与原语优化

超节点的核心优势在于其非阻塞、全互联的通信能力。标准的通信库（如 NCCL 或 HCCL）在通用环境下表现优异，但在超节点特定拓扑下，往往存在冗余的握手协议或通用的算法选择（如默认使用 Ring 算法）。

- **拓扑感知算法：**软件栈应能自动识别超节点内部的物理链路（如 NVLink 或自定义高速互联总线）。针对 All-Reduce、All-to-All 等高频操作，优化程序应弃用传统的环形拓扑（Ring），转而采用基于多叉树（Tree）或混合型拓扑的通信算法，以减少跳数并充分利用物理层面的多路径并发能力。
- **通信与计算重叠（Overlap）：**通过精细化的 Pipeline 调度，将数据切片并开启异步通信流。在计算内核（Kernel）执行的同时，预取下一轮迭代所需的数据，确保通信开销被掩盖在计算耗时之内。

2. 内存管理与零拷贝技术

在大模型训练等超大规模计算场景中，内存带宽往往比计算频率更容易成为瓶颈。

- **显存池化管理：**频繁的内存申请与释放会产生大量碎片。通过构建专用的内存池，实现显存的统一配给与复用。同时，利用超节点内部的统一编址（Unified Memory）特性，允许计算节点直接跨链路访问邻近节点的内存空间，减少 CPU 介入。
- **零拷贝（Zero-copy）机制：**优化软件路径，确保数据在网络接口卡（NIC）、存储器与计算核心之间传输时，无需经过多次 CPU 内存拷贝。利用 RDMA（远程直接内存访问）技术结合用户态驱动，使数据流直接穿透至应用层。

3. 算子库融合与编译优化

超节点的强大算力需要高效的算子来驱动。标准的深度学习算子往往针对通用场景，缺乏对特定指令集的极致压榨。

- **算子融合（Operator Fusion）：**软件层面应利用图编译器（Graph Compiler）将多个连续的计算算子合并为一个大算子。这不仅减少了 Kernel Launch 的频率，还通过寄存器级的数据复用，显著降低了访存延迟。
- **动态 Shape 适配：**针对推理场景中多变的输入尺寸，软件栈需支持动态维度优化，避免因 Padding（填充）带来的无效计算，从而提升超节点在长文本或多模态任务中的吞吐量。

4. 并行策略的精细化组合

超节点不仅是物理上的集成，更是逻辑上的单体化。软件优化需重新定义张量并行（TP）、流水线并行（PP）与数据并行（DP）的权重。

- **内聚型模型并行：**由于超节点内部延迟极低，软件设计应倾向于将计算强度最高、通信最频繁的模型切片置于同一台超节点内部。

5. 全栈性能可观测性

再完美的算法也需要实际环境的校准。软件优化的最后环节是建立“毫秒级”的监控体系。

- **链路级追踪：**开发深度性能剖析工具（Profiling），实时捕获每一条物理链路的拥塞情况及重传率。
- **异常自动降级：**当检测到某个计算单元或通信链路性能波动时，软件层需具备自动降级或旁路路由能力，确保超节点集群的整体稳定性，避免“长尾效应”拖慢整个计算任务。

超节点性能监控

1. 超节点性能监控的重要性

超节点作为大规模 AI 计算的核心载体，其系统复杂度呈指数级增长，芯片、机 柜数量的剧增，使得硬件故障（如芯片损坏、链路中断）成为常态，性能监控已从“可选项”变为“生命线”。

首先，规避“静默故障”的毒化效应：这是超节点监控面临的最大挑战，在大规模分布式训练中，超节点中哪怕只有一张 GPU 出现“静默故障”（即表面在线但性能大幅下降），就会像毒药一样污染整个训练任务的梯度;

其次，保障超节点规模下的线性加速比：超节点依赖 Scale up 高速互连实现大量 GPU 卡协同，任何单点的性能衰减（如 PCIe 重传增加、Scale up 带宽波动）都会通过“木桶效应”拖慢整个训练任务。没有全栈、全维度的监控，无法定位性能瓶颈究竟源于计算、显存、通信还是数据预处理;

第三，支撑超节点的精细化调度和运营：以监控作为依据，从而优化模型并行策略、数据流水线和作业调度，超节点的功率密度极高，可追踪功耗与温度以优化 PUE，是实现“每瓦特每秒性能”最大化目标的前提。

2. 超节点性能监控范围与关键监控指标

超节点性能监控需构建“端到端、分层立体”的监控体系，覆盖从芯片到数据中心、从基础设施到 AI 作业的全栈范围，如下为典型监控指标举例：

监控层级	监控范围	关键监控指标
硬件基础设施层	GPU	算力利用率、显存利用率、GPU温度/功耗、ScaleUp/ScaleOut带宽利用率/错误计数、ECC错误计数
	CPU与内存	CPU利用率、内存利用率/已用量、缓存命中率、NUMA状态
	高速互连网络	端口发送/接收带宽、端口误码率、拥塞控制计数、延迟、交换机温度/功耗
	存储	IO带宽（读/写）、IOPS、延迟、存储池使用率、等并行文件系统OST/MDT状态
	供电与散热	机架/节点功耗（PUE）、进出风温度、风扇转速、冷却水流量/温度
系统与平台层	操作系统	内核错误日志（dmesg）、进程/线程数、文件描述符数、网络连接状态
	集群调度器	作业队列状态、作业资源分配情况、Pending/失败作业数、调度器性能
	容器运行时	容器资源限制与使用量、镜像拉取状态、运行时状态

监控层级	监控范围	关键监控指标
AI框架与作业层	分布式训练框架(PyTorch, Tensorflow)	全局迭代速度（iterations/sec）、Loss曲线、梯度同步时间、数据加载时间
	集合通信库(CCL)	AllReduce、AllGather等集合通信操作耗时、算法选择、通信吞吐量
	模型与算法	张量并行/流水线并行的负载均衡度、CheckPoint检查点保存时长； 推理时延、请求队列长度、批处理大小

3. 超节点性能监控工具链技术

工具链通常由数据采集代理、时序数据库、可视化与告警平台及上层分析工具构成。

工具类别	关键作用	常用工具
数据采集	数据采集代理，负责超节点基础设施、系统平台、到AI框架层的指标采集，并输出到上层数据存储工具	Prometheus Node Exporter，是监控超节点服务器基础指标的首选，其他 OpenTelemetry Collector、Telegraf等都支持对接Prometheus后端。 NVIDIA DCGM，专为NVIDIA GPU设计，可监控NVLink、SM利用率等关键指标
数据存储	提供多维监控数据存储，需要高性能写入和查询能力，通常支持时序数据的处理、支持跟踪（Traces）、指标（Metrics）和 日志（Logs）类可观测性数据	Prometheus、OpenTelemetry是云原生监控的标准工具，具有丰富的生态， VictoriaMetrics具有高压缩比和高性能特性，可作为Prometheus的扩展，提供超大规模数据的长期存储 InfluxDB适合对写入吞吐量要求极高的时序数据存储场景
可视化与告警平台	提供超节点监控数据的可视化和分析，支持丰富的数据源和、提供仪表板编辑和告警功能	Grafana，是监控栈可视化的事实标准，能够将全栈监控数据在一个平台统一展示。 Kibana擅长日志数据的可视化分析，当需要将指标监控与日志（如训练作业日志、内核日志）进行关联分析时使用。
高级分析与剖析	AI框架的剖析工具，支持模型与算法层的性能分析，跟踪模型前向反向传播、算子耗时、CUDA内核执行等，以优化超节点场景中的模型计算图、数据加载和算子实现	PyTorch Profiler，TensorBoard

一个优秀的超节点监控系统，应能做到从物理功耗到模型 Loss，从单卡 SM 利用率到全局通信拓扑的全栈可观测，并将数据转化为自动化的运维动作和性能优化建议，从而保障这座“AI 算力超节点”的稳定、高效运行。

Prometheus + Grafana 性能监控组合成熟、开源、灵活，覆盖了 90%的监控需求，是构建自定义监控体系的最常用选择。

另外为了适配不同超节点产品的特点，超节点算力资源提供商往往会提供私有的 Agent 代理采集器+分析工具作为补充（比如 H3C 的 ADDC Agent 采集器+ADDC 分析器），提供更多差异化的监控指标，以支撑上层资源调度和业务优化。

超节点集群管理平台

在超节点技术架构中，集群管理平台是实现物理资源池化与逻辑任务协同的核心中枢。该平台打破了传统数据中心以“单台服务器”为最小调度单元的限制，通过深度整合高速互联拓扑与内存语义通信，实现了对整簇超节点资源的高效编排与自动化调度。

1. 深度 Scale-up 感知的精细化调度机制

不同于传统的分布式调度，超节点平台通过“穿透”物理边界，实现对 Scale-up 架构内内存语义通信与全互联拓扑的极致利用：

- **Scale-up 拓扑亲和性建模：**调度器由感应“节点间网络”升华为感应“总线级拓扑”。平台实时映射超节点内部的 Scale-Up 互联矩阵，识别极低延迟的共享内存域。调度决策不再基于“是否有空闲节点”，而是基于“任务通信半径是否在 Scale-up 最佳互联带宽内”。超节点能够提供更大的 Scale-Up 域，因此能够在调度亲和性上更加灵活
- **跨节点语义对齐的统一编排：**平台通过感知 Scale-up 架构下的物理拓扑对称性，实现算力单元的“逻辑粘合”。针对大模型训练，自动将模型并行（TP）颗粒度限制在单组 Scale-up 域内，避免跨机传统以太网的协议损耗。
- **自愈式资源碎片重整：**平台具备 Scale-up 动态重构能力。当超节点内出现算力空洞时，调度器通过非对称资源整合，动态调整逻辑边界，将散落的物理卡重新聚合成高带宽、大显存的逻辑 Scale-up 单元，提升超大规模作业的准入门槛。
- **根据互联拓扑进行通信优化：**通过集合通信库的优化，充分利用 Scale-Up 网络高带宽、低时延、多路径等能力，将跨轨的 Scale-Out 流量通过 Scale-Up 进行优化，可以有效提升 MoE 模型的 All-to-All 的流量性能。
- **千亿级参数模型的平滑支撑：**凭借对全局内存池（Global Memory Pool）的调度感知，平台支持将超大模型权重跨物理实体放置在 Scale-up 域内，解决了单机显存无法承载超大上下文（Context Length）的痛点，显著提升了推理与训练的吞吐量。

2. 软件定义的池化资源编排

超节点突破了传统单机柜物理边界，通过高速交换背板与低延迟内存语义协议，将分布在不同 Tray 中的计算与存储资源深度重组，实现从“单设备池化”向“多 Tray 整体池化”的跨越：

- **跨 Tray 显存级联与超大规模池化：**集群管理平台能够跨越物理 Tray 的限制，将多个算力单元的显存通过物理链路整合为统一的虚拟显存逻辑池。在处理超大规模参数模型（如万亿级 LLM）时，平台可动态调用相邻甚至跨 Tray 的空闲显存，为单一计算任务提供超越物理单卡、单 Tray 极限的存储支撑，真正实现“显存随算力动，容量随任务扩”。
- **多维度异构资源细粒度编排：**平台构建了覆盖全簇 Tray 的标准化资源图谱，将各 Tray 内的 GPU、NPU、DPU 及高速缓存统一编号。通过增强型调度引擎，实现“算力单元+跨 Tray 显存+网络带宽”的三元组精确匹配，彻底消除因 Tray 间物理隔离导致的资源碎片，确保整机系统无孤岛。
- **全链路存算协同加载：**管理平台通过感知全局数据分布，利用超节点内部的横向高速互联拓扑，实现跨 Tray 的数据预热与并行加载。在任务下发瞬间，平台协同多 Tray 存储资源将热点数据秒级推送至近端池化显存中，极大压减了大模型训练与推理任务的初始化等待时间。

3. 任务全生命周期的自动化管理

针对千卡级以上的超大规模智算作业，集群管理平台提供了从部署、校验到执行的全流程自动化支撑。

- **优化软件栈的一致性分发：**平台预置了深度适配超节点硬件特性的环境镜像，包含特定版本的通信库（如定制化 xCCL）、编译器及加速插件。利用高并发镜像分发技术，可实现千量级节点软件环境的秒级同步与校验，确保集群内部不存在版本漂移引发的性能波动。
- **作业预校验流程：**在正式拉起大规模计算任务前，平台会自动执行链路完整性测试与硬件基准校验。只有当参与计算的所有节点及互联路径均满足预设的带宽与时延阈值时，任务方可进入执行状态，极大降低了长周期作业因底层链路隐患导致的运行中断。
- **弹性扩展与动态迁移：**得益于超节点的资源池化特性，平台支持任务的“无感扩容”。当业务优先级调整时，管理平台可动态调度额外的池化算力加入当前的计算域，并自动触发通信域重组，确保算力规模的线性增长。

4. 多租户隔离与资源效能精细化运营

为了满足高性能计算环境下的多租户诉求，平台在逻辑安全与物理资源分配上进行了深度定制，平衡了资源共享的灵活性与任务运行的稳定性。

- **逻辑安全隔离与协同调度：**平台通过逻辑分区与访问控制（ACL）技术，确保租户间的数据访问严格隔离。在流量管理上，重点从任务排队算法入手，通过感知拓扑的静态路由分配，在调度阶段规避不同任务间的链路重合，从而降低网络拥塞风险，实现业务层面的“类带宽保障”。
- **Scale-up 与 Scale-out 双维隔离策略：**
    - **Scale-out（横向）隔离：**针对多节点间的并行任务，平台利用调度引擎实现节点组的物理划分，确保大规模计算任务在独占的节点集群内运行。
    - **Scale-up（纵向）隔离：**针对单机/超节点内部，平台支持 GPU 切分与内存亲和性绑定。通过限制单一任务对片上互联总线（如 NVLink/CXL）和显存带宽的占用，防止“噪声邻居”效应，确保单机内部多租户任务的性能确定性。

- **多维资源计量与配额模型：** 针对池化资源特性，平台提供基于算力当量、显存容量及计算周期（GPU-Hours）的核算体系。通过设置精细的软硬配额（Quota），确保核心科研任务在资源竞争时具备确定的调度优先级。
- **资产与能效可视化管理：** 平台提供机柜级的热力分布与实时功耗图谱。通过智能化的负载迁移策略，将高热任务与低功耗区域进行匹配，在保障硬件寿命的同时，有效降低超节点集群的整体运行能耗（PUE）。

操作系统和超节点

在人工智能进入万亿参数时代，超节点（Super Pod）已成为 AI 基础设施的核心构建单元。以 NVIDIA GB200 NVL72 为代表的超节点系统，通过第五代 NVLink 域将 72 张 Blackwell GPU 互联成逻辑统一的“巨型 GPU”，提供 130 TB/s 以上的聚合带宽 [1]。在此架构下，操作系统不再是单纯的资源管理者，而是 Scale-up 互联（机架内高速直连）的核心协调层，负责拓扑发现、链路管理、NUMA/CXL 感知、通信库优化、功耗热控、故障域隔离等多维度功能。

操作系统同时也承担 Scale-out（跨节点集群）管理职责，例如通过 RDMA 子系统、GPUDirect 和 Fabric Manager 接口支持 InfiniBand/RoCEv2 的跨机架扩展。然而，本白皮书重点聚焦 Scale-up 场景下的 OS 作用——即如何让单个机架内的数十至数百 GPU 像单一大 GPU 一样高效协同，而非详尽探讨集群级网络优化。

后续章节将从 AI 负载画像入手，逐步剖析 CPU-GPU 异构互联、单节点多 GPU、超节点 Scale-up 管理、资源池化，直至统一内存语义演进，帮助系统架构师与运维工程师构建高效、可扩展的 AI Factory。操作系统在超节点时代已从传统的资源管理器，彻底演变为 AI 基础设施的“神经中枢”与“互联大脑”。本白皮书主要聚焦于单机架/单节点内的 Scale-up 场景——即如何通过 NVLink、CXL、PCIe 等高速直连技术，将数十至数百张 GPU 融合为逻辑统一的“巨型加速器”，并由操作系统负责拓扑发现、NUMA/CXL 亲和调度、Pinned Memory 优化、GPUDirect 路径管理、功耗热控与故障域隔离等核心功能。

同时需要说明的是，操作系统在更大规模的 Scale-out 场景（跨节点集群、万卡级分布式训练）中同样承担关键职责，例如通过 RDMA 子系统、NCCL/SHARP 集成、GPU Direct Storage、CXL 内存池化驱动、topology-aware 调度以及 Fabric Manager 接口，支持 InfiniBand/RoCEv2 的跨机架扩展、AllReduce 通信优化、多租户资源隔离与动态借用。然而，Scale-out 涉及更复杂的网络 Fabric 管理、拥塞控制、SHARP in-network compute、UFM 遥测与集群级调度等内容，其深度远超单节点互联范畴，因此本白皮书将不作为重点展开，而是将讨论重心严格限定在 Scale-up 场景下的操作系统角色与优化实践。后续章节将从 AI 负载画像入手，逐步剖析 CPU-GPU 异构互联、单节点多 GPU 管理、超节点 Scale-up 资源池化，直至统一内存语义演进，帮助系统架构师与运维工程师构建高效、可扩展的 AI Factory 单机架基础。

1 AI 工作负载场景与系统互联问题全景

AI 工作负载类型：训练、推理与 AI Factory

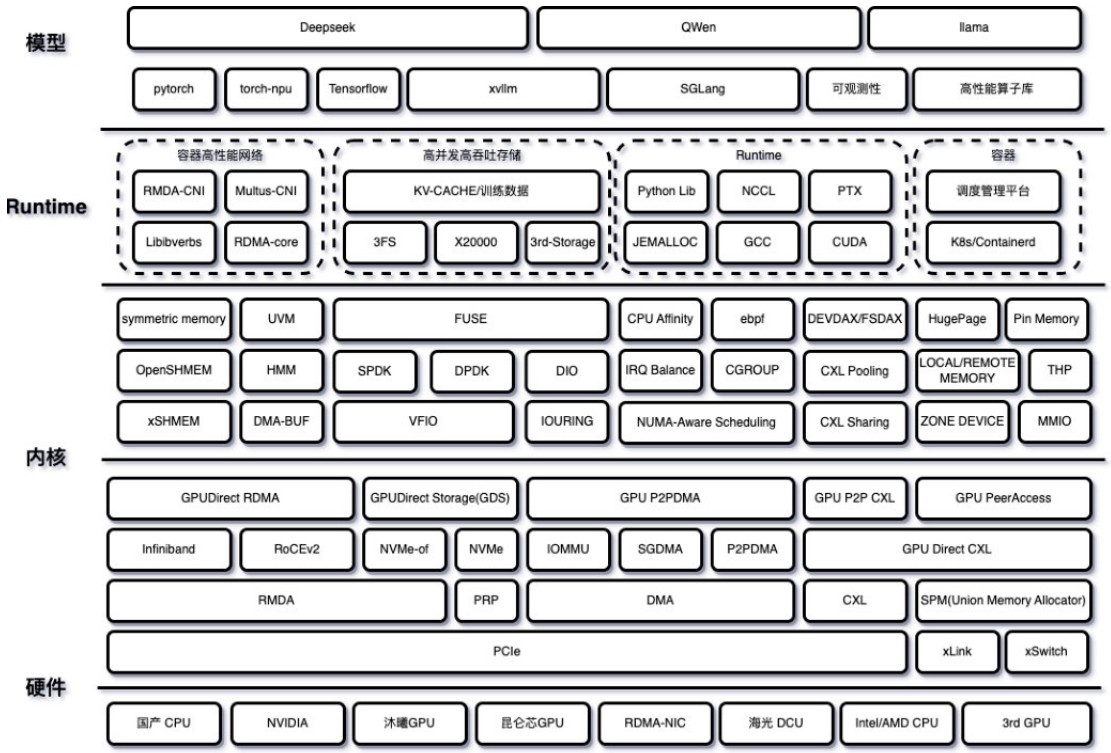


图154 OS 系统架构

AI 工作负载主要分为训练和推理两大类，前者涉及大规模参数优化（如大型语言模型 LLM 的预训练），需要海量计算资源、数据并行和模型并行处理，对高速互联（如 NVLink、CXL、高性能 RDMA 网卡）依赖极高；后者聚焦实时响应（如聊天机器人、图像识别、推荐系统），强调低延迟、高吞吐和高效的 KV Cache 访问，内存带宽和内存容量往往成为瓶颈。

随着 AI Factory 概念的兴起，这些负载往往在共享基础设施中混合运行，形成一个高效的“工厂”式生态：训练任务需要极致的计算密度和 AllReduce 通信效率，推理任务则追求高并发 QPS 和动态批处理能力。整个生态从底层硬件到上层应用形成完整的垂直栈：

- **硬件层：**国产 CPU + NVIDIA GPU + 昆仑芯/沐曦/海光 DCU 等 xPU + 高性能 RDMA 网卡 + NVLink/CXL 互联，提供异构算力底座。
- **内核与虚拟化基础：**内存优化、cgroup、ebpf、rdmacm、runc/youki 容器运行时、容器网络（multus-cni/rdma-cni）、容器镜像瘦身、GPU 可视观（nvidia-smi/spiderpool）、资源编排（vGPU 引擎、离线调度、负载均衡）。
- **推理引擎与优化：**优化 ollama、vllm、mindie、sglang、dynamO、chitu 等推理框架，结合 CUDNN、NNAL、FlashMLA、DeepEP、DeepGEMM 等算子库，实现高吞吐、低延迟推理。

- **大模型支持与分布式：**DeepSeek-R1/R2/v3、Qwen、Llama 系列模型 + Ray 分布式调度，实现训练/推理混合部署。
- **上层产品与场景：**NingOS AI（AI 操作系统）、容器云&管理平台、云原生&AI 容器镜像生态，覆盖 AI For 可观测、AI For 智能运维、AI For 系统优化、AI 知识库-RAG 应用等场景。

训练负载对带宽和同步延迟极其敏感，例如 MoE 模型的专家路由和 AllReduce 操作会放大跨节点互联瓶颈，容易导致利用率下降；推理负载则更依赖 KV Cache 的快速内存访问和动态批处理，内存容量不足或跨 NUMA 延迟过高会显著影响首 token 延迟（TTFT）和 tokens/s 吞吐。AI Factory 通过动态调度、多租户隔离和资源亲和性来缓解这些问题，但互联瓶颈（如跨 socket/跨节点延迟、CXL 内存池化效率、RDMA 网络抖动）仍然是导致任务抢占、碎片化和整体利用率波动的核心挑战。未来，随着 CXL 3.0/PCIe 7.0、NVLink 5.0 和高密度 xPU 集群的普及，AI Factory 需要在系统级（内核、容器、调度）和硬件级（异构互联、内存扩展）同时优化，才能真正实现“算力工厂”级的高效共享与弹性扩展。

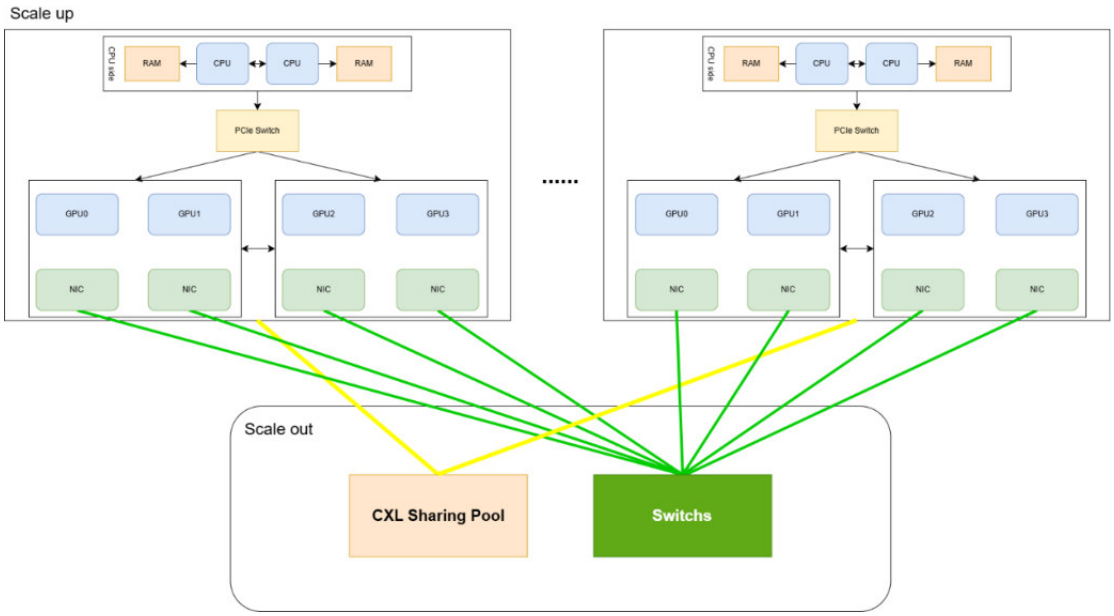


图155 Scale up 拓扑

单节点场景

在单节点环境下，使用少量 GPU（如 2-8 个）的训练场景常见于原型开发或边缘 AI 应用，而高并发推理则服务于生产环境，如 API 端点处理数千查询/秒。这里，互联问题主要体现在 CPU-GPU 数据传输和 GPU 间同步上。未优化的 PCIe 互联可能导致数据拷贝消耗大量时间，高并发推理则需借助多进程服务（MPS）来隔离负载，确保资源利用最大化，而互联瓶颈如 NUMA 效应会进一步放大延迟。例如，在一个 4-GPU 节点上运行 Llama-7B 模型的 Few-GPU 训练时，如果不绑定 CPU 亲和性，跨 socket 访问可能增加延迟；在高并发推理中，处理 1000 TPS（Transactions Per Second）的服务，如果使用 Pinned Memory 优化，可减少 CPU-GPU 拷贝时间。总体而言，单节点场景强调本地优化，如使用 NVLink 实现 GPU 间直连，以最小化 PCIe 瓶颈，从而提升整体吞吐量和响应时间。

多节点场景

大规模分布式训练涉及数百甚至数千 GPU，如训练 GPT-like 模型，需要跨节点高效通信。LLM 和 MoE 的并行策略（如数据并行、专家并行）高度依赖于网络 fabric 的带宽和延迟。在使用 InfiniBand 的万卡集群可将 AllReduce 操作时间缩短至微秒级，但 RoCE v2 在高负载下易受拥塞影响。系统互联的挑战在于平衡计算与通信开销，确保模型收敛速度不因网络瓶颈而放缓。例如，在一个 1024-GPU 集群上训练 MoE 模型时，通过 NCCL 优化将通信开销降低，这涉及环状算法和分层 AllReduce 的结合。进一步地，分布式训练还需考虑故障恢复，如 Checkpoint 机制依赖低延迟存储互联。MoE 模型的特殊性在于专家路由的动态性，这放大了对 Fabric 可靠性的需求；在 LLM 中，长序列训练则测试带宽极限。

资源池化场景

资源池化通过解耦计算、内存和存储来应对 AI 负载的动态性，例如 KV Cache 在推理中缓存键值对以加速生成，RAG（Retrieval-Augmented Generation）则处理长上下文输入，导致内存需求激增。混合负载场景下，训练和推理共存会产生资源碎片。CXL 技术可实现内存池化，允许动态分配资源，提高利用率。互联问题在这里体现为跨设备访问延迟，需要精细的调度机制。例如，在 RAG 应用中，长上下文可能要求数百 GB 内存，如果不池化，单节点容易溢出；KV Cache 池化则允许共享缓存，减少重复计算。使用 CXL Switch 将内存从 DRAM 扩展到远程池，实现了推理延迟降低效果。进一步地，资源池化还涉及存储层，如 NVMe-oF 支持 GPUDirect Storage，实现零拷贝数据路径。这不仅提升效率，还降低功耗。

系统互联与内存的共性瓶颈

从 NUMA（Non-Uniform Memory Access）到网络 Fabric 再到 CXL（Compute Express Link），互联瓶颈的核心在于非均匀访问延迟和带宽限制。NUMA 在多 socket CPU 中导致跨 socket 访问慢于本地，而 Fabric 扩展到集群级，CXL 则提供统一内存语义。这些瓶颈常导致 AI 训练的瓶颈转移从计算到 I/O，优化需从硬件拓扑到软件栈的全链路考虑。例如，NUMA 效应在双 socket 系统中可增加访问延迟；在 Fabric 层面，InfiniBand vs Ethernet 的带宽差异影响 AllReduce 效率；CXL 则通过 Type-3 设备实现内存共享，缓解瓶颈。共性问题还包括一致性开销和功耗，CXL 的演进正逐步解决这些。总体而言，这些技术从本地到全局演进，共同构筑 AI 系统的互联基础。

互联瓶颈定义操作系统新战场

本章剖析了 AI 各类负载，揭示其共同面临的核心约束：系统互联性能已成为比单点算力更关键的瓶颈。这一瓶颈贯穿于节点内的 PCIe 与 NUMA 拓扑、集群间的 RDMA 网络效率，以及资源池化中 CXL 等技术引入的非均匀访问延迟。其影响直接表现为训练周期延长、推理响应受损与资源碎片化加剧。

面对这一全景式挑战及异构硬件的复杂生态，操作系统的核心使命已发生根本转变：它必须从传统资源管理器，演进为能够统一调度异构互联、智能管理非均匀资源并深度适配多元化算力底座的“AI 基础设施中枢”。这一定位重塑了其所有技术演进的方向。

2 CPU-to-CPU 同构互联

AI 场景中的 CPU 角色与典型负载模式

在人工智能（AI）工作负载中，中央处理器（CPU）扮演着多重关键角色，不仅作为系统的“指挥中心”负责整体协调，还直接参与数据预处理、模型调度和后处理任务。这些角色在训练和推理阶段各有侧重：在训练过程中，CPU 主要处理数据集的加载、增强和分发，例如通过多线程管道（如 PyTorch 的 DataLoader）将图像或文本数据高效传输到 GPU；在推理阶段，CPU 则负责查询路由、批处理和结果聚合，如在高并发聊天应用中协调数千个推理请求。

从负载模式来看，AI 场景下的 CPU 往往呈现出高 I/O 密集型和计算辅助型特征。高 I/O 模式常见于数据密集型任务，例如在大型语言模型（LLM）预训练中，CPU 需从存储系统中读取 TB 级数据集，并进行 tokenization 和 shuffle 操作，这要求高效的内存管理和 I/O 调度。计算辅助模式则体现在模型并行中，CPU 辅助 GPU 进行参数同步或梯度聚合，尤其在混合精度训练中处理浮点转换。

进一步分析典型负载，可以看到周期性峰值模式：在训练迭代中，CPU 负载在数据准备阶段峰值，而在计算阶段相对空闲；在推理服务中，负载更均匀，但突发查询可能导致瞬时峰值。举例而言，在处理 MoE（Mixture of Experts）模型时，CPU 需动态路由专家模块，这增加了分支预测和缓存利用的压力。

AI 工作负载下的 NUMA 效应与性能瓶颈

Non-Uniform Memory Access（NUMA）效应是多 socket CPU 架构的核心挑战，在 AI 工作负载中表现为远程内存访问延迟远高于本地访问，导致性能瓶颈显著放大。具体而言，在双 socket 或多 socket 系统中，跨 socket 访问需通过互联链路（如 UPI 或 Infinity Fabric），延迟可达本地访问的多倍，这在数据并行训练中尤为突出：当一个 socket 的 CPU 线程访问另一个 socket 的内存时，会引入额外的一致性开销和带宽竞争。

在 AI 场景下，这些瓶颈往往体现在几个关键方面。首先是数据加载瓶颈：在 LLM 训练中，数据集分布不均可能导致跨 socket 拷贝总时间变长，未优化的 NUMA 配置使 ImageNet 数据管道效率下降严重。其次是同步开销：在分布式框架如 TensorFlow 的 Horovod 中，跨 socket 的 AllGather 操作会因 NUMA 放大通信延迟，影响模型收敛速度。第三是资源争用：高并发推理中，多线程竞争共享缓存会导致 L3 缓存失效率上升，进而增加功耗和热量。

通过深入剖析，NUMA 瓶颈还与负载特性交互：例如，在 MoE 模型的专家并行中，不均匀分配会放大跨 socket 流量；在长序列 RAG（Retrieval-Augmented Generation）中，内存膨胀进一步压力测试 NUMA 带宽。为缓解，建议采用亲和性绑定和内存交织策略。

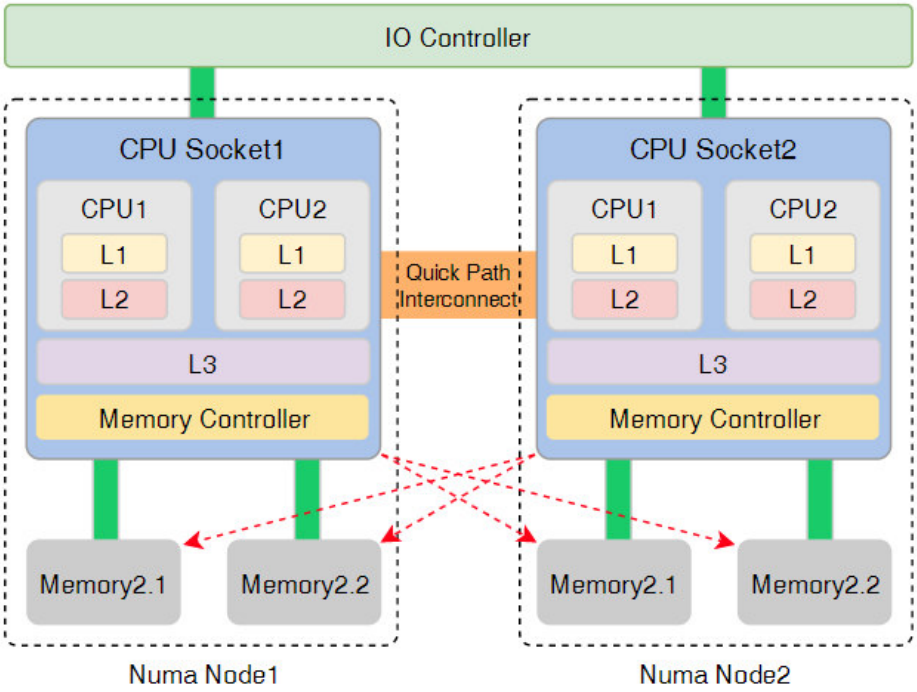


图156 CPU NUMA 拓扑

主流硬件技术对比

Intel Ultra Path Interconnect（UPI）和 AMD Infinity Fabric 代表了当代 CPU 同构互联的两大主流技术，前者强调高带宽点对点连接，后者采用模块化 chiplet 设计，提供更灵活的可扩展性。UPI 在每个链接上提供高达 12.8 GT/s[2]的速率，支持双 socket 带宽达数百 GB/s，特别适合要求低延迟一致性的 AI 任务，如实时推理调度；Infinity Fabric 则通过 die-to-die 链接实现更高聚合带宽，在多 socket 扩展中表现出色，尤其适用于大规模数据预处理。

从 AI 性能视角看，这些技术的差异显著影响负载效率。UPI 在 NUMA 敏感的 PyTorch 训练中延迟更低，得益于其优化的缓存一致性协议（MESIF 变体），但在高线程数场景下易受链接数限制；Infinity Fabric 则在 AMD 平台上展现出更好的均匀性，在 Mixtral 模型训练中吞吐量更高，因为其 fabric 支持动态路由和 GPU 集成。功耗方面，UPI 更高效（每 GB/s 功耗较低），而 Infinity Fabric 在 chiplet 架构下虽灵活，但需精细管理热分布。

进一步对比拓扑和演进：UPI 采用环状或全网格拓扑，易于 OS 感知；Infinity Fabric 向 3D 堆叠演进，支持未来异构集成。在 AI 应用中，选择取决于场景：Intel 平台适合一致性密集型负载，AMD 则青睐扩展性强的集群。

Linux 内核 NUMA 感知机制

Linux 内核通过 NUMA 节点抽象高效管理多 socket 系统，支持自动和手动优化机制，以最小化跨 socket 访问开销。核心组件包括内存策略（mpol，如 localalloc 确保本地分配）和迁移守护进程（numad），后者动态监控负载并迁移页以平衡节点。在 AI 负载下，这些机制至关重要：例如，在高 I/O 训练中，localalloc 可将数据加载延迟降低很多。

内核的 NUMA 感知还体现在调度器中，CFS（Completely Fair Scheduler）考虑 NUMA 亲和性，避免线程跨节点迁移；sysfs 接口（如/sys/devices/system/node）允许查询拓扑和统计。启用 numad 在 TensorFlow 多线程预处理中缓存利用率大大提高。高级特性如 weighted interleaving（在内核 5.x+引入）预演 CXL 池化，支持根据权重分配内存，适用于混合 AI 负载。

AI 框架集成

AI 框架与 Linux NUMA 机制的集成是优化性能的关键，PyTorch 通过 torch.set_num_threads 和 torch.utils.cpu_affinity 绑定线程到特定 NUMA 节点，减少跨 socket 访问；TensorFlow 使用 tf.config.threading.set_inter_op_parallelism_threads 结合 numactl 外部绑定；JAX 则原生支持 jax.config.update('jax_platform_name', 'cpu')并集成 numactl。在集成中，这些绑定可将 NUMA 开销降低很多。

例如，在 PyTorch 的 DistributedDataParallel 中，未绑定时数据管道跨 socket 会导致瓶颈；TensorFlow 的 Eager 执行受益于本地分配，在高并发推理中响应时间大大缩短。

性能调优与基准测试

性能调优依赖工具链：hwloc 发现拓扑、likwid 测量带宽/计数、perf 跟踪事件。在 Llama 训练案例中，用 hwloc-ls 映射 NUMA 节点，再 likwid-perfctr 监控跨 socket 流量，结果显示优化后带宽利用上升；perf stat 捕获 L3 失效，揭示瓶颈。

在 Mixtral 中，工具链分析 MoE 路由：likwid 显示专家分配不均导致峰值流量，优化绑定后效率提升。

监控与故障排除

监控工具包括 numastat（节点统计）、numad（自动平衡）和 sysfs（拓扑查询）。故障排除流程：1. numastat 识别高远程访问；2. sysfs 查询节点；3. numad 或手动绑定修复；4. 验证性能。

在 AI 中，常见瓶颈如数据倾斜：在推理中，用 numastat 定位，修复后延迟降低。

3 CPU-to-GPU 异构互联

AI 场景中的 CPU-GPU 协同与数据路径画像

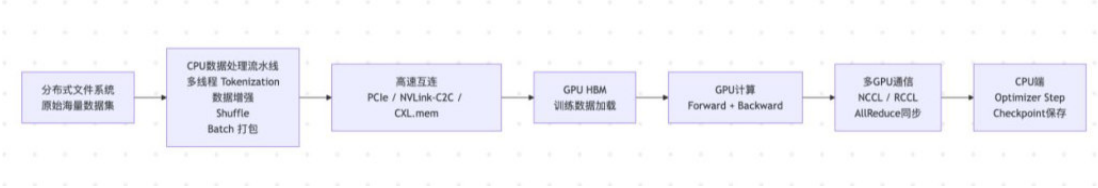


图157 CPU 与 GPU 数据流向

在当代大规模人工智能系统中，CPU 与 GPU 的协同早已超越了传统的“主机+加速卡”简单分工，而是形成了一种高度耦合、互为依赖的异构计算闭环。CPU 作为全局大脑，负责数据管道的构建、任务调度、I/O 协

调、预处理与后处理，而 GPU 则专注于张量运算、矩阵乘法和注意力机制的核心加速，这种分工在不同工作负载下呈现出截然不同的数据流动特征。在大规模预训练场景中，典型路径从 CPU 侧开始：从分布式文件系统读取原始海量数据集，经过多线程 tokenization、数据增强、shuffle 和 batch 打包后，通过 PCIe、NVLink-C2C 或 CXL.mem 将数据高效传输到 GPU 的 HBM 中，随后 GPU 执行前向与反向传播，期间通过 NCCL/RCCL 完成跨 GPU 的 AllReduce 同步，最后梯度或参数更新片段再回传 CPU 用于优化器步进或 Checkpoint 保存。

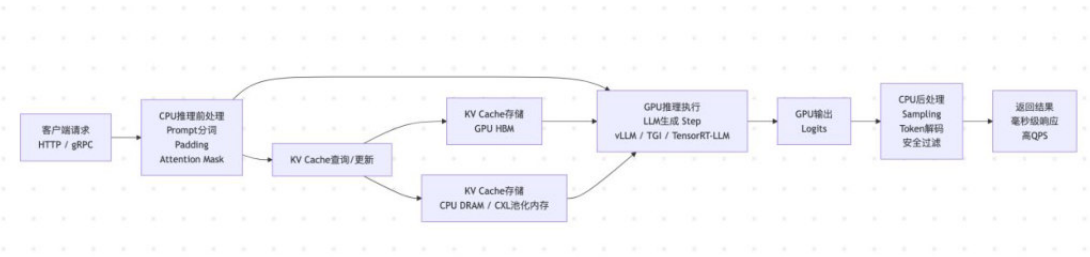


图158 推理数据流向

而在高并发在线推理（如 vLLM、TGI 或 TensorRT-LLM 部署）中，路径则更强调低延迟与实时性：CPU 首先接收 HTTP/gRPC 请求，对 Prompt 进行分词、padding 和 attention mask 生成，同时查询或更新 KV Cache（部分 KV Cache 可能驻留在 CPU 侧 DRAM 或 CXL 池化内存），随后将 token 序列和必要状态指针传输给 GPU 执行生成步骤，输出 logits 回传 CPU 进行采样、解码和安全过滤，整个过程需要在毫秒级内完成数千 QPS。

在混合负载（如训练+推理共存、RAG+微调并行）下，数据路径进一步复杂化，CPU 必须同时管理长上下文检索（FAISS/Milvus）、动态 batch 调度、资源隔离（MIG/MPS/cgroup）和跨设备内存迁移。这些画像揭示了异构互联的核心价值：它不仅是带宽通道，更是决定 AI 系统整体效率与响应性的关键枢纽，CPU-GPU 协同已从单纯加速转向真正统一的内存语义与动态资源调度。

PCIe Gen5/6 与 CXL 的异构互联演进

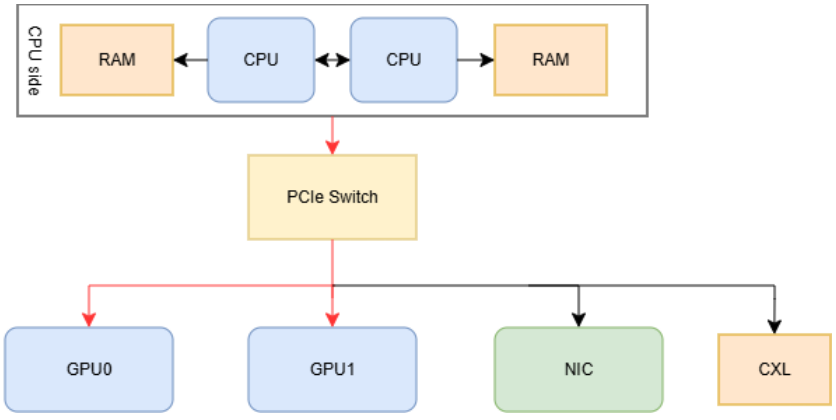


图159 CXL 拓扑

PCIe 与 CXL 代表了当前异构互联的两条主要演进路线，前者以原始传输带宽和通用性为核心，后者则通过引入缓存一致性与内存语义扩展彻底改变了 CPU-GPU 协同范式。PCIe Gen5（32 GT/s）已成为 2024 -

2025 年主流 AI 服务器的标准接口，x16 配置下双向有效带宽约 110 – 115 GB/s，受协议开销（FLIT、CRC、重传）限制，实际 AI 负载中难以突破 120 GB/s 天花板；PCIe Gen6（64 GT/s）在 2025 年底开始规模量产，x16 双向带宽翻倍至 256 GB/s，采用 PAM4 调制 + 前向纠错（FEC），虽然延迟略升 15–20 ns，但对 HBM↔系统内存大批量拷贝场景的缓解作用显著，尤其在多 GPU 单节点 All-to-All 通信中能大幅降低瓶颈，并为 AI 训练提供更高吞吐。CXL 则从根本上重塑了互联语义：CXL 1.1/2.0 基于 PCIe 物理层引入 CXL.io（兼容 PCIe）、CXL.cache（设备可缓存 CPU 内存）和 CXL.mem（CPU 可直接寻址设备内存），CXL 2.0 支持 switch fabric 与多主机共享；到 CXL 3.0（截至 20226 未大规模落地）基于 PCIe Gen6 物理层，带宽翻倍至 x16 256 GB/s，并新增 switch-based pooling、weighted interleaving、内存分级（near/far）、TSP 安全协议以及 Type-2/3 设备支持，实现多主机共享同一内存段的硬件一致性、P2P 传输绕过 CPU 以及多级切换拓扑（支持数千节点 fabric）。

厂商实现差异

三大主流厂商在 CPU-GPU 异构互联上的实现路径各具特色，形成了鲜明的技术分化。NVIDIA 的 NVLink-C2C（Chip-to-Chip）是当前性能标杆，在 Grace-Hopper Superchip、GB200 和后续 Rubin/Blackwell Ultra 架构中提供 900 GB/s 至 1.8 TB/s 的双向带宽，延迟低至 15 – 20 ns[3]，并实现全硬件缓存一致性（兼容 CHI 协议）；2025 年推出的 NVLink Fusion 进一步开放接口，允许 Fujitsu、Qualcomm、Intel 等第三方 CPU 通过 NVLink-C2C 直连 Blackwell/Rubin GPU，形成半定制化 AI 基础设施，在 GB200 NVL72 等系统中实测显示 CPU-GPU 数据路径延迟比 PCIe Gen5 低 7 倍，能效比提升 5 倍以上。AMD 的 Infinity Fabric GPU 直连（XGMI）则依托 chiplet 架构的天然优势，在 MI300X/MI325X/MI355X 系列上实现 448 GB/s – 1 TB/s 的聚合带宽，通过第五代 Infinity Fabric 的 mesh-like 拓扑确保带宽均匀性，尤其在 MoE 模型的动态路由阶段表现平稳；2025 – 2026 年 AMD 通过 Helios 机架级 fabric 将 Infinity Fabric 从单节点扩展到整机架，实现 CPU-GPU-加速器统一互联，其开放生态（ROCm）与高可扩展性成为吸引开源社区的重要因素，MI325X 在 8-GPU 平台上提供 2 TB HBM3E 共享内存和 48 TB/s 聚合带宽。Intel 的 Xe Link（应用于 Data Center GPU Max 系列）基于 PCIe 物理层 + 专有协议，提供 128 – 512 GB/s 带宽，支持硬件一致性，2025 年后通过与 NVIDIA 的合作开始兼容 NVLink Fusion，允许 Xeon CPU 与 NVIDIA GPU 混合部署，在 oneAPI 生态中表现出色，适合企业级混合工作负载。从实际性能对比看，NVLink-C2C 在极致延迟与带宽上领先（GB200 NVL72 在 Llama 3.1 405B 推理中达 224 tokens/s），AMD XGMI 在扩展性和 MoE 负载中更均衡（MI325X 峰值 FP16 性能超 H100 30%+），Intel Xe Link 则在生态兼容性与企业集成上占优。

OS 驱动与运行时支持

操作系统与驱动层是异构互联从硬件到应用的桥梁，直接决定了实际可用性和性能天花板。NVIDIA CUDA Driver（R550/R560 系列及以上）全面支持 NVLink-C2C、GPUDirect RDMA over dma-buf（已无需 nv_peer_mem 模块）、HMM 扩展以及 cudaMallocAsync/cudaGraph 对长序列推理的零拷贝优化，2025 – 2026 年重点增强了 GPUDirect Storage v3 与 NVMe-oF 集成，实现从存储直达 GPU 的端到端路径，并在 Blackwell 平台上支持 NVFP4 量化加速推理。AMD ROCm 大幅补齐 Windows/Linux 双栈支持，XGMI 直连完整暴露给 HIP runtime，dma-buf 与 HMM 支持已趋成熟，但一致性开销仍略高于 NVLink-C2C。Intel oneAPI Level Zero 作为原生 API 直接映射 Xe Link 与 CXL 特性，2025 年后因与 NVIDIA 合作而增强跨厂商兼容性，在 dma-buf 和 HMM 支持上最完整（Intel 长期主导内核相关开发）。Linux 内核的 dma-buf 是用户态缓冲区共享的基础，HMM（Heterogeneous Memory Management，从内核

5.10 开始逐步完善）允许 GPU 直接访问系统内存页而无需显式拷贝，2025 年关键补丁包括对 device-private pages 的 P2P RDMA 支持以及页面迁移延迟的大幅优化，CONFIG_PCI_P2PDMA 已成为现代 GPUDirect RDMA 的推荐路径，取代了旧的 nvidia-peermem 模块。这些驱动与内核机制的协同，使得从传统显式搬运向统一内存访问的过渡成为现实，也为后续 CXL 池化与 NVLink-C2C 统一语义奠定了坚实基础。

拓扑发现与配置

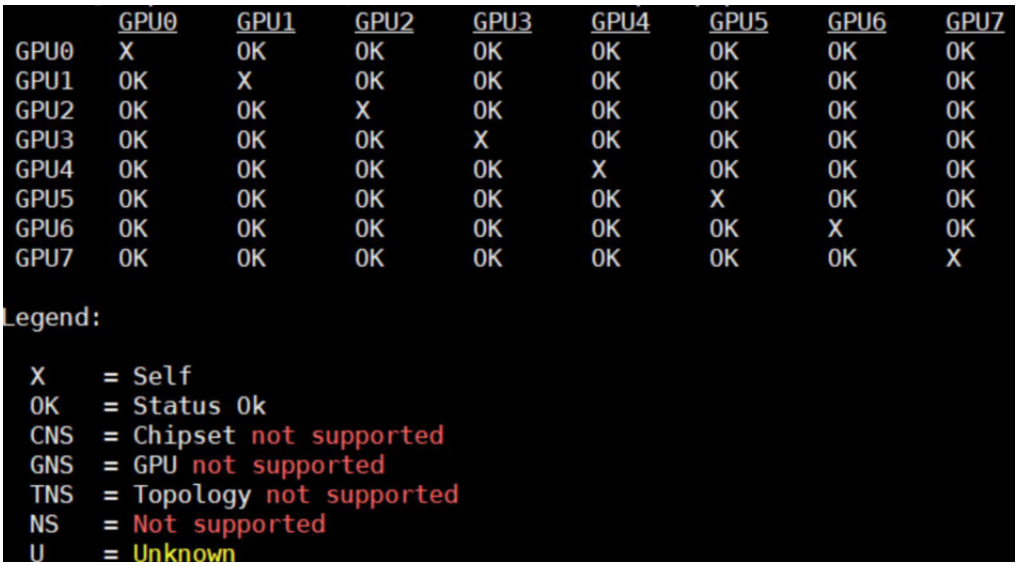


图160 GPU 拓扑

异构拓扑的准确发现与精细配置是单节点多 GPU 系统性能优化的起点，整个过程依赖多层工具与接口的协同。在硬件层面，nvidia-smi topo -m、rocm-smi --showtopo 和 level-zero 的 zeDeviceGetProperties 可直接输出 GPU 间的链路类型（NVLink、XGMI、PCIe）、带宽与延迟；在 OS 层面，hwloc（lstopo）结合 lspci -vvv | grep CXL 能完整映射 NUMA 节点、PCIe 总线和 CXL 端口拓扑，/sys/bus/cxl/devices 则提供 CXL 设备的实时状态；在驱动与运行时层面，nccl-tests 的 all_reduce_perf -t cuda -b 8 -e 1G -f 2 可量化实际带宽，/sys/kernel/debug/dma-buf 显示共享缓冲区信息。生产级配置流程通常包括：首先通过 hwloc-dump 生成 XML 拓扑文件供后续调度参考，然后使用 numactl --cpunodebind 和 --membind 绑定 CPU 线程与内存节点，再通过 nvidia-smi nvlink --setenable 1 或 cxl-cli enumerate 激活高速链路，最后设置环境变量如 export NCCL_IB_DISABLE=0（强制 GPUDirect RDMA）、export NCCL_P2P_LEVEL=NV（优先 NVLink P2P）来引导通信库选择最优路径。在复杂多 GPU 节点中，还需结合 nvidia-fabricmanager 守护进程动态管理 NVSwitch 状态，确保拓扑变化时（热插拔、MIG 重配置）系统自动重新枚举。这些步骤看似琐碎，却能将实际通信效率从 50% 提升至 90% 以上，是所有单节点优化的基石。

AI 工作负载优化

AI 工作负载的性能优化高度依赖于 CPU-GPU 数据路径的精细打磨，其中数据预取、Pinned Memory 和 GPUDirect RDMA 零拷贝路径是三板斧级别的核心技术。在训练场景中，数据预取通过 CUDA Stream +

cudaMemcpyAsync 实现计算与传输的重叠，Blackwell 平台的 Tensor Memory Accelerator 进一步引入硬件预取引擎，根据历史访问模式自动预测下一批 token 或梯度块；在推理场景中，vLLM 的 paged attention 机制结合预取可将 KV Cache 加载延迟从数百微秒降至数十微秒。Pinned Memory（cudaMallocHost / cudaHostRegister）将主机内存页锁定在物理地址，避免页面故障和分页开销，在 PCIe Gen5 系统上可将主机到设备拷贝带宽从 50 GB/s 提升至 110 GB/s 以上，但会占用宝贵的系统 DRAM，需要结合 hugepages 和 cgroup 内存限额谨慎使用。GPUDirect RDMA 是零拷贝路径的巅峰，通过 dma-buf 将 GPU 内存直接暴露给 NIC（InfiniBand/RoCE），实现从存储 → GPU 或 GPU → GPU 的直接传输，绕过 CPU 内存拷贝，在万卡集群的单节点 AllReduce 中可将通信时间大大缩短，2025–2026 年 dma-buf 路径已成为 NCCL/RCCL 的默认选项，无需额外 nv_peer_mem 模块。这些优化技术的协同，不仅大幅压缩了 I/O 瓶颈，还为统一内存语义的全面落地铺平了道路。

4 单节点多 GPU 互联

单节点多 GPU 训练与推理场景画像

在人工智能基础设施中，单节点多 GPU 配置已成为从原型开发到生产部署的核心单元，通常涉及 4–8 张 GPU 的标准服务器（如 DGX H200 或 GB200），或扩展至 16–72 张的超节点系统（如 NVL72），这些场景不仅支持中小模型的全参数训练，还能处理大模型的高效微调和高并发推理服务。在训练方面，典型画像包括 7B–70B 参数规模的模型如 Llama-3-70B 或 Mixtral-8x22B，在 8×H200 平台上采用混合精度（BF16/FP8）进行全量预训练或微调，batch size 范围 1–16，强调数据并行（DP）和张量并行（TP）的结合，以最大化 GPU 利用率，同时应对显存压力通过 ZeRO-Offload 将部分参数迁移到 CPU DRAM 或 CXL 池化内存。

在推理场景中，高并发服务如 vLLM 或 TensorRT-LLM 部署针对 70B–405B 模型，支持 1000–5000+ QPS，采用 continuous batching、paged attention 和 KV Cache 量化技术，长上下文处理可达 128k–1M tokens，混合负载则允许训练与推理共存，一部分 GPU 用于持续微调（如 LoRA/QLoRA/DoRA），另一部分处理生产查询，需要精细的资源隔离以避免干扰。

进一步剖析这些场景的瓶颈与优化点，训练负载高度依赖 AllReduce/AllGather/ReduceScatter 操作的效率，这些通信往往占大量的时间，在 MoE 模型中专家路由的动态性进一步放大 P2P 拷贝需求，而推理则更注重 KV Cache 管理的低延迟，RAG 应用中长上下文会导致显存膨胀至数百 GB，迫使系统引入动态内存 tiering（HBM → DRAM → SSD）。在混合负载中，资源碎片化问题突出，MIG/MPS 隔离虽有效但引入额外开销。总体而言，单节点场景的画像不仅反映了硬件演进（如从 H100 到 GB200 的带宽跃升），还强调软件栈（如 PyTorch/DeepSpeed）的适应性，帮助从业者从实际痛点入手设计高效 AI 系统。

单节点 GPU 互联技术栈

单节点 GPU 互联技术栈的核心在于实现高效的 GPU 间通信和数据共享，NVIDIA 的 NVLink 4th/5th Gen 结合 NVSwitch 是当前主导方案，NVLink 4（H100/H200）提供单链路 100–200 GB/s 双向带宽，全节点聚合可达 3.2–4.8 TB/s，而 NVLink 5（Blackwell/GB200）翻倍至 200–400 GB/s[4]，支持更强的缓存一致性和 P2P 原子操作，NVSwitch 作为专用交换芯片确保全互联拓扑，允许任意 GPU 间直接传输而无需 CPU 中转，在 Rubin 时代预计聚合带宽突破 20 TB/s。AMD 的 Infinity Fabric 3（XGMI）采用 chiplet 模块化设计，在 MI300X/MI325X 上实现 128–256 GB/s 单链路带宽，全节点聚合 2–4 TB/s[5]，其 mesh 拓

扑在均匀负载下表现出色，尤其适合 MoE 专家并行，且与 CPU Infinity Fabric 无缝融合，支持异构直连。PCIe Peer-to-Peer(P2P)作为通用 fallback，在无专用 Fabric 的服务器中依赖 dma-buf 和 GPUDirect，实现 ~64 GB/s（Gen5 x16）或 ~128 GB/s（Gen6 x16）带宽，虽然延迟较高（~100 ns），但兼容性强，适用于入门级或混合厂商系统。

在实际部署中，这些技术栈的演进还涉及功耗与热管理，NVSwitch 在高负载下功耗可达数百瓦，需要精细的 OS 调度，而 AMD Fabric 的 chiplet 设计虽灵活但一致性开销稍高，PCIe P2P 则易受总线竞争影响。

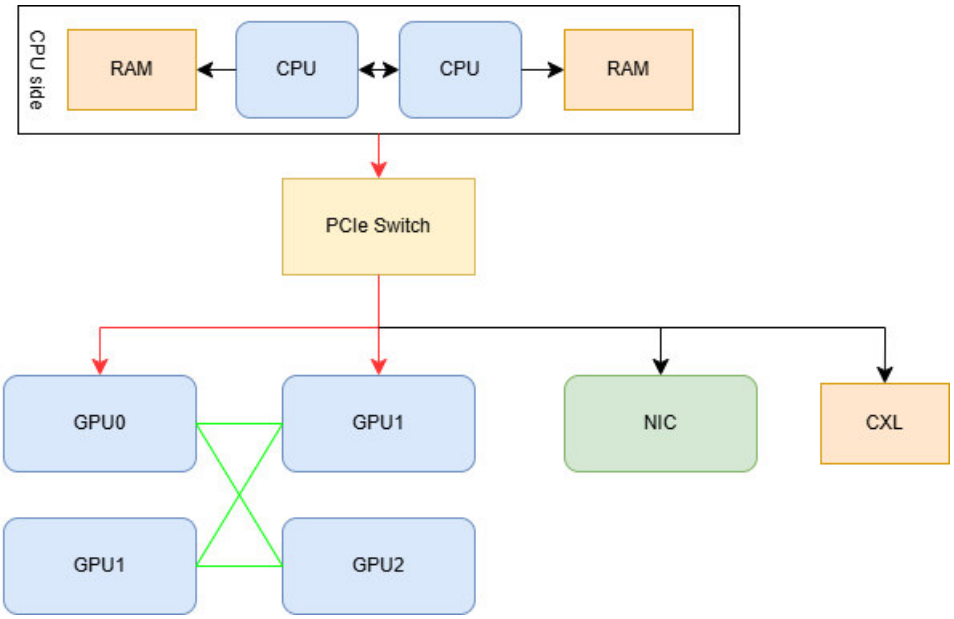


图161 GPU 互联

拓扑类型与带宽计算

单节点 GPU 拓扑类型直接决定了通信效率和可扩展性，全互联（All-to-All）是理想方案，在 DGX H200 上通过 NVSwitch 实现，每对 GPU 独立链路带宽 200 GB/s，8 GPU 配置聚合 4.8 TB/s[6]，在 GB200 NVL72 中扩展至数十 TB/s（~130 TB/s），带宽计算公式为总聚合 = GPU 数 × (链路数/2) × 单链路双向带宽，实际利用率依赖 NCCL 算法选择。树状拓扑常见于 AMD MI325X，采用分层多跳结构，同组 GPU 带宽高但跨组衰减，适合层次化并行如 pipeline parallelism，环状拓扑在早期 A100/H100 部分配置中使用，通信 hop 数多导致延迟累积，已被全互联取代。

进一步计算带宽时，需考虑协议开销和负载不均，全互联的均匀性在 DGX H200 上使 AllGather 操作时间从 200 μs 降至 50 μs，树状在大规模 GB200 中通过优化分支因子支持数千 GPU 扩展，环状则仅适用于小节点以节省成本。

OS 与通信库支持

操作系统和通信库是单节点互联的软件基石，NCCL 2.x/3.x 是 NVIDIA 首选集体通信库，支持 NVLink/NVSwitch 的 topo-aware 算法（如 Ring/Tree/Double Binary Tree），NCCL 3.x 引入分层 AllReduce 和 NVLink P2P 优化，在 Linux 6.x 上与 ib_core 集成实现 GPUDirect RDMA。RCCL 作为 AMD ROCm 对应物，针对 XGMI 优化算法兼容 NCCL，oneCCL（Intel）则提供跨厂商接口，常作为

Horovod/DeepSpeed 的 backend。Linux ib_core + GPUDirect 提供底层 RDMA 子系统，通过 dma-buf 启用 GPU 间零拷贝 P2P，CONFIG_IB_GPU_DIRECT 在内核 6.8+ 中成熟，支持 CXL 扩展。

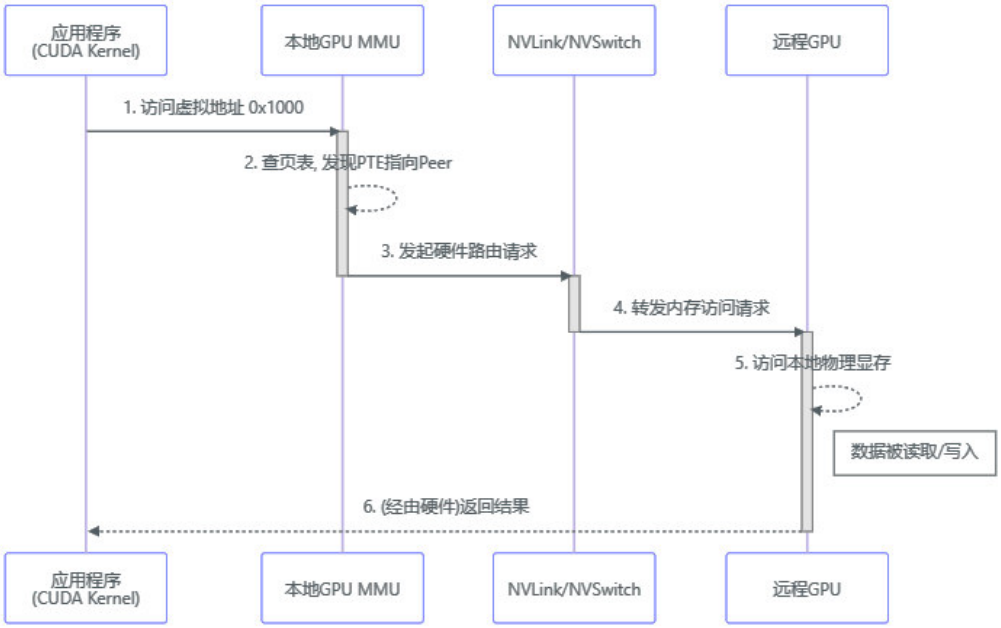


图162 CUDA 访问 CPU 过程

AI 框架原生集成

AI 框架如 PyTorch Distributed 以 NCCL backend 为基础，支持 torch.distributed.init_process_group(backend='nccl')，自动感知 NVLink 拓扑进行数据/张量并行。Megatron-LM 和 DeepSpeed 深度优化单节点 AllReduce，DeepSpeed ZeRO-Infinity 通过 offload 减少显存 4–8×，Megatron 的 flash attention 与 sequence parallel 降低通信量。框架优化的关键在于算法选择，DeepSpeed 的 pipeline + tensor parallel 在 MoE 中将 AllReduce 开销降 25%，与 OS cgroup 协同限制资源[7]。

5 超节点 GPU 互联

超节点和 Super POD 架构的强大效能并非仅源于硬件的简单堆砌。真正将这些异构、高速的组件——国产 CPU、各类 xPU、高速网络与池化内存——凝聚成一个高效、稳定、可扩展的“AI 算力工厂”的，是现代操作系统中一系列深邃而关键的技术模块。

它们如同精密的神经系统与调度中枢，深入内核，驱动硬件，将物理上的复杂互联拓扑，转化为对上层应用简洁、安全且高效可用的算力资源。接下来剖析一下这些构成 AI 时代操作系统核心竞争力的关键技术。

OS 内核态 GPU 互联协议栈支持

在超节点（SuperPOD）架构中，操作系统内核通过原生集成底层硬件协议，构建了支撑 GPU 间高速互联的协议栈基础。这首先依赖于对 PCIe 管理这一复杂子系统的精确掌控。从开机自检开始，操作系统的 PCIe 核心层就负责枚举由 CPU、GPU、CXL 交换机及各类加速卡组成的复杂拓扑中的每一个设备，为它们分配唯一的地址空间和中断资源，并优化数据通路，这是整个系统高速 I/O 的底层基石和拓扑感知调度的信息来源。

内核直接加载并管理 PCIe、RDMA 及 CXL 等物理接口的驱动模块，将底层能力抽象为统一的系统调用接口。其中，P2P 权限控制 是实现设备间直接数据流通的“通行证管理器”。操作系统驱动负责探测并验证 GPU 之间、GPU 与 CXL 设备间通过 NVLink 或 PCIe 的直连能力，在应用程序请求时建立安全的直接内存访问窗口，确保数据能沿授权的“快速通道”流动，避免冲突或越界。

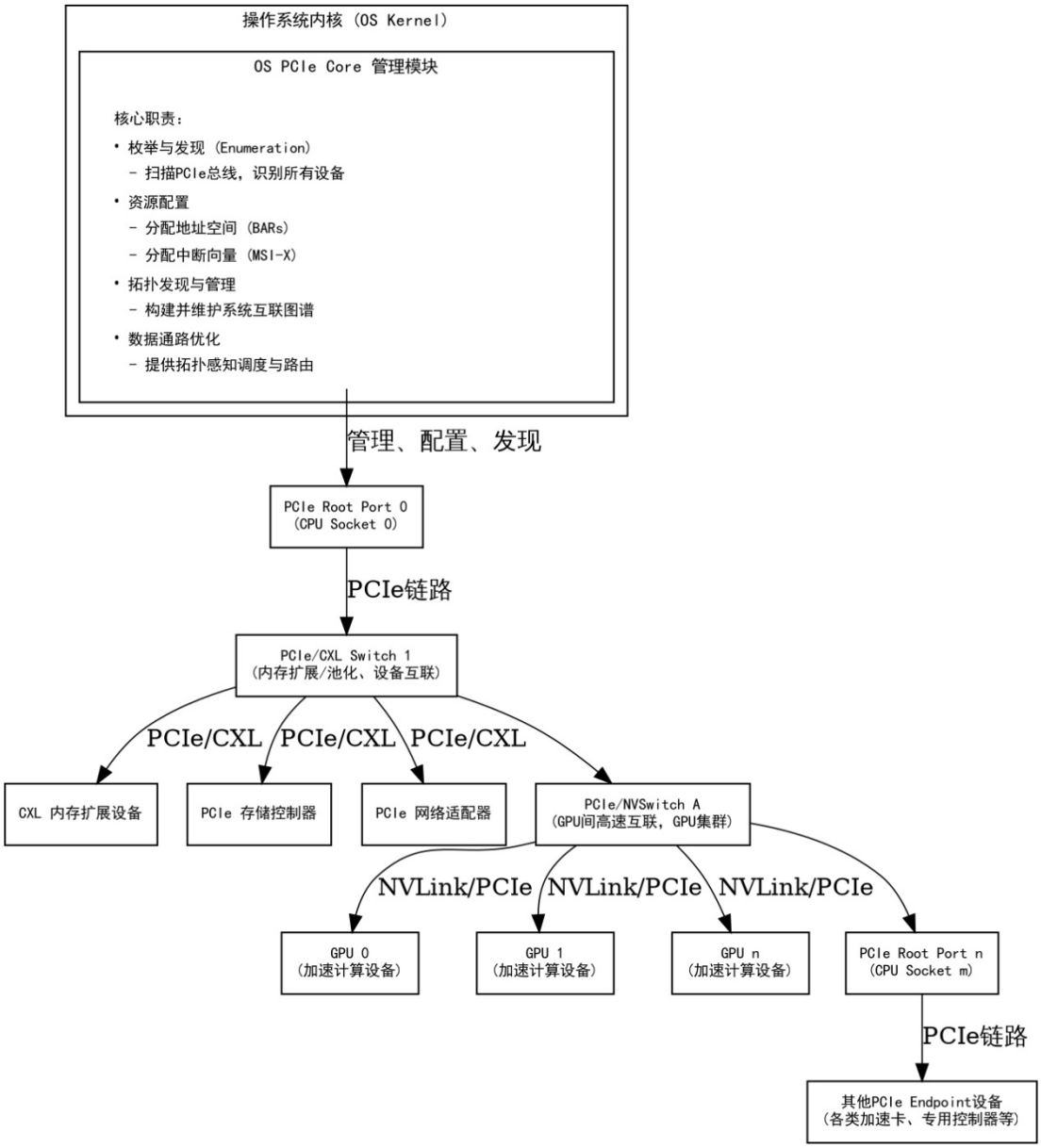


图163 PCIe 管理架构图

对于跨节点通信，GPU Direct RDMA 技术至关重要。它允许不同服务器上的 GPU，通过支持 RDMA 的高性能网络直接读取彼此显存。这需要操作系统的网络驱动、RDMA 协议栈与 GPU 驱动深度配合，共同将 GPU 显存缓冲区安全地“注册”给本地的 RDMA 网卡，实现数据从 GPU 显存到远端 GPU 显存的直接传输，完全绕过两端 CPU 和内存，将跨节点通信延迟降至最低，是构建万卡集群的关键。

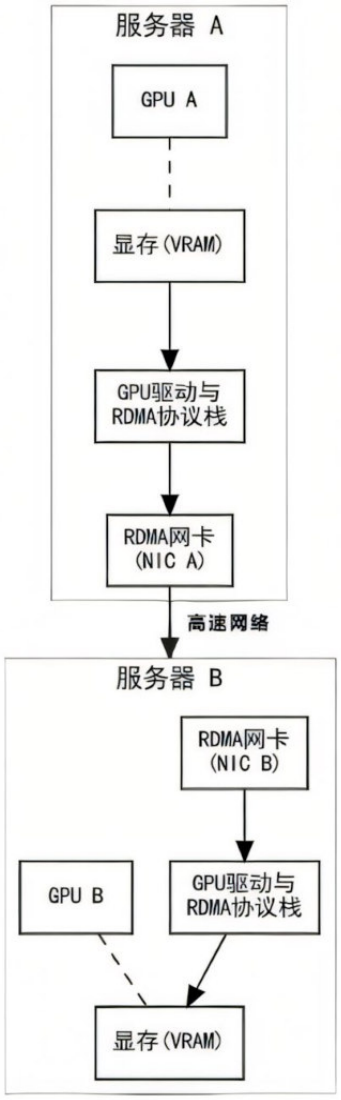


图164 GPU Direct RDMA 架构图

这种抽象机制屏蔽了不同厂商硬件驱动的差异性，使得上层 AI Runtime 无需关心底层物理介质的具体实现，仅需通过标准化的内核 API 即可发起跨节点的内存访问与数据通信请求，从而实现了异构硬件环境下的统一视图与互操作性。

针对大规模分布式训练场景，内核态协议栈实现了全链路零拷贝（Zero-Copy）数据传输路径。通过优化 DMA 映射机制，内核能够直接将 GPU 显存物理地址注册至网络接口卡或 CXL 交换机的地址空间，数据包在传输过程中完全绕过 CPU 内存缓冲区。DMA 作为“数据搬运的自动传送带”，允许外设在 CPU 指令下自行搬运大批量数据，将 CPU 从繁重拷贝中解放出来。其高级形态 SGDMA（Scatter-Gather DMA）则能通过描述符链表自动处理一系列不连续的数据块搬运，进一步减少了设备与 CPU 的交互次数，提升了 I/O 效率。配

合内核级的中断聚合策略与无锁环形缓冲区技术，系统有效降低了上下文切换频率与中断处理开销，显著减少了微秒级的通信延迟，确保在万卡规模下跨机架、跨节点的 GPU 集群能够维持线速的数据吞吐能力。

此外，内核协议栈内置了自动化的拓扑发现与资源协商机制，能够动态识别 Super POD 内的 GPU 互联拓扑结构，并根据硬件特性自动协商最优的传输协议参数与流控策略。系统通过标准化的内核对象模型，支持对不同代际、不同架构的 GPU 及互联设备进行动态适配，自动处理地址空间映射冲突与权限校验，消除了因驱动版本不一致导致的兼容性瓶颈。同时，内核集成了细粒度的 QoS 调度算法，依据 AI 训练任务的优先级动态分配带宽资源，防止关键通信流受到背景流量干扰，为上层应用提供低延迟、高吞吐且具备强一致性的通信原语。

统一内存子系统与共享内存管理

在超节点（Super POD）架构中，操作系统统一内存子系统通过深度集成 CXL 内存管理、UVM（统一虚拟寻址）、xSHMEM 及 DMA-BUF 等关键技术，构建了跨越 CPU 与 GPU 边界的连续虚拟地址空间。CXL 驱动是将下一代异构内存生态接入系统的桥梁，它负责发现、初始化和管理通过 CXL 总线连接的池化内存设备，将它们无缝集成到系统总内存资源池中，从而突破本地内存容量限制，缓解“内存墙”瓶颈。

内核内存管理器不再将 GPU 显存视为独立的物理设备，而是将其纳入系统全局虚拟内存（VMA）的统一调度范畴，这背后离不开 SPM（统一内存分配器）这一关键软件模块的智能调度。SPM 作为“全局内存调度中心”，对上层应用抽象出巨大的统一地址空间，并根据数据的热度、访问模式及设备亲和性，智能地将数据页面分配到最合适的物理位置（如本地显存或 CXL 内存池），最大化整体内存利用率和访问效率。它利用 UVM 机制实现 CPU 与 GPU 对同一内存区域的透明访问。当应用层发起跨设备内存访问时，内核自动处理页表映射与缺页中断，确保数据在 CPU 内存与 GPU 显存之间按需迁移，从而为大规模 AI 训练任务提供逻辑上连续、物理上分布的单一地址视图，彻底消除了传统模式下显式数据拷贝带来的性能损耗。

针对高性能数据交换需求，OS 内核通过 DMA-BUF 框架 实现了多设备间的零拷贝共享机制。该机制允许 GPU、RDMA 网卡及存储控制器直接共享同一块物理内存缓冲区，无需经过 CPU 内存的中转复制。内核在分配内存时即预留 DMA 能力，并建立统一的物理地址到设备虚拟地址的映射表，使得不同硬件单元能够直接通过物理地址进行读写操作。结合 CXL Type-3 设备特性，系统进一步支持了内存池化与扩展，允许 GPU 直接访问远程 CXL 内存池，大幅突破了单卡显存容量限制，同时保持了极高的带宽利用率，满足千亿参数模型训练对海量数据吞吐的严苛要求。

在内存生命周期管理与安全隔离方面，统一内存子系统提供了细粒度的分配、映射与回收策略，而 IOMMU 是实现硬件资源安全隔离与高效直通的核心硬件保障。IOMMU 如同“交通警察”，位于 CPU 与 PCIe 设备之间，负责将设备发起的物理地址请求转换为正确的系统物理地址并进行严格的权限检查，防止某个任务或用户的 GPU 误访问或篡改其他任务的内存空间，为多租户环境下的虚拟化、容器化和安全隔离提供了硬件基础。内核采用基于引用计数的共享内存管理机制，自动追踪 xSHMEM 等通信原语中的内存使用状态，确保在多进程、多容器并发场景下内存资源的安全释放，防止内存泄漏。同时，系统支持大页（HugePage）与透明大页（THP）的自动适配，减少 TLB 缺失带来的性能抖动，优化了页表遍历效率。通过智能预取算法与异步页面回写机制，内核能够预测 AI 训练中的数据访问模式，提前将所需数据加载至高速缓存或显存中，显著降低了内存访问延迟，为超节点集群提供了高效、稳定且可扩展的内存资源底座。

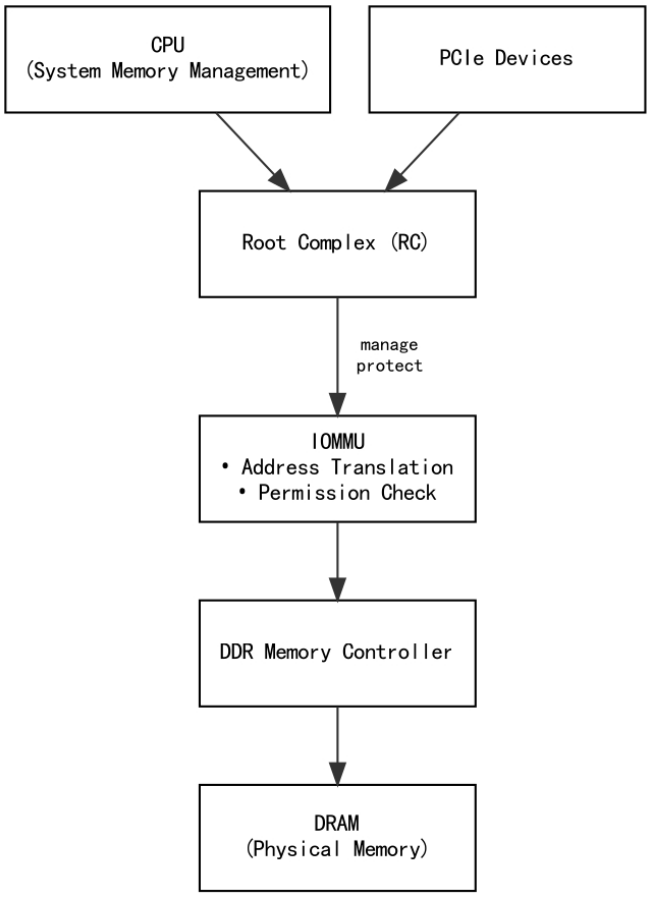


图165 OMMU 系统架构图

高性能网络栈与容器网络加速

在超节点（SuperPOD）架构中，操作系统网络栈针对容器化环境下的 AI 训练场景进行了深度重构，以突破传统 TCP/IP 协议栈的性能瓶颈。InfiniBand 与 RoCE 这两种高性能网络协议是构建低延迟集群的物理基础，而操作系统的相关协议栈和驱动则是发挥其效能的“灵魂”。无论是专有的 InfiniBand 还是基于以太网的 RoCE，其核心在于提供超低延迟、高带宽和 CPU 零参与的 RDMA 能力。内核中的 OFED 或 RDMA_CM 等协议栈实现了这些复杂网络的通信语义，使得 MPI、NCCL 等库能以近乎硬件极限的速度进行跨节点数据交换。

内核原生集成了对 RDMA-CNI 与 Multus-CNI 等容器网络插件的完整支持，使得 Kubernetes 容器能够直接挂载高性能 RDMA 网卡，绕过内核协议栈的冗余处理路径。通过在内核态实现零拷贝数据路径，网络数据包直接从应用层内存传输至网卡硬件，消除了用户态与内核态之间的多次上下文切换与数据拷贝开销。这种机制确保了在大规模分布式训练中，容器内的 GPU 进程能够以微秒级延迟和线速吞吐量进行跨节点通信，满足参数同步与梯度聚合的严苛时效要求。

为了进一步提升网络 I/O 效率，操作系统深度集成了 DPDK、io_uring 等高性能用户态网络库与内核接口。通过 io_uring 提供的异步 I/O 提交与完成队列机制，应用层能够高效地批量处理网络请求，显著降低了系统调用频率与 CPU 中断负载。同时，利用 DPDK 的用户态轮询驱动技术，关键通信路径完全脱离内核协议栈，直接在用户空间完成数据包的分发与处理，实现了确定性的低延迟传输。

在容器网络隔离、资源调度及存储扩展方面，OS 网络栈通过命名空间与 cgroup 机制实现了精细化的流量控制与资源保障。同时，NVMe-oF 技术作为操作系统的“空间折叠术”，通过内核客户端驱动，让计算节点能够像访问本地硬盘一样访问网络存储池中的 NVMe 固态盘，实现了计算与存储的解耦。这使得 AI 工厂可以弹性挂载不同性能等级的存储卷，实现存储资源的集中化管理与灵活扩展。系统支持基于硬件队列的流控策略，能够为不同的训练任务分配独立的带宽配额与优先级。结合内核级的拥塞控制算法与动态路由选择机制，网络栈能够根据实时网络拓扑与负载情况自动优化数据传输路径。此外，针对多网卡聚合场景，内核实现了智能负载均衡与故障快速切换功能，确保在链路波动或设备故障时，AI 训练任务仍能保持高可用性，维持集群的整体通信稳定性。

资源隔离与调度策略增强

在超节点（SuperPOD）架构中，操作系统通过深度集成 cgroup v2 与容器运行时接口，构建了多维度的资源隔离体系，确保多 GPU、多容器环境下的资源独占性与公平性。内核利用 cgroup 的 CPU 配额、内存限制及 IO 带宽控制功能，将每个训练任务严格限制在指定的计算资源范围内，防止“嘈杂邻居”效应导致的资源争抢。针对 GPU 资源，系统结合设备插件机制实现了显存与计算单元的精细化切分，支持将单张物理 GPU 逻辑划分为多个独立实例，或通过 MIG（Multi-Instance GPU）技术提供硬件级的隔离保障。这种隔离机制不仅保障了关键 AI 训练任务的 SLA，还有效避免了不同任务间因资源竞争引发的性能抖动，为大规模集群的稳定运行提供了基础约束。

为了最大化硬件利用率并降低通信延迟，OS 调度器实施了严格的 NUMA（非统一内存访问）感知策略与 CPU 亲和性（CPU Affinity）绑定机制。内核在进程调度时，优先将计算任务分配给与 GPU 位于同一 NUMA 节点的 CPU 核心，确保数据访问走本地内存通道，避免跨 NUMA 总线带来的高延迟与带宽损耗。同时，通过 IRQ Balance 与中断亲和性设置，系统将网卡、RDMA 设备及 GPU 产生的硬中断精准绑定至特定的 CPU 核心，避免中断风暴干扰计算线程的执行。这种细粒度的拓扑感知调度，显著减少了上下文切换开销与缓存失效（Cache Miss），使得计算密集型任务能够持续保持高吞吐状态，充分发挥超节点硬件的峰值性能。

在负载均衡与故障恢复方面，操作系统内核引入了动态负载监测与智能迁移策略。系统实时监控各计算节点、GPU 单元及网络链路的负载水位，当检测到局部热点或资源瓶颈时，调度器可依据预设策略将非关键任务迁移至空闲节点，实现集群层面的全局负载均衡。针对关键训练任务，内核支持热迁移与快速故障恢复机制，一旦检测到硬件异常或节点宕机，能够迅速将任务状态保存并重启至备用资源上，最小化训练中断时间。此外，通过结合 QoS 优先级队列与实时调度策略（如 SCHED_FIFO），系统能够确保高优先级的参数同步与梯度聚合操作优先获得 CPU 时间与 I/O 带宽，从而在复杂的并发场景下维持整体集群的高可用性与确定性性能表现。

异构计算资源抽象与设备管理

在超节点（SuperPOD）架构中，操作系统通过 VFIO（Virtual Function I/O）与 ZONE DEVICE 机制构建了统一的异构计算资源抽象层，屏蔽了底层硬件厂商（如国产 CPU、NVIDIA GPU、AMD GPU）的私有指令集与寄存器差异。VFIO 利用 IOMMU 硬件辅助功能，将物理设备直接映射到用户空间，提供安全的 DMA 访问通道，使得 AI Runtime 无需加载特定的内核驱动模块即可直接操控异构加速器。ZONE DEVICE 框架则进一步将不同架构的计算单元抽象为标准的内存区域对象，内核统一处理这些设备的内存分配、地址翻译及生命周期管理，从而在系统层面形成了一套标准化的设备访问接口。这种抽象机制有效解耦了上层应用与底层硬件的强依赖关系，显著降低了多厂商硬件混部场景下的适配复杂度，提升了系统的可移植性。

针对异构环境下的内存访问效率问题，操作系统深度集成了 THP（透明大页）与 HugePage 技术，优化了跨设备数据交换的页表管理。在混合架构中，不同硬件对内存页大小的支持存在差异，内核通过智能识别与自动配置，为关键计算任务分配大页内存，大幅减少了 TLB（页表缓冲区）缺失率与页表遍历开销。对于国产 CPU 与异构 GPU 之间的数据共享，系统利用统一的虚拟地址空间映射机制，确保大页内存能够在不同设备间无缝流转，避免了因页表碎片化导致的性能抖动。此外，内核还实现了针对异构内存带宽的自适应调度策略，根据实时负载动态调整内存分配粒度，确保高带宽需求的数据流能够充分利用物理内存通道的峰值能力。

在设备管理与兼容性保障方面，OS 内核建立了标准化的设备枚举与属性查询接口，自动识别并注册接入系统的各类异构计算单元。无论底层硬件是 NVIDIA 的 CUDA 架构、AMD 的 ROCm 架构还是国产加速卡，内核均能通过统一的字符设备节点或网络接口暴露其计算能力与显存状态。系统内置的兼容性适配层能够自动协商不同硬件间的通信协议与传输参数，消除因驱动版本不一致或指令集差异导致的运行错误。同时，内核提供了细粒度的设备健康监控与故障隔离机制，当检测到某类异构设备异常时，能够自动将其从资源池中剔除并触发任务迁移，确保集群在部分硬件故障情况下仍能维持整体服务的连续性与稳定性，为大规模异构算力集群的长期稳定运行提供了坚实的软件基础。

持久化存储与高速 I/O 加速

在超节点（SuperPOD）架构中，操作系统通过深度集成 FUSE 框架与高性能并行文件系统（如 3FS、X20000），构建了面向 AI 训练场景的持久化存储底座。内核利用 FUSE 的用户态扩展能力，将 3rd-Storage 等外部存储协议无缝挂载至本地文件系统命名空间，使得应用层能够以标准 POSIX 接口访问分布式存储资源。针对大规模模型检查点（Checkpoint）与海量训练数据集的读写需求，系统采用 3FS 与 X20000 的元数据分离架构，将元数据操作卸载至专用控制节点，大幅提升了小文件并发读写性能。这种架构设计消除了传统单机文件系统在面对 PB 级数据时的元数据瓶颈，确保了在万卡集群环境下，模型参数的快速保存与加载能够线性扩展，满足高频次断点续训的严苛时效要求。

为了进一步突破 I/O 性能极限，操作系统内核直接集成了 DIO（Direct I/O）与 SPDK（Storage Performance Development Kit）等用户态加速技术，构建了绕过内核缓冲区的零拷贝数据路径。这其中，GPU Direct Storage 是为化解数据供给瓶颈而开启的“存储直送专线”。传统流程中，训练数据需从 SSD 经 CPU 内存再拷贝到 GPU 显存。GDS 通过在操作系统内核中引入新的驱动路径，让 GPU 能直接向 NVMe SSD 发起 DMA 请求，将数据块直接搬移到显存，极大地减轻了 CPU 负担，提升了数据加载吞吐量，使 GPU 能持续处于高效计算状态。

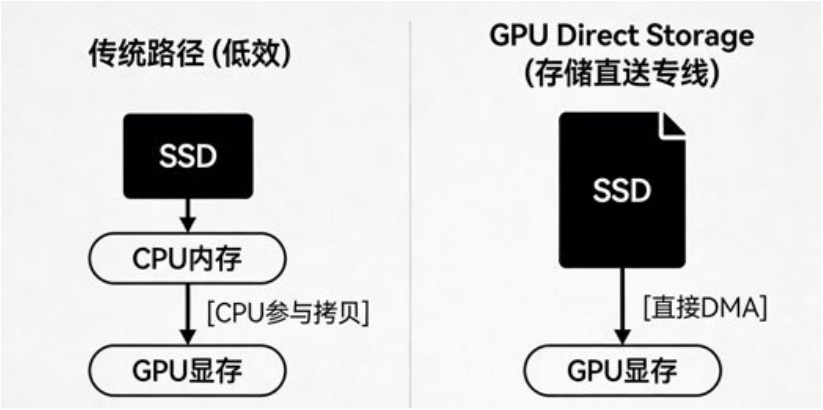


图166 GPU Direct Storage 和传统技术对比图

通过 DIO 机制，应用程序能够直接将数据从用户空间传输至磁盘或网络存储设备，避免了内核页缓存（Page Cache）带来的内存拷贝与一致性维护开销，显著降低了 CPU 占用率。结合 SPDK 的用户态轮询驱动与 NVMe-oF 协议支持，存储子系统能够利用多核并行处理能力，实现微秒级的 I/O 延迟与线速吞吐量。这种全链路加速机制有效解决了 GPU 计算速度远超传统存储读写速度的“木桶效应”，确保在大规模分布式训练中，数据加载与模型持久化操作不再成为制约整体训练效率的瓶颈。

在存储资源调度与可靠性保障方面，OS 实现了智能 I/O 流控与多级缓存协同策略。内核根据训练任务的优先级动态调整 I/O 队列深度与带宽配额，优先保障关键检查点写入任务的实时性，防止背景数据备份任务干扰核心计算流程。同时，系统利用本地 SSD 作为高速缓存层，结合远程并行文件系统的持久化能力，构建了“本地高速缓存 + 远程高可靠存储”的分级存储体系。针对突发的大规模数据读取请求，内核自动触发预读算法与缓存命中优化，减少网络传输延迟。此外，存储子系统内置了数据校验与自动修复机制，能够在硬件故障发生时快速恢复数据完整性，确保超节点集群在长期连续运行中的数据安全与业务连续性。

Runtime 层与 OS 的协同优化

在超节点（SuperPOD）架构中，操作系统内核与 AI Runtime 层（包括 JEMALLOC、GCC、CUDA、PTX 及 NCCL 等组件）建立了深度的协同优化机制，通过暴露底层硬件性能调优接口，打通了从应用逻辑到物理硬件的端到端数据路径。内核提供了优化的 Pin Memory（锁定内存）分配策略，允许 Runtime 直接申请物理连续且不可换页的内存区域，确保 GPU 在进行 DMA 传输时无需进行地址转换或数据拷贝。同时，系统通过 LOCAL/REMOTE MEMORY 标识符，引导 Runtime 智能识别内存拓扑位置，自动将高频访问数据调度至本地 NUMA 节点或显存中，减少跨节点访问延迟。这种紧密协作使得 NCCL 等通信库能够直接利用内核提供的拓扑感知能力，动态选择最优的通信路径，显著提升了大规模分布式训练中的参数同步效率。

针对计算密集型任务，操作系统与编译器栈（如 GCC、CUDA 编译器）及 PTX 中间表示层进行了深度集成，实现了指令级与线程级的上下文切换优化。内核通过细粒度的 CPU 亲和性绑定与中断屏蔽技术，为 Runtime 的关键计算线程提供独占的执行环境，避免操作系统调度器引入的上下文切换抖动。同时，系统支持大页内存（HugePage）与透明大页（THP）的自动适配，结合 JEMALLOC 的内存池化策略，Runtime 能够以更低的开销管理海量临时数据，减少页表遍历带来的 TLB 缺失。此外，内核提供的性能计数器（Performance Counters）接口被 Runtime 深度利用，用于实时监控缓存命中率、指令流水线停顿等指标，进而动态调整 PTX 代码的调度策略与寄存器分配，最大化硬件计算单元的利用率。

在内存布局与资源调度层面，OS 与 Runtime 共同构建了统一的虚拟地址空间管理模型，消除了用户态与内核态之间的地址映射鸿沟。Runtime 层利用操作系统提供的共享内存机制（如 xSHMEM、CXL 共享），实现多进程、多容器间的高效数据交换，而内核则负责维护这些共享区域的权限校验与一致性同步。系统通过智能预取算法与异步 I/O 提交队列，协助 Runtime 提前加载下一轮迭代所需的训练数据，掩盖 I/O 延迟。这种端到端的协同优化不仅降低了系统调用的开销，还确保了在复杂异构环境下，AI 应用能够始终运行在硬件性能的峰值区间，实现了从底层驱动到上层算法的全链路性能提升。

安全与可靠性保障机制

在超节点（SuperPOD）架构中，操作系统利用 IOMMU（输入输出内存管理单元）、PRP（优先级重映射）及 MMIO（内存映射 I/O）等硬件辅助机制，构建了严密的内存保护与访问控制体系。IOMMU 通过硬件级的地址转换与权限校验，强制隔离了不同 GPU、网卡及存储设备之间的直接内存访问路径，防止恶意或故障设备越界访问其他进程的物理内存空间，确保数据隔离的绝对安全。PRP 机制则对关键系统资源的访问请求实

施优先级重映射，保障高优先级的训练任务指令在总线拥塞时仍能获得确定的执行窗口。同时，内核严格管控 MMIO 区域的映射范围，仅允许授权驱动访问特定的寄存器空间，杜绝了非法指令注入导致的硬件状态异常，从底层物理层面确立了系统的可信执行环境。

针对大规模并发场景下的实时监控与故障响应，操作系统深度集成了 eBPF（扩展伯克利数据包过滤器）技术与 CXL Pooling 动态管理机制。eBPF 程序在内核态以沙箱模式运行，能够无侵入地采集网络流量、内存使用率及设备健康度等关键指标，实时构建系统运行的全景视图，并在检测到异常行为（如内存泄漏、死锁或异常流量）时触发自动告警或熔断策略。结合 CXL 内存池化技术，系统实现了对共享内存资源的细粒度监控与动态回收，一旦检测到某个计算节点或内存池出现不可恢复的硬件错误，内核能够迅速将该资源标记为不可用并触发隔离操作，防止故障扩散至整个集群，确保关键 AI 训练任务的连续性与数据完整性。

在系统稳定性保障方面，OS 内核实施了多层级的容错与自愈机制。通过硬件看门狗与软件心跳检测的双重校验，系统能够实时感知节点状态，一旦主节点失效，备份机制将立即接管控制权并重新调度任务。针对高负载下的资源竞争，内核利用动态 QoS 策略与流量整形技术，限制非关键后台任务对核心计算资源的占用，防止因资源耗尽导致的系统雪崩。此外，系统支持配置的热补丁更新与无损重启功能，允许在不中断正在进行的 AI 训练任务的前提下修复内核漏洞或升级驱动版本，从而在保障数据安全与系统高可用的同时，满足超节点集群 7x24 小时不间断运行的严苛要求。

硬件生态适配与支持

在超节点（SuperPOD）架构中，操作系统内核针对海光 DCU、昆仑芯 GPU 及各类国产 CPU 等硬件进行了深度的指令集与微架构适配。内核通过集成专用的设备驱动模块与固件接口，实现了对国产加速器计算单元、显存控制器及片上互联总线的原生支持。系统利用标准化的 PCIe 枚举机制与自定义的寄存器映射表，自动识别并初始化异构国产设备，确保其能够无缝接入现有的内存管理与 I/O 调度框架。针对国产 CPU 特有的多核拓扑与缓存一致性协议，操作系统优化了 NUMA 调度策略与中断处理路径，消除了跨架构通信中的性能损耗，使得国产算力资源能够以接近理论峰值的效率参与大规模分布式训练任务。

为了构建完整的自主软件生态，操作系统提供了从底层驱动到上层工具链的全栈支持能力。内核模块深度集成了国产硬件的特定扩展指令集，支持对海光 DCU 的张量计算单元及昆仑芯的 AI 加速引擎进行直接调用。同时，系统构建了兼容性的设备管理器，统一暴露国产设备的状态监控、功耗管理及故障诊断接口，使得运维工具能够以一致的方式管理混合异构集群。在编译与调试环节，OS 内置的工具链支持针对国产 CPU 架构的交叉编译与性能分析，能够生成高度优化的机器码，并配合国产编译器栈实现从源代码到可执行文件的端到端构建，显著降低了开发者在自主可控环境下的迁移成本与适配复杂度。

在系统完整性与高性能保障方面，操作系统通过内核级的资源隔离与动态负载均衡机制，确保了自主环境下的稳定运行。系统能够根据国产硬件的实际性能特征，自动调整内存分配粒度、线程绑定策略及网络通信参数，最大化利用国产芯片的计算带宽与存储吞吐能力。针对国产硬件可能存在的微架构差异，内核实现了自适应的容错与降级策略，当检测到特定功能模块异常时，能够自动切换至备用执行路径或启用软件模拟模式，防止单点故障导致整个训练任务中断。这种深度适配不仅满足了超节点方案在自主可控环境下的完整性要求，还通过精细化的软硬件协同优化，确保了自主替代场景下的高性能表现与业务连续性。

面向超大规模 AI 集群的韧性操作系统设计

当 GPU 集群规模从数十张扩展至数千上万张，系统可靠性的设计哲学便发生了根本性转向：从追求避免故障转变为主动管理故障。传统架构中，单点故障可导致服务中断，其应对措施集中于本地冗余与事后恢复。而在

千卡级 AI 超算中，硬件故障因规模效应成为常态，因此以 SuperPOD 为代表的新一代操作系统，其核心使命是构建一个能在持续局部故障中保持全局算力持续输出的韧性系统。这一目标通过一个多层协同的自愈体系实现：在硬件层，依托热插拔 GPU 与冗余基础设施，实现故障组件的在线隔离与更换，奠定“不停机修复”的物理基础；在软件层，通过异步增量式检查点技术，将训练任务状态持久化于共享存储，为任意故障提供精准的回溯点；在调度层，融合被动迁移与基于预测的主动疏散，实现计算任务在健康节点间的无缝漂移；在网络层，通过多轨物理冗余与智能路径切换，保障即使出现链路或交换机故障，集群内部的高速通信也不中断、不降级。这套体系共同作用，将故障影响从传统的“整机停机、任务重启、算力空转”优化为“组件级隔离、任务秒级恢复、算力无损”，从而将万卡集群的总体有效算力最大化，真正使可靠性转化为生产力。

对比维度	传统OS可靠性设计	SuperPOD OS / 现代AI集群OS设计
设计哲学	避免故障，追求单机长时稳定运行。	管理故障，假定故障为常态，追求系统在故障中的持续服务。
故障预期	小概率事件（MTBF数万小时）。	必然事件（千卡集群日级故障率显著）。
核心机制	本地硬件冗余（RAID）、定期备份、系统重启。	全局状态感知、热插拔隔离、检查点快照、实时任务迁移。
恢复模式	**停止-修复-重启**：服务中断，从备份恢复，耗时较长。	**在线隔离-迁移-恢复**：服务无感或快速重构，算力中断时间极短。
网络设计	侧重带宽聚合与单链路失效保护。	**多轨物理冗余**与智能路径切换，防止集群分裂，保障集体通信确定性。
运维范式	故障发生后被动响应，人工介入处理。	系统主动预测、自动隔离与迁移，实现预防性维护。
规模适应性	适用于数十至数百节点，故障影响局部。	为千卡至万卡集群设计，确保单点故障不影响全局任务进度与算力输出。

图167 普通 OS 与超节点 OS 对比

集群节点 Fabric 管理

超节点架构可采用 Scale-up 技术互连计算节点、加速器与内存扩展设备连接为统一的 Scale-up Fabric，从而支持资源解耦、容量扩展与共享访问。释放硬件潜能的关键在于软件控制平面——Fabric Manage(FM)。

FM 作为管理实体，可采用集中式或分层/分布式部署，并可通过节点侧代理/本地管理组件配合完成发现、监控与执行。FM 负责对底层 Scale-up Fabric 进行初始化、配置、监控与变更控制，核心功能通常包括：

Fabric 发现、枚举与拓扑/状态建模：在系统启动或结构变更时，FM 通过管理通道对 Fabric 内的交换设备、端口与连接关系进行发现与枚举，采集组件能力与运行状态，形成可用于运维与自动化的 Fabric 视图（包括设备清单、端口状态、链路属性、可用能力等）。当拓扑或组件状态发生变化时，FM 负责更新视图并触发相应的配置校验与后续编排流程。

资源发现与资源编排（绑定、分配、映射）：FM 负责发现并管理 Fabric 内可被编排的资源，例如内存扩展（以及相关的访问能力与约束。FM 根据策略完成资源的分配与编排，包括但不限于：将目标资源绑定到指定主机/节点或主机集合;配置与下发与资源访问相关的参数（例如访问窗口、解码/映射、隔离域等）;在资源变更（扩容、缩容、迁移、回收）时进行一致性控制，确保配置生效过程可控、可回滚、可审计。

安全、隔离与访问控制：FM 负责在 Fabric 范围内实施安全与隔离策略，确保资源共享场景下的边界清晰与权限最小化，包括：多主机/多租户场景的访问控制与隔离域管理；设备接入与能力校验（合规性检查）；与安全相关配置的统一下发、变更控制与审计。

可靠性、可用性与可服务性与事件处置编排：FM 负责汇聚并处理 Fabric 内的事件与错误信息（包括链路/端口状态变化、设备告警、错误计数与健康状态等），并基于策略执行故障处置编排，例如：故障定位与影响域评估（哪些主机/资源/连接受影响）；隔离/降级/资源替换（将受影响资源下线、切换到冗余资源或重新分配）；触发恢复流程并维护一致性状态（确保恢复过程中资源绑定与映射关系正确）。

变更管理与一致性下发：针对设备上线/下线、资源增减、权限调整等操作，FM 负责统一的变更编排与一致性下发，保证变更过程可控（顺序、依赖、校验）、对业务影响可评估，并支持回滚与审计。

面向 PD 分离架构的跨节点通信加速

在面向预填充与解码分离架构的跨节点通信加速领域，操作系统内核深度整合了 GPU Direct RDMA 与 GPU Direct Storage 两大核心技术。通过将 P 节点生成的中间激活值直接映射至 D 节点的显存空间，系统彻底绕过了传统 CPU 内存拷贝路径，实现了数据在 GPU 间零拷贝、低延迟的传输。这种设计不仅显著压缩了 Token 生成过程中因数据搬运产生的网络开销，更将通信延迟从毫秒级降低至微秒级，为大规模并发推理场景下的实时响应能力提供了坚实基础。

为进一步突破跨机柜通信的性能瓶颈，系统构建了基于 Scale-up 交换技术的虚拟统一内存空间。借助 Scale-up 技术提供的远程内存访问能力，D 节点可直接访问 P 节点的计算结果，无需进行序列化、反序列化或数据复制操作，从根本上消除了传统通信模型中的“软件开销墙”。该架构将原本分散在不同物理节点上的显存资源抽象为一个逻辑连续的内存池，配合高速互联协议如 InfiniBand 与 RDMA NIC，实现了跨机柜级的内存共享与数据同步，极大提升了模型推理的整体吞吐量与资源利用率。

针对 PD 分离架构下负载动态波动的特性，操作系统网络栈引入了自适应路由调度与拥塞控制机制，对 P-D 节点间的通信流量实施精细化管理。系统优先保障控制面（如请求调度、状态同步）与关键数据面（如激活值流）的 QoS 等级，通过动态调整带宽分配策略，在高负载环境下仍能维持通信通道的稳定性和低延迟特性。结合 NUMA-aware 调度、内存共享与 GPU P2P DMA 等底层能力，OS 确保跨节点通信带宽能够随集群规模线性扩展，从而满足 AI 推理服务对低延迟、高吞吐 SLA 的核心诉求。

超节点 LLM 推理演进中操作系统的关键角色与优化实践

在大语言模型推理日益成为计算核心负载的今天，Scale-up 超节点架构通过集成数十张高性能 GPU 和极速内部互联，为处理这类任务提供了强大的硬件基础。在这一架构中，操作系统扮演着底层资源协调与保障的关键角色，而像 NVIDIA NIXL 这样的专用库则构成了智能调度与执行的中枢神经系统。

NIXL 库的核心使命是优化生产环境中的 AI 推理吞吐与延迟。它并非简单的通信库，而是一个高级的推理服务编排引擎。在超节点环境中，NIXL 动态管理着海量的并发请求，执行智能的批处理操作以填满 GPU 计算单元，并将请求精准路由到部署在特定 GPU 上的对应模型实例。操作系统则为这一切提供了稳固的基石，它通过容器化技术实现多租户 GPU 资源的强隔离与安全共享，通过内核驱动抽象硬件细节，并保障整个服务栈的高可用性。

当前沿技术向大语言模型推理深度演进时，超节点内部的协同优化出现了新的范式。业界领先的解决方案已经开始提供类似 NVIDIA NIXL 库中 NIXL_Transfer 这样的硬件级优化接口。该接口旨在直接解决 LLM 推理中 Prefill（预填充）阶段与 Decode（解码）阶段之间的关键性能瓶颈，即庞大的 KV Cache 数据传输问题。传统方式需要经过主机内存中转，而 NIXL_Transfer 支持 GPU 之间的直接异步传输，彻底规避了昂贵的主机内存拷贝开销，有望将端到端推理延迟显著降低。

这一硬件协同优化趋势对操作系统提出了更深层次的要求。首先，操作系统必须与硬件驱动进行更深度协同，其内核需要将 GPU 之间的高速物理互联能力，安全且可靠地抽象为一种标准化的服务，并向上层库开放可控的访问接口。其次，系统的资源调度器，从操作系统本身的进程调度到 Kubernetes 的 Pod 调度器，都需要进化出对 GPU 互联拓扑的感知能力。这意味着调度决策不仅要考虑“是否有 GPU”，更要理解“哪些 GPU 之间拥有最高的直接带宽”，从而实现计算任务与硬件拓扑结构的最优匹配。最后，当 GPU 间直接数据通道成为常态，操作系统必须在硬件层面强化对这一新路径的安全隔离，例如依托 IOMMU 等技术，确保在云原生多租户环境下，不同用户间的数据绝对隔离，这是 Scale-up 架构走向规模化商用的必要前提。

综合来看，以 NIXL_Transfer 为代表的技术是突破 LLM 推理性能瓶颈的关键路径，它标志着优化重点从单一芯片计算效率转向了芯片间协同效率。这一方向必将引发所有 AI 硬件厂商的广泛跟进，从而推动从硬件互联、设备驱动、操作系统内核到中间件和调度器的全栈协同标准化。未来一至两年，这将是 Scale-up 推理软件栈最核心的优化方向，最终目标是将超节点从一台强大的硬件设备，彻底转变为一个高效、透明且易管理的巨型推理算力单元。

Scale-up 网络技术的应用价值闭环

Scale-up 架构中 IODIE 与高速交换网络的开发，其根本价值在于通过重塑硬件通信能力基本面，构建了一条从底层物理优化直达大模型应用性能提升的完整技术链条。这一链条始于超节点内部提供的高带宽、低延迟与硬件原子操作能力，为海量的 Read、Write、Atomic 操作设立了全新的性能上限。中间的软件栈，特别是通信库与调度器，通过精密的算法将原始操作封装为高级语义接口，并借助流控、保序与负载均衡机制，确保数据流能稳定、高效地逼近硬件极限。最终，这一整套协同优化使得大模型训练中的梯度同步时间急剧缩短，GPU 计算利用率得以持续饱和；在推理侧，则让 KV Cache 等关键数据的传输延迟被有效隐藏。其核心因果逻辑在于，将网络从传统分布式计算中的性能瓶颈，转化为让数十张 GPU 凝聚为单一强大算力容器的“神经束”，从而直接将底层的每一次比特移动，无损地转化为上层模型迭代速度的飞跃与推理服务质量的质变。

6 资源池化

AI 工作负载下的资源碎片化与池化诉求

AI 工作负载的极端动态性和异质性导致传统紧耦合服务器架构下资源碎片化问题日益严重：在万卡级训练中，模型参数、激活值、梯度、Optimizer 状态和 KV Cache 的显存占用极不均匀，MoE 模型的专家并行进一步放大碎片，导致整体 GPU 利用率不高；推理场景中，长上下文 RAG 任务的 KV Cache 可瞬间膨胀数百 GB，而后续生成阶段又迅速收缩，单节点 HBM 无法弹性应对，造成大量显存浪费或频繁 OOM。在混合负载 AI Factory 中，训练、微调、推理、embedding 生成等多租户并发，资源需求在时间和空间维度上剧烈波动，传统静态分区方式（MIG/MPS）虽能隔离但无法实现细粒度动态复用，碎片化最终转化为算力浪费和 TCO 暴增。

资源池化核心概念

资源池化本质上是 Disaggregated Infrastructure 在 AI 领域的落地实践，即将计算（GPU/CPU）、内存（HBM/DRAM/CXL）、存储（NVMe-oF）和网络（RDMA Fabric）从物理紧耦合转向逻辑解耦，形成可独立扩展、按需组合的资源池。计算池以 GPU/TPU 为核心，支持 MIG/SR-IOV 切分后按需分配；内存池

通过 CXL 或 PCIe Switch 实现远程 DRAM/HBM 共享，允许单个 GPU 访问远超本地容量的内存空间；存储池借助 NVMe-oF + GPUDirect Storage 实现从远程 SSD/NVMe 直达 GPU 的零拷贝路径；网络池则通过高性能 Fabric（如 InfiniBand 800G 或 RoCE v2）确保跨池访问延迟可控。真正有效的池化需满足三个关键特性：一是语义统一（统一 Load/Store 访问模型），二是低延迟（<200ns 远程访问），三是强隔离（硬件级加密+虚拟化）。解耦后，AI Factory 可实现真正的“计算即服务”：一个训练作业可动态借用空闲节点的 HBM 作为扩展 KV Cache，推理服务可瞬时扩展内存支持 1M+上下文，而无需重启或迁移整个容器。

CXL 2.0 在内存池化中的关键作用

CXL 2.0 与即将到来的 CXL 3.0 是内存池化的核心使能技术。CXL 2.0 基于 PCIe Gen6 物理层，提供 256 GB/s x16 双向带宽，支持 Type-2 设备（带缓存一致性的加速器，如 SmartNIC 或自定义内存控制器）和 Type-3 设备（纯内存扩展器），Switch-Based Pooling 允许数十至数百个主机共享同一内存池，通过 CXL switch fabric 实现多对多连接。Weighted Interleaving 是 CXL 3.0 的关键创新，根据权重动态分配内存访问（例如将 80%访问分配给低延迟本地 DRAM，20%到远程 CXL 内存），极大缓解了 NUMA 效应。CXL 4.0（预计 2028 - 2030）进一步提升至 512 GB/s+带宽，并引入更强的 TSP 安全协议和光互联准备。

OS 内核与软件栈支持

Linux 内核从 6.9 开始对 CXL 支持进入生产级成熟阶段，cxl driver 全面支持 CXL 2.0 的 Type-2/3 设备枚举、端口管理和内存映射；weighted interleaving 通过/sys/devices/cxl/接口配置权重，实现异构内存的非均匀访问策略；flat memory mode 将 CXL 内存暴露为统一地址空间，配合 HMM（Heterogeneous Memory Management）实现 GPU 直接 Load/Store 远程页。virtio-mem 提供动态热插拔能力，允许运行时添加/移除内存段，结合 madvise(MADV_COLD)实现内存 tiering（DRAM→CXL→SSD）。

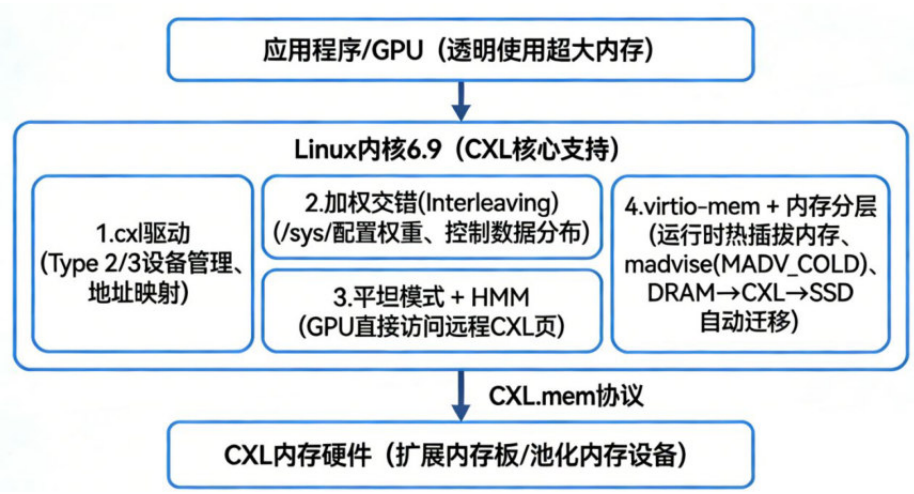


图168 内核 6.9 支持 CXL 图

编排层实现

控制平面集成 Kubernetes Dynamic Resource Allocation（DRA）从 1.26 开始支持 CXL 内存等可扩展资源，通过 ResourceClaim 模板定义“cxl-memory-gb”资源类型，Custom Scheduler（Volcano/Kueue

扩展）根据拓扑、负载和内存需求实时分配。Liqid 和 GigaIO 的控制平面通过 REST API 与 Kubernetes 集成，实现运行时资源组装：Pod 启动时动态申请远程 DRAM，释放时归还池中。

AI 特有收益

AI 特有收益是资源池化的最大价值所在：KV Cache 池化允许将长序列生成的键值对从本地 HBM 溢出到 CXL 共享内存，按需加载回 GPU，在 1M 上下文 RAG 任务中，池化后单节点可支持原先更高的并发 QPS；RAG 长上下文的内存膨胀/收缩通过 virtio-mem + weighted interleaving 动态调整，膨胀时热插拔远程 DRAM，收缩时冷页迁移到 SSD，内存利用率提升。训练/推理混合负载下，池化实现动态分配：训练高峰期借用空闲推理节点的 HBM，推理突发时反向借用训练节点的 DRAM，结合 Checkpoint/Restart 的无缝迁移，整个集群在峰谷负载波动中保持较高利用率。

7 互联语义演进：从 Read/Write 到统一 Load/Store 内存访问

未来 AI 工作负载下的统一内存访问需求画像

未来 AI 工作负载，尤其是 2026–2030 年间的万亿甚至十万亿参数模型，将对内存访问模型提出前所未有的统一性要求：训练阶段需要无缝访问海量参数、激活、梯度、Optimizer 状态和 KV Cache；推理阶段要求毫秒级响应长上下文生成，同时支持动态 KV Cache 膨胀/收缩和 RAG 检索；混合负载下，训练、微调、推理、embedding 生成等多任务并发，内存需求在时间和空间上剧烈波动，单次生成可能瞬间需要数百 GB 额外内存，而下一迭代又迅速释放。传统 DMA Read/Write 模型下，这些需求导致频繁的显式拷贝、Pinned Memory 分配爆炸、页面故障抖动和编程复杂度指数级上升。在前瞻研究和多个前沿 AI Factory 项目中，统一 Load/Store 语义已成为必然趋势：GPU/CPU/加速器像访问本地 DRAM 一样直接 Load/Store 远程 CXL 内存或 Fabric 共享内存，编程模型接近单机统一内存（UMA），无需 cudaMemcpy、Pinned 分配或显式同步。这种需求画像的核心痛点在于：当前异构系统下，内存仍是“私有孤岛”，而未来 AI 将要求“全局内存湖”，任何设备都能以近似本地速度访问全集群内存池，实现真正的“内存即服务”。

传统 DMA Read/Write 模型的局限

传统 DMA Read/Write 模型是当前异构计算的主流路径，但其局限性在超大规模 AI 中已暴露无遗。首先，Pinned Memory（cudaHostRegister / cudaMallocHost）虽能加速主机→设备拷贝，但会锁定主机物理页，导致系统 DRAM 碎片化严重，在高并发推理中 Pinned 分配往往耗尽主机内存，触发 OOM 或 swap 抖动。其次，复制开销巨大：训练中每迭代需多次主机↔设备、GPU↔GPU 拷贝，AllReduce 前梯度需显式 Pinned 并拷贝，KV Cache 管理需反复搬运长序列数据。第三，编程复杂度爆炸：开发者需手动管理 Pinned 分配、流同步、异步拷贝、重叠计算、错误处理和内存释放，代码中充斥 cudaMemcpyAsync、cudaStreamSynchronize 和 cudaHostUnregister，极易引入 race condition 或内存泄漏。在多租户 AI Factory 中，这种显式模型还导致隔离困难，租户间 Pinned 内存竞争引发性能噪声。

统一 Load/Store 语义的体系图

统一 Load/Store 语义的核心是构建一个全局地址空间，让 CPU、GPU、DPU、CXL 内存扩展器和 Fabric 共享内存像访问本地一样直接使用 load/store 指令，而无需显式 DMA。体系图从底层到上层可分为四层协同：

硬件互联层：CXL.mem 提供远程内存的 Load/Store 语义，NVLink-C2C 实现 CPU-GPU 间统一地址空间，InfiniBand/RoCE Fabric 通过 GPUDirect RDMA 扩展到跨节点。

- 内存管理层：HMM（Heterogeneous Memory Management）+页表共享让 GPU 页表直接映射 CXL/Fabric 内存，远程页故障时触发页面迁移或就地访问。
- OS 内核层：Linux 内存子系统扩展为 tiered/flat 模式，CXL 驱动注册远程内存为系统 NUMA 节点，virtio-mem 支持动态热插拔。
- 运行时/框架层：CUDA/HIP/oneAPI/XLA 通过统一虚拟地址（UVA）暴露给用户代码，PyTorch/TensorFlow/JAX 原生支持 cudaMallocManaged 或 torch.cuda.UnifiedMemory。 在这一体系中，CPU 发起的 load/store 可透明穿越 CXL switch 到远程 DRAM，GPU 的 tensor 操作可直接访问 Fabric 上的共享 KV Cache，整个路径延迟从传统 DMA 的微秒级降至纳秒级，编程模型接近单机 UMA。

硬件与互联层使能

硬件与互联层是统一 Load/Store 语义的物理基础。未来 CXL.mem（CXL 3.0+）允许 Type-3 设备作为系统内存扩展，CPU/GPU 通过 load/store 直接访问远程 DRAM，支持 switch pooling 和 weighted interleaving，实现多主机共享。NVLink-C2C（Grace-Hopper/GB200/Rubin）在 CPU-GPU 间构建全硬件一致性 UMA 域，带宽达 1.8 TB/s+，支持原子操作和缓存一致性，GPU 可直接 load/store CPU DRAM，反之亦然。HMM 扩展页表共享，让 GPU 页表映射 CXL/Fabric 内存，页面故障时内核触发迁移或就地访问。UMA（Unified Memory Architecture）架构在硬件层面实现全局地址空间，Blackwell/Rubin 的 Tensor Memory Accelerator 进一步加速统一内存预取。

OS 内核与内存管理演进

Linux 内核从 6.9 开始加速向统一内存演进，页表共享通过 HMM 的 device-private pages 和 multi-page table 支持 GPU 直接映射远程内存；远程内存注册为 NUMA 节点，cxl driver 提供/dev/cxl/接口，允许动态添加/移除 CXL 内存段。内存分级(tiering)结合 damon、madvise(MADV_COLD/HOT)和 autonuma，实现 DRAM（热层）→CXL（温层）→SSD（冷层）的自动迁移，weighted interleaving 根据负载权重分配访问比例。virtio-mem 和 hotplug 内存支持运行时扩展，结合 numactl --membind 和 kernel.numa_balancing=0（生产环境常关闭自动平衡以防抖动）精细控制策略。

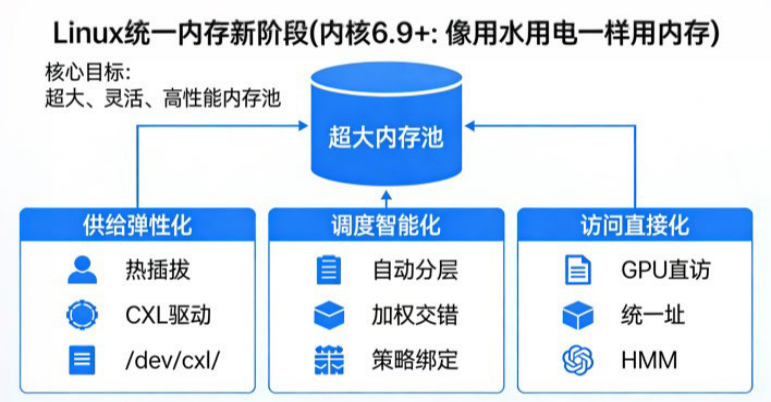


图169 Linux 内核 6.9 内存管理图

AI 框架与运行时适配

AI 框架对统一内存的适配是落地统一语义的关键一环。PyTorch 通过 torch.cuda.UnifiedMemory 和 torch.utils.checkpoint 实现托管内存分配，torch.compile + inductor 后端自动识别 load/store 模式，减少显式拷贝；TensorFlow 的 UnifiedMemory API 和 tf.experimental.dlpack 支持直接共享 CXL 内存；JAX 的 jax.Array 在 pmap/shmap 下原生利用 HMM 映射远程页；XLA 编译器通过 Buffer Donation 和 Unified Buffer 优化统一内存访问路径，避免不必要的分配/释放。

8 总结与展望

操作系统在超节点时代已从传统的资源管理器，彻底演变为 AI 基础设施的“神经中枢”与“互联大脑”。在单节点多 GPU 的 Scale-up 场景中，OS 负责拓扑发现、NUMA/CXL 亲和调度、Pinned Memory 优化、GPUDirect 路径管理、功耗热控与故障域隔离，将数十至数百张 GPU 真正融合为逻辑统一的“巨型加速器”；在跨节点万卡级的 Scale-out 场景中，OS 通过 RDMA 子系统、NCCL/SHARP 集成、GPUDirect Storage、CXL 内存池化驱动和 topology-aware 调度，支撑 AllReduce、KV Cache 共享与动态资源借用，真正让“cluster = giant GPU”从概念落地为生产级现实。

当前，OS 面临的挑战依然严峻：NUMA/CXL 异构内存访问的延迟非均匀性、跨节点通信的拥塞风暴、多租户场景下的噪声与碎片化、异构 xPU 驱动的兼容性与稳定性，以及海量小文件 I/O 与长序列 KV Cache 的内存膨胀压力，都在考验内核内存子系统、调度器、RDMA 栈与容器运行时的极限协同能力。自主化进程进一步放大了这些挑战的紧迫性：从飞腾/海光/鲲鹏 CPU 到昆仑芯/沐曦/天数/寒武纪 xPU 等等，再到国产高性能网卡与 CXL 内存扩展器，全栈自主可控要求 OS 在驱动适配、通信库优化、拓扑感知与安全隔离上实现端到端国产替代。

展望未来 5 - 10 年，随着 PCIe 7.0/8.0、CXL 3.0/4.0、NVLink 5.0/6.0、光互联与片上光电混合计算的商用，操作系统将迎来全面重构：统一内存语义（Unified Load/Store）将成为默认编程模型，HMM 与 CXL.mem 深度融合实现全局地址空间；内存分级与自动 tiering 演进为 AI 感知的智能分层（AI-driven tiering），根据模型层级、注意力分布与访问热图动态迁移 KV Cache 与激活值；内核调度器将从 NUMA-aware 升级为 fabric-aware 与 memory-semantic-aware，支持跨节点、跨内存池的原生迁移与借用；容器与虚拟化层通过 DRA、virtio-mem 与 runc/youki 的深度改造，实现亚秒级资源组装与热插拔；安全层面，硬件级 TSP（Trusted Security Protocol）与 eBPF 驱动的零信任隔离，将成为 AI Factory 多租户的标配。

最终，操作系统将从“基础设施”升华为“AI 原生操作系统”，与硬件、通信库、框架和调度器共同构成一个自适应、自愈、自我优化的算力中枢。国产 AI Factory 的成功，不再仅仅取决于芯片算力密度，而将取决于谁能率先在 OS 层实现全栈自主可控的互联统一、内存统一与调度统一。操作系统，正迎来它在 AI 时代最激动人心的重生。

9 参考文献

[1] https://arxiv.org/abs/2306.03745

[2] https://en.wikipedia.org/wiki/Intel_Ultra_Path_Interconnect

[3] https://developer.nvidia.com/blog/nvidia-grace-hopper-superchip-architecture-in-depth

[4] https://www.nvidia.com/en-us/data-center/h200/

[5] https://www.amd.com/en/products/accelerators/instinct/mi300.html

[6] https://www.nvidia.com/en-us/data-center/dgx-h200/

[7] https://arxiv.org/abs/2306.03745

超节点技术选型与部署实践

1 选型方法论

超节点（Super Pod）作为支撑大规模人工智能模型训练与推理的核心基础设施单元，其技术选型直接关系到系统性能、成本效率、长期演进能力及合规安全性。面对异构硬件生态快速演进、模型架构持续迭代、业务场景日益多元的复杂环境，传统的“性能优先”或“成本导向”单一维度评估已难以满足实际需求。选型方法论提出一套系统化、可量化、可复用的超节点技术选型框架，旨在通过“需求驱动—多维评估—动态决策”的闭环流程，实现科学、稳健且面向未来的选型决策。

1.1 技术选型的意义

超节点作为 AI 时代智算基础设施的核心计算单元，其技术选型并非简单的硬件或软件搭配，其核心意义主要体现在五个方面。

一，支撑超大规模 AI 任务高效落地。当前大模型参数已从千亿迈向万亿级别，MoE 等先进架构广泛应用，对算力规模、通信带宽、延迟控制提出极高要求，合理的技术选型能实现计算、存储、网络资源的深度融合，破解传统集群架构的扩展性瓶颈，确保超节点可高效支撑超大规模模型训练与推理任务，契合 Scale Up 架构的性能提升需求，如英伟达 Blackwell 超节点 72 卡配置，基于任务需求实现性能与复杂度的均衡适配。

二，保障业务性能与体验的稳定性。超节点的核心价值的是通过一体化设计实现高性能、低延迟、高可靠的计算能力，技术选型直接决定算力利用率、延迟控制精度、系统容错能力等关键指标，能否匹配业务的延迟敏感需求、算力弹性需求，直接影响下游业务的运行稳定性，例如实时交互类场景的毫秒级延迟诉求、离线训练场景的高吞吐量需求，均需通过精准选型实现落地。

三，优化全生命周期总拥有成本（TCO）。超节点部署涉及初始硬件投入、软件授权、部署实施、运营维护、迭代升级等多环节成本，科学的技术选型可实现性能与成本的平衡，避免盲目追求高端配置导致的资源浪费，同时降低后续运维难度与升级成本。

四，规避生态兼容与合规风险。超节点技术选型需兼顾现有技术生态的适配性与行业合规要求，合理的选型可减少与现有系统、第三方工具的对接成本，避免出现生态碎片化或厂商锁定问题，同时满足数据安全、隐私保护、行业监管等合规要求，为超节点长期稳定运行筑牢基础。

五，构建长期技术竞争力。超节点技术迭代速度快，硬件芯片、互联协议、软件框架持续升级，精准的技术选型能兼顾当前需求与未来扩展潜力，确保超节点系统可适配技术迭代与业务增长需求，避免选型滞后导致的系

统淘汰或大规模重构，同时通过自主可控相关技术的选型布局，提升供应链安全性，构建差异化技术优势，为业务长期发展提供算力支撑。

1.2 方法论整体框架

框架由两大核心模块构成：需求分析 与 业务指标，二者相互支撑、逐层递进。

需求分析 聚焦业务与技术场景的本质诉求，从五个关键维度对超节点的能力边界进行精准刻画：

- 模型规模：明确所支持模型的参数量级、结构类型（如 Transformer、MoE、扩散模型等）及其对计算图和内存布局的特殊要求；
- 算力需求：量化训练/推理任务在峰值算力、内存容量、带宽及通信效率等方面的硬性指标；
- 延迟敏感程度：区分实时交互、近实时批处理与离线训练等场景，定义可接受的延迟阈值与服务质量（QoS）等级；

业务指标 在明确需求基础上，围绕五大高阶维度对候选技术方案进行综合评估：

- 性能要求：不仅关注理论峰值性能，更强调实际工作负载下的有效吞吐、扩展效率与能效表现；

通过该框架，组织可在复杂技术选项中快速识别最优解，降低试错成本，提升基础设施投资回报率，并为 AI 系统的可持续演进奠定坚实基础。

1.3 模型运行基础环境

在超节点技术选型过程中，模型规模是决定硬件资源配置与系统架构设计的核心前置条件。模型的参数量级与架构复杂度直接决定了计算、内存、通信等资源的需求基线，而训练与推理两个阶段在资源使用模式上存在显著差异，必须分别建模、精准评估。本节将从参数量级与架构特征出发，系统分析不同阶段的资源需求特点，为后续算力匹配与系统设计提供依据。

1. 参数量级与架构复杂度集群管理平台

模型的参数量级是衡量其规模最直观的指标，通常以“百万（M）”、“十亿（B）”或“万亿（T）”为单位。当前主流大模型参数量普遍处于 7B 至 671B 区间，部分领先模型已突破万亿级别。参数量越大，对显存容量、带宽和计算吞吐的要求呈非线性增长。

在评估参数量级时，需结合模型架构类型进行综合判断，常见架构包括：

- 稠密 Transformer 架构：如 LLaMA、GPT 系列，所有参数在每次前向传播中均被激活，训练阶段显存占用约为参数量的 16 - 20 字节/参数（含优化器状态与梯度）。
- 稀疏专家模型（MoE, Mixture of Experts）：如 Mixtral、GLaM，仅激活部分专家网络，理论参数量虽高，但实际激活量可控。评估时需关注“激活参数比例”与“路由机制开销”，避免被“纸面参数”误导资源规划。
- 多模态架构：如 CLIP、Flamingo，涉及视觉与语言模态对齐，需额外评估图像编码器与跨模态注意力模块的计算与内存开销。

- 其他特殊架构：如图神经网络（GNN）、状态空间模型（SSM），其计算模式与 Transformer 差异显著，需针对性评估硬件支持度。

此外，还需关注以下架构复杂度因素：

- 上下文长度：支持 32K、128K、256K 甚至更长百万级别的上下文模型，KV Cache 内存占用急剧上升，对显存容量与带宽提出更高要求；
- 注意力机制变体：如滑动窗口、稀疏注意力、全局+局部混合注意力，影响内存访问模式与并行效率；
- 量化与稀疏化支持：模型是否支持 INT8/INT4 量化或结构化稀疏，直接影响硬件部署效率与吞吐表现。

评估建议：在选型初期，应获取目标模型的完整架构文档或配置文件（如 config.json），明确参数总量、层数、头数、隐藏维度、专家数量等关键信息，并结合架构类型预估实际资源消耗，避免仅依赖“参数量”单一指标进行粗略估算。

2. 训练/推理显存评估

推算模型在训练或推理过程中所需的显存（GPU Memory），是模型部署和优化中的关键步骤。显存消耗主要来自模型参数、梯度、优化器状态、激活值（activations）以及临时缓冲区等。其中，模型参数本身的存储是最基础的部分，其大小直接与参数数量和所使用的数值精度（如 FP32、FP16）相关。

- **基本单位换算**
  - FP32：每个参数占 4 字节（bytes）
  - FP16：每个参数占 2 字节（bytes）
  - BF16：占 2 字节
  - FP8：1 字节，是现在大模型高效训练、长上下文推理的主流选择
  - INT8：1 字节（用于量化推理）
- **显存构成**

  模型显存占用分为三部分（推理 / 训练场景差异大）：
  - 模型核心：模型参数（权重）本身的显存
  - 训练专属：梯度（与参数数量 / 精度一致）、优化器状态（如 Adam 需额外存储 m/v 值）
  - 通用：临时计算张量（中间结果，与 batch size、序列长度等相关）
- **显存推算方法**
  - 仅推理场景（仅模型权重）

    计算参数本身的显存：FP32 显存(GB) = (参数数量 × 4) / 1073741824

    示例：70 亿（7B）参数的模型 FP32 推理显存 = $(7\times10^9\times4)$ / 1073741824 ≈ 26.02 GB
  - 训练场景（含模型权重+梯度 + 优化器）

    训练时总显存≈ 基础显存 + 临时张量显存

    基础显存=权重显存 + 梯度显存（同权重）+ Adam 优化器显存（8 字节/参数）

临时张量=经验值为基础显存的 1–3 倍（与模型结构/batch size 相关）

FP32 训练基础显存(GB)=(参数数量 × (4+4+8)) / 1073741824 =(参数数量 ×16) / 1073741824

FP32 训练总显存 ≈ 基础显存 + 临时张量显存

示例：70 亿参数 FP32 训练基础显存 =(7×10⁹ ×16)/1073741824 ≈ 104.08 GB，加临时张量后需 100–300GB 显存。

- **FP32 与 FP16、INT8 显存计算差异**

FP32 与 FP16 的显存计算差异，本质是单参数字节数的 2:1 差异，在无额外显存开销的推理场景呈现严格的 50% 显存占比；在有梯度、优化器开销的训练场景，因 Adam 优化器默认采用 FP32 存储，FP16 混合精度的显存计算差异被稀释，最终比纯 FP32 节省 25% 左右的基础显存，这也是工业界选择 FP16 混合精度训练的核心原因 ——以极小的精度补偿（如损失缩放），实现显著的显存节省。

FP8 单参数仅占 1 字节，与 FP32 形成 4:1、与 FP16 形成 2:1 的字节数差异，推理场景下显存可进一步压至 FP16 的 50%、FP32 的 25%，且仍为浮点格式，精度稳定性远优于 INT8 量化。

组件	FP32	FP16（混合精度训练）	INT8
模型参数	4N	2N（基于优化策略差异，混合精度可能需再维护一个FP32主副本）	1N
梯度（Gradients）	4N	2N（通常与参数同精度）	1N
优化器状态（如 Adam）	8N（m + v 各 4N）	若使用 FP32 主权重，则仍需 8N	若使用 FP32 主权重，则仍需 8N
激活值（Activations）	与 batch size、序列长度、层数相关，通常远大于参数	可减半（若用 FP16 计算）	可减四分之三

3. AI 模型训练时间与推理时间

- **训练与推理的核心运算差异**

计算量（FLOPs，浮点运算次数）是模型处理数据需完成的总运算量，为模型固有属性；

有效算力（FLOPS，浮点运算次数/秒）= GPU 理论算力 × 算力利用率（η，0~1），需结合实际硬件负载估算（避免仅用理论算力导致误差过大）。模型训练与推理的耗时计算逻辑完全通用，核心差异仅体现在计算量的构成上：

  - 模型训练：需完整执行前向传播 + 反向传播 + 参数更新全流程，运算环节多、数据迭代次数多，对应总计算量更大。
  - 模型推理：仅需执行前向传播单一流程，无需参数迭代与反向计算，运算环节大幅简化，总计算量显著降低。

- **基础核心概念**

  单样本前向计算量（F_forward）：模型对单个样本（图像、文本 token 等）进行一次前向传播的浮点运算总数（FLOPs），可通过 thop、torchprofile 等工具快速测算，无需手动推导。

  硬件理论算力：GPU/TPU 等智算硬件的标称峰值算力（如 A100 FP16 理论算力 312 TFLOPS），需匹配模型运行精度（FP32/FP16/INT8）选取对应算力值。

  算力利用率（η）：硬件实际发挥的算力占理论算力的比例，受显存负载、数据加载、并行策略影响，训练场景通常为 0.3~0.7，推理场景为 0.6~0.9（专用框架优化后可更高）。

  Batch Size：单次运算处理的样本数，不影响总计算量，仅影响单步耗时和硬件利用率（合理增大可提升利用率）。

- **推理时间计算**

  - 计算逻辑

    推理仅需完成前向传播（无需反向传播和参数更新），单样本推理计算量=F_forward，场景分为批量推理（离线处理）和实时推理（在线服务），公式可通用，仅关注指标不同。

  - 核心公式（可直接套用）

    核心公式：推理总耗时 =（总推理样本数 × F_forward）÷（硬件理论算力 × η）

  - 实操示例

    已知：F_forward=1 GFLOPs，总推理样本数 100 万，A100 FP16 理论算力 312 TFLOPS，η=0.7（推理利用率）

    计算过程：总计算量=$10^9 \times 10^6 = 1 \times 10^{15}$ FLOPs；有效算力=$312 \times 10^{12} \times 0.7 \approx 218.4 \times 10^{12}$ FLOPS

    推理总耗时=$1 \times 10^{15} \div 218.4 \times 10^{12} \approx 4.6$ 秒；单样本延迟≈$4.6 \div 10^9 \approx 0.0046$ ms

- **训练时间计算**

  - 计算逻辑

    训练的核心运算为“前向传播（计算预测结果）+ 反向传播（修正模型参数）”，其中反向传播计算量约为前向传播的 2 倍（行业通用近似），因此单样本训练计算量≈3×F_forward；参数更新计算量极小，可忽略不计。

  - 核心公式

    核心公式：训练总耗时 =（3× F_forward × 总训练样本数 × 训练轮数 ）÷（GPU 理论算力 × η）

    补充说明：

    总训练样本数（N_train）：训练数据集的样本总量；

    训练轮数（E）：训练集被完整遍历的次数；

    3×F_forward：单样本前向+反向的总计算量（简化估算，精准场景可通过工具实测）；

    简化推导：总计算量=3×F_forward×N_train×E，有效算力=GPU 理论算力×η，总耗时=总计算量÷有效算力。

- 实操示例

  已知：F_forward=1 GFLOPs（单样本前向计算量），总训练样本数 100 万，训练轮数 10，A100 FP16 理论算力 312 TFLOPS，η=0.5（训练利用率）。

  计算过程：总计算量=$3\times10^9\times10^6\times10=3\times10^{16}$ FLOPs；有效算力=$312\times10^{12}\times0.5=156\times10^{12}$ FLOPS

  训练总耗时=$3\times10^{16}\div156\times10^{12}\approx192$ 秒 ≈ 3.2 分钟

1.4 业务指标（按训练/推理场景拆分）

业务指标基于具体场景区分核心指标，其中训练场景重点关注“算力效率、并行能力、训练周期”，核心指标为训练吞吐量、训练并发量；推理场景重点关注“服务质量、用户体验、成本效益”，核心指标为推理吞吐量、有效吞吐量、推理并发量，具体拆分如下：

训练场景核心关注指标

训练场景的核心目标是“高效完成模型收敛、提升研发效率、控制训练成本”，核心关注 2 类指标：训练吞吐量、训练并发量，具体细节如下：

1. 训练吞吐量（Training Throughput）

定义

指超节点在单位时间内，能够完成的模型训练相关计算总量，是对硬件标称算力输出能力的直观度量，核心衡量维度包括：

- 单位时间内完成的浮点运算量（FLOPs/s），直接对应硬件算力的兑现能力；
- 单位时间内处理的训练样本数、完成的训练轮数，直接关联模型训练进度。

用途

- 衡量超节点理论峰值算力的实际兑现能力，是硬件算力规格选型的核心基础；
- 对比不同厂商、不同配置超节点的算力输出上限，筛选适配模型训练的硬件方案；
- 为模型训练周期、算力成本测算提供基础输入，比如根据吞吐量预估同等规模模型的收敛时间。

决定模型训练的关键表现

- 模型训练总周期长短：吞吐量越高，同等参数规模、同等数据集的模型，完成收敛的速度越快，缩短研发周期；
- 算力成本控制：高吞吐量意味着单位时间内完成的训练进度更快，可降低整体算力投入成本；
- 算力平台训练承载上限：标识超节点在无瓶颈、满负载下的最大训练算力输出潜力，决定可支撑的模型训练规模（如超大参数量模型、超大数据集训练）。

2. 训练并发量（Training Concurrency）

定义

指超节点在保持稳定性能、不出现明显算力瓶颈、不影响训练收敛效果的前提下，可同时承载的模型训练任务数量，核心衡量维度包括：

- 同时运行的独立训练任务数；
- 单训练任务内的并行训练流数（如数据并行、模型并行的并行度）。

用途

- 衡量超节点的多任务调度、资源隔离与异构协同能力，适配多团队、多实验并行的研发场景；
- 支撑研发效率评估，判断超节点能否同时承载多个模型训练、多个实验任务，减少实验排队时间；
- 判断超节点能否满足大规模研发团队、多业务线并行训练的需求，提升算力资源利用率。

决定模型训练的关键表现

- 研发效率：并发量越高，可同时开展的训练实验越多，缩短模型迭代周期（如多参数组合、多模型架构并行测试）；
- 算力资源利用率：并发能力越强，单位算力可支撑的训练任务越多，降低算力资源闲置率；
- 突发训练任务承载能力：如多个研发人员同时提交训练任务、突发大规模模型微调任务时，系统的稳定性与响应速度。

推理场景核心关注指标

推理场景的核心目标是“保障用户体验、提升服务效率、控制服务成本”，核心关注 3 类指标：推理吞吐量、有效吞吐量、推理并发量，具体细节如下：

1. 推理吞吐量（Inference Throughput）

定义

指超节点在单位时间内，能够完成的模型推理相关数据处理总量，是衡量推理服务效率的基础指标，核心衡量维度包括：

- 单位时间内处理的推理请求数（RPS）；
- 单位时间内生成的 Token 数（针对 LLM 场景）；
- 单位时间内处理的图像、语音等多模态样本数（针对多模态模型场景）。

当前主流大模型（LLM）推理服务系统（如 vLLM、TensorRT-LLM 等），普遍将推理吞吐量作为核心基础指标，其与服务成本（$/req）直接强相关，是工业界衡量推理服务效率的主流度量方式。

用途

- 衡量超节点推理算力的实际输出能力，是推理硬件选型、推理引擎优化的基础参考；
- 对比不同配置超节点的推理效率，筛选高性价比的推理硬件方案；
- 为离线推理、批量推理任务的总量测算、时间预估提供基础输入，比如预估单日可处理的文档摘要、批量生成任务总量。

决定模型推理的关键表现

- 批量推理任务处理效率：吞吐量越高，同等规模的离线推理、批量生成任务，完成耗时越短；
- 推理服务的成本效益：高吞吐量意味着单位算力可处理的推理请求越多，降低单请求推理成本；
- 推理算力的理论承载上限：标识超节点在无瓶颈、满负载下的最大推理输出潜力，决定可支撑的推理任务规模。

2. 有效吞吐量（Goodput）

定义

推理场景特有的核心指标，指系统在满足预设延迟约束（如 TTFT/TPOT SLO）的前提下，真正完成的有效推理请求数量。若一个请求因延迟过长被用户放弃、或超过服务约束而无效，即便产生了 Token，也不计入有效产出。

大模型推理服务中最核心的延迟类 SLO（服务等级目标）包括：

- TTFT（Time To First Token）：首 Token 响应延迟，直接决定用户感知的“等待时长”，是实时交互体验的关键；
- TPOT（Time Per Output Token）：输出 Token 间平均生成延迟，表征模型连续生成的流畅程度。

DistServe 论文中明确了 Goodput 的核心价值：将吞吐量与延迟 SLO 绑定，是比单纯吞吐量更优的衡量指标，贴合 LLM 在线服务的本质。以某应用为例：其要求至少 90%的请求同时满足 TTFT＜200ms 且 TPOT＜50ms，则该场景下的 Goodput 定义为——在保证 90%请求达标上述双延迟约束的条件下，系统所能维持的最大每秒请求数（RPS）。

用途

- 同时量化吞吐效率与服务质量：一个指标同时反映推理服务的成本效益（吞吐）与用户体验（延迟），避免“高吞吐、低体验”的假象；
- 真实反映用户实际体验：直接对齐用户感知，即便系统吞吐量再高，若大量请求不满足延迟 SLO，Goodput 会显著下降，倒逼服务优化；

决定模型推理的关键表现

- 推理服务的真实有效能力：不看理论吞吐量，只看满足 SLO 的实际可交付请求能力，是推理服务真正的“有效产能”；
- 延迟与吞吐的综合权衡水平：同时体现 TTFT 响应速度、TPOT 生成流畅度、系统吞吐量三者的平衡能力，是在线推理服务的核心竞争力；

- 推理调度与资源利用效率：反映推理引擎（如动态批处理、KV Cache、调度策略）的实际落地效果，直接影响服务质量与成本；
- 高并发下的稳定性与体验一致性：直接体现高并发压力下 SLO 达标率，避免“峰值卡顿、部分用户体验差”的问题；
- 算力投入的真实性价比：衡量硬件成本换来的有效服务价值，而非单纯的算力占用，是推理场景成本优化的核心参考。

3. 推理并发量（Inference Concurrency）

定义

指超节点在保持稳定性能、不出现明显延迟飙升、满足 SLO 约束的前提下，可同时承载的模型推理请求数量，体现超节点的多用户、多请求承载能力，核心衡量维度包括：

- 每秒处理的推理请求数（QPS）；

用途

衡量在线服务、AI 推理系统并发处理能力与业务承载能力的核心业务指标。

决定模型推理的关键表现

在大模型对话、AIGC 生成等场景中，QPS 表示系统每秒能够稳定处理并返回结果的用户请求数量。

1.5 选型的实际案例

通用计算依据（算力+显存统一口径）

- **推理算力计算**：单 Token 推理计算量≈2×模型参数量，峰值总吞吐 = 单轮计算量 × 业务 QPS
- **推理显存计算**：总显存需求 = 模型参数显存 + KV 缓存显存 + 激活与中间计算显存，量化精度直接决定显存占用
- **训练算力计算**：总训练计算量≈6× 模型参数量 × 总训练 Token 数
- **训练显存计算**：训练所需总最小显存计算：总最小显存 = 模型总静态显存（总参数权重显存 + 优化器状态显存 + 梯度显存）+ 动态显存（激活值显存含 Batch 数据 + 通信与专家路由缓存(MoE 特有)）
- **核心指标**：有效吞吐量、端到端延迟、QPS、显存利用率、算力利用率

场景一：某公司在线 AIGC 对话推理场景

1. 客户实际业务与指标要求

客户业务为**通用对话大模型在线推理服务**，面向 C 端用户提供实时问答、文案生成能力，核心指标要求如下：

- 模型规模：70B 参数稠密模型
- 业务负载：峰值 QPS ⩾ 400，单轮平均生成 512 Token

- 可用性要求：7×24h 在线，算力利用率≥65%
- 要求模型推理显存无溢出、无显存交换导致的延迟飙升

2. 算力与显存指标测算

- **算力需求测算**
  - 单轮请求生成总计算量 = 70B×2×512Token=7.168×10^{13} FLOP
  - 峰值总计算吞吐需求 = $7.168\times10^{13}\times400$QPS=$2.867\times10^{16}$FLOP/s=28.67PFLOPS
  - 硬件总算力≈28.67÷(0.65×0.85)≈51.9PFLOPS，多卡并行效率 85%
- **显存需求测算**

  本次推理采用 INT8 精度部署：
  - 70B 模型参数显存占用 = 70B×1Byte=70GB
  - 单轮 512Token 生成对应的 KV 缓存显存单请求约 1.2GB（由模型 架构参数决定，不再展开计算）

    峰值并发下 KV 缓存总显存需求≈400QPS *1.2GB=480GB
  - 叠加激活值、中间计算临时显存 120GB,（由模型 架构参数决定，不再展开计算）

  系统总显存需求≈70GB+480GB+120GB=670GB

  同时需预留 20% 冗余显存应对突发请求，**最终实际所需总显存≥804GB**

结论：优先满足显存大于 800GB，总算力大于 51.9PFLOPS 的超节点，传统 GPU 服务器受 GPU 卡数限制，总的显存池大小有限。以单卡显存 80GB 为例，8 卡只能达到 640GB，无法满足要求，需通过多 GPU 服务器级联显存，通信效率低，因此难以满足高 QPS 要求。超节点方案具备更大的显存池化、低延迟互联特性，可完美匹配算力与显存双重需求。

场景二：科研机构 1.4 万亿 MoE 大模型训练场景

1. 客户实际业务与指标要求

客户为前沿科研机构，开展 1.4 万亿参数 MoE 大模型训练，目标为通用基座模型，指标如下：

- 模型规模：1.4T 参数 MoE 架构（假设每次激活参数量为 0.5T）
- 训练目标：总 Token 数 2.4 万亿，期望总训练周期 ≤ 28 天
- 算力要求：全天连续训练，由于 MoE 存在通信瓶颈，期望有效算力利用率（MFU）> 40%
- 并行要求：支持专家并行(EP)、数据并行(DP)、张量并行(TP)、流水线并行(PP)的 4D 混合并行
- 长期约束：低功耗、高稳定、全互联网络，便于后续模型迭代

2. 算力与显存指标测算

- **算力需求测算**
  - MoE 模型按激活参数量等效计算总训练计算量 ≈ 6 × 0.5T × 2.4 万亿 Token = 7.2 × 10^24 FLOPs
  - 所需有效算力吞吐 ≈ 7.2 × 10^24 FLOPs / 2419200 秒（28 天）≈ 2.97 EFLOPS
  - 结合 MoE 模型典型的算力利用率（MFU）约 40% 折算，集群所需硬件标称总算力 ≈ 2.97 EFLOPS / 40% = 7.42 EFLOPS
- **显存需求测算**

  本次训练建议采用 BF16 精度：
  - 静态显存（模型及状态）：1.4T 物理参数占用 = 1.4T ×（2 Byte 权重 + 2 Byte 梯度 + 8 Byte Adam 状态）= 16.8 TB
  - 动态显存与冗余：考虑大 Batch Size 下的激活值显存（Activation Memory）、MoE 架构专家并行的 All-to-All 通信 Buffer 缓冲，以及避免 OOM 的安全冗余，实际训练静态与动态显存比通常在 1:1 到 1:2 之间。训练该模型的基础显存需求总量约在 35 TB – 50 TB 之间

总结：

- 要满足 28 天的训练要求，必须构建标称算力不低于 7.42 EFLOPS 的智算集群。
- 对于万亿参数 MoE 模型，普通的 GPU 集群由于跨节点通信延迟高，会导致专家并行（EP）的 All-to-All 通信成为严重瓶颈，算力利用率将大幅跌落，从而无限期拉长训练周期。
- 因此，该场景必须依赖具备**无阻塞高带宽网络（如 InfiniBand 架构或超大规模 RoCE 网络）**的大规模智算集群，结合优秀的 4D 并行切分策略，才能同时满足算力利用率、单卡显存不溢出、通信低延迟的三重核心要求。

2 部署关键要求

2.1 机房环境要求

供电、散热(液冷/风冷选型)、空间布局

随着大模型训练与推理场景的规模化落地，AI 集群正加速从传统通用分布式计算架构，向高密度、高功耗、高互联的超节点架构深度演进，算力密度、供电负荷与散热需求呈指数级提升，单柜功率已从传统标准机柜的数千瓦跃升至数十千瓦乃至上百千瓦。

传统数据中心以标准机柜为核心交付单元，采用低密度风冷散热、分散式供电的建设模式，在供电容量冗余、机柜承重等级、制冷效率适配、布线空间预留等核心维度，已难以适配新一代智算场景下超节点的部署需求，高功率密度部署过程中频繁面临供电瓶颈导致的算力输出受限、散热失效引发的设备稳定性下降、基础设施架构与超节点集成需求不兼容等系统性难题，严重制约智算中心的算力交付效率与长期运维可靠性。

在此背景下，以大功率整机柜为基本交付单元的超节点方案，凭借其高集成度、高算力密度、高能效比的核心优势，已成为当前智算中心提升算力规模化交付能力的主流选择，但其与生俱来的高功率负载、高度集成化设计、风液混合制冷（部分场景为全液冷）的特性，对机房基础设施的顶层规划、方案设计、建设实施及全生命周期运维，提出了更严苛、更体系化的要求，倒逼机房建设从传统“被动适配”向“主动协同”转型。

面向新一代智算中心高质量建设的核心需求，当前行业已逐步形成 “算电冷网一体化、预制模块化、机柜单元化”的整体部署框架，通过供电架构升级、制冷模式革新、空间布局优化与机柜标准化设计的深度协同，实现超节点从单柜部署到集群规模化落地的高效、有序推进。

结合国内多地大型智算中心的实践案例来看，以整机柜为核心的部署模式，并非简单的设备堆叠，而是需要在机房规划初期，就同步完成基础设施与算力单元的一体化设计：

- 供电系统方面，根据超节点功率需求，部分场景灵活采用高压直流供电，提升供电稳定性与能效；
- 制冷系统方面，针对超节点高散热需求，采用风液混合制冷或全液冷技术，机房内气流组织管理，室外采用节能型冷源，确保制冷和节能效果。精准匹配机柜内部热源分布，确保设备长期在适宜温度区间运行；
- 机柜与空间方面，对机房地面进行机柜级承重加固，运输确保高价值的整机柜安全可行性运输。优化机柜布局与布线通道，满足超节点高密度布线与设备扩容需求；
- 运维方面，部署智能运维管控系统，实现对机房基础设施、超节点设备的实时监测与精准调度。

通过以上全链路协同设计，为超节点大规模、高可靠、快速交付提供稳定的基础设施基座，同时为后续算力扩容、技术迭代预留统一接口，保障智算中心长期可持续发展。

本章节，目的是阐述数据中心机房液冷机柜与液冷机房的接口设计，通过标准化接口实现液冷机房对多厂家液冷机柜设备的兼容部署，有效提升基础设施部署效率及运维灵活性。

机柜基础参数标准化

- 机柜尺寸与结构：
  - 宽度：600mm 公差要求±1mm（-3mm~0mm）
  - 深度：1400mm（不含机柜门）；公差要求±1mm
  - 高度：2250mm~2300（带脚轮）；
  - 整柜灌液重量：约 1.8 吨
- 并柜结构：

  机柜底部通过螺栓孔与机柜基座连接，通过机柜并柜片进行锁紧。
- 机柜承重抗震要求：：

  机柜设计可以满足 2000kg 静载及相关整机柜运输测试要求，机柜设计需要满足中国《建筑抗震设计规范》（GB 50011-2010）和《YD 5083-2005 电信设备抗地震性能检测规范》8 烈度抗震要求或 NEBS 认证 Zone 3 抗震要求。

设备运输和减震要求

- 超节点的运输货车要求采用公路型精密仪器气垫减震货车。
- 机柜必须垂直固定，底部加防滑减震垫。超节点产品固定应满足减震防撞击的部署要求。
- 机柜包装安装防倾斜功能标签。
- 运输全程倾角不大于 3 度，超过 5 度立刻停车调整。
- 卸货到机房机柜预留位置，整个运输过程要求保持直立。
- 裸柜运输电梯、室内短坡道坡道倾斜度不大于 5 度。
- 连续坡道大于 50 米的情况坡道不大于 3 度，并设分段缓坡。
- 机房的荷重需按超级点整机柜的落地面积复核荷重，如果大于 12kn/平米的机房设计荷载，考虑机房地面局部做散力架均重。

机房环境要求

- 交付机房不能见明灰。
- 机房照度满足 500lx 的机房标准。
- 温湿度满足机房标准。

供电接口标准化

- 机柜侧供电类型与参数。
    - 整机柜供电系统架构：整机柜内部有多个 Power Shelf 模块，每个 PowerShelf 模块采用独立供电和线缆连接。机柜供电仅采用上部走线方案。
    - PowerShelf 电源输入制式：支持 380V AC（三相五线制支持供电模式）， 支持 AC+AC 双路输入。
    - PowerShelf 电压输入波动范围要求：198Vac～264Vac；整机柜正常工作时，输入电压波形失真度≤5%。
    - PowerShelf 供电接口：采用多路 IEC 60309-2 工业连接器（3P+N+E），额定电流 63A，公头从机房取电。
    - 每个 PowerShelf 均有 2 路供电，采用工业连接器及线缆（Power Cable）与外部供电设备连接。
    - Power Cable:承受电流不小于 63A，截面积不小于 16mm2，机房侧提供母头与 Power Shelf 工业连接器公头对接。
    - 电源线缆工作环境温度：工作温度：5℃~40℃； 工作湿度：10%~90%。
    - 机柜供电单元的安全要求应符合 GB 4943.1 的规定，无线电骚扰应符合 GB/T 9254.1 的规定，谐波电流应符合 GB 17625.1 的规定。

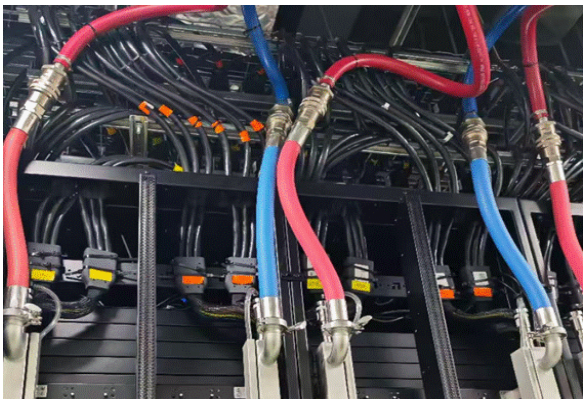


图170 数据中心机房水电走管示意图

- 机房侧配电与保护设计。

  整机柜配电设计和选型：可采用列头柜配电方式和小母线等配电方式。配电方式的选用应考虑业务特性和机柜容量分配、调度和未来扩容等需求。整机柜配电系统需考虑如下内容。

  - 液冷机柜区域：建议预留末端配电扩容空间，满足后续扩容和功率升级需求。
  - 液冷机柜配电开关和降额建议：各支路输出微型断路器之间宜预留散热间隙，安装间距可≥9mm。配电系统断路器应散热良好，降额系数＞0.9。
  - 液冷机柜配电监控：回路宜选用具备支路电流检测、显示及支持监控功能的产品。

  整机柜配电路由设计：A 和 B 路电源桥架全段尽量不交叉。

  - 桥架应至少配有 60%的备用空间（按照机柜演进预留空间），AB 路电缆不应敷设在同一路由的多层桥架的不同层。
  - 电气桥架应 与 IT，ELV 等其他桥架物理隔离，以避免潜在的 EMC 干扰。
  - 不应将电缆桥架穿越在 水，卫生设备，喷水灭火器和任何其他水管下。
  - 强电桥架于 IT/ELV 桥架之间的垂直距离应为 200mm 如受条件所限，应至少达到 150mm。
  - 配电箱需位于机柜上方，需测量机柜到电源插座的距离（需考虑电源线弯曲半径＞100mm）。

  整机柜配电设备失效隔离和保护。

  - 支路断路器：采用微型断路器，参数最低要求如下：Icu≥15kA，Ics≥50%Icu；脱扣器特性：C 型或 D 型（Z 型）（现实中都是 C 型），采用热磁脱扣器具有二段保护特性（过载长延时、短路瞬时），极数：满足电源设计要求。
  - 整机柜电源内部故障,前级配电设备 C63\C80 空开不跳（前级空开是按电流设置跳闸的，大于设定值就会跳闸，无法设置）。
  - 整机柜默认电源框的电缆的进、出线方式要求上进上出，小母线插接箱\强电线缆桥架到机柜按照空间，整机柜进出线需考虑电缆弯曲半径要求。
  - 整机柜连接的所有电缆均应符合 YD/T 1173-2016 通信电源用阻燃耐火软电缆的要求，各连接电缆的线径应满足设计载流量的要求。电缆和母线的绝缘层或外护套颜色应符合 YD/T 585 的要求。

接地与安全接口标准化

- 整机柜接地要求
  - 保护接地（PE）：机柜框架、冷媒管路、金属外壳均需接地，接地电阻≤1Ω。
  - 机柜接地接口：机柜顶部前后及底部前后预留 M6 或者 M8 接地点方便机柜与机房接地。
  - 机柜主接地线位置：支持上走线和下走线（机柜顶部、机柜底部前后均设置有 M6 或者 M8 主接地点）。
  - 机柜-机房接地线缆规格：线缆的截面面积不小于 16mm²，线缆采用 OT 端子，适配 M6 或 M8 螺钉。
  - 整机柜内各带电回路（该回路不直接接地）对地（或柜体）绝缘电 阻应≥10MΩ（500V 兆欧表测量 1min 后读数）。
- 机房安全防护

  机房配电柜防雷要求：
  - 源列柜防雷应满足整机柜要求。
  - 浪涌保护器采用类型、保护电压、标称放电电流等参数详见设计图纸要求。
  - 浪涌保护器具备远程监控状态的触点，支持远程监控。
  - 浪涌保护器接线类型宜采用 4+0 的共模保护。
  - 浪涌保护器须配置专用后备保护开关，不接受 MCB 或隔离开关、熔断器等。
  - PD 支持热插拔维护。

  机房接地和绝缘电阻：
  - 宜选 TNS 系统（当地法律法规有其它要求时除外）。接地网不应大于 1 欧姆。每列整机柜建议至少有一个接地排，接地排的尺寸应参考 TIA 607。
  - 绝缘强度：各带电回路对地（或柜体）以及两个非电气连接的带电回 路之间，应能经电压 1890V/50Hz 正弦交流电压试验，经 1min 无击穿或飞弧现象，漏 电流≤10mA。

液冷接口标准化

液冷机柜 manifold、排水水管均为下走管设计 ,机房地板需要预留对应的净空不小于 200mm（不允许布置其他设施）和开孔位置，否则会影响液冷系统走管。

- 液冷工质规格
  - 液冷工质类型：支持去离子水+缓释剂/除菌剂、20%~25%乙二醇水溶液（-20℃~50℃工况）；20%~25%丙二醇水溶液（-20℃~50℃工况）。
  - 入口温度范围：25℃~40℃，进液水温＞机房环境露点温度+3℃。最终根据实际特殊机型需求确认供液温度。

- 液冷工质规格：不同类型（乙二醇、丙二醇、去离子水+缓蚀剂）工质各项离子参数，需满足对应的散热和质量要求后，才可接入机柜。

- 液冷 manifold 管路接口规格
  - Manifold 接口位置要求：液冷机柜 manifold 接口和机房对接接口位置宜布置在机柜下方的斜后方。管路应合理分成模块化预制管路，现场少量对接安装，不允许现场焊接，尽量减少现场装配工程量，以确保环路对接质量。
  - Manifold 接口高度位置最小要求：主机房架空地板设计高度应考虑液冷管路部署和维护要求。因 DN50 以上规格软管弯曲半径和局部应力较大，安装维护困难，架空地板设计满足机柜需连接 DN50、DN65 规格液冷软管的安装空间要求，接管高度至少预留 400mm。

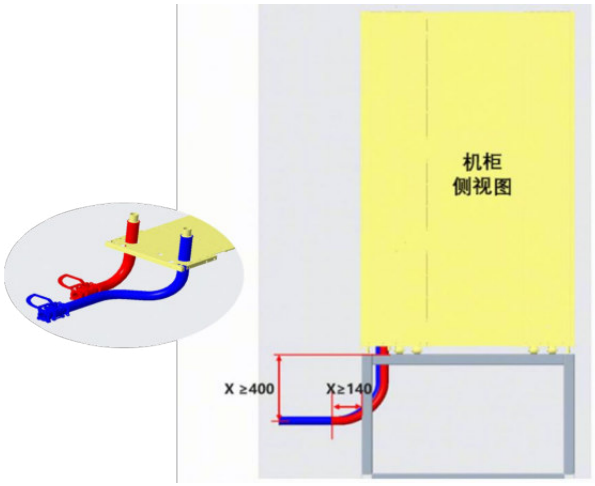


图171 机柜 Manifold 走管示意图

  - 接口类型：二次侧分支管路应安装可物理断开自封式互锁型球阀与二次管路软管末端的球阀进行连接。

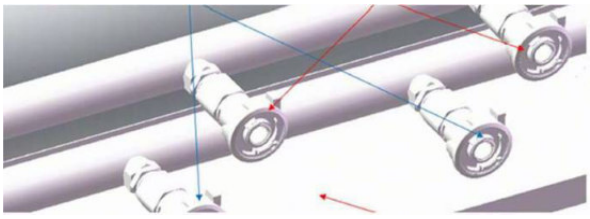


图172 二次侧管路示意图

  - 机柜 manifold 接口尺寸：DN50，采用 ISO2852 标准法兰与软管对接连接；机房侧最小的允许的安装空间。
  - 管路标识：液冷管路入口/出口标注颜色（入口蓝色、出口红色），含流向箭头。

- 液冷散热能力、流量规格
  - CDU 流量设计要求：CDU 二次侧管路应采用环路系统设计，需保证各整机柜支路流量满足设计流量。CDU 的泵应该在二次侧环网设计、环网上阀门、分集水器、服务器冷板模组的压力损失条件下；在单点故障模式下，最不利机柜能达到单机柜设计流量的 额定需求。

- 二次侧环网管路设计：硬管应为预制 304 及以上不锈钢钢管。
- 二次管路流量设计：参考业界标准的液冷设计规范《数据中心冷板式液冷系统技术规范》中液冷系统设计应进行水力计算要求， 各并联环路之间的压力损失相对差额应小于等于 15%。

- 液冷漏液检测和排水
  - 二次环路侧应沿管路敷设/缠绕漏水告警绳，需要 1 米级有漏液点定位功能，漏水绳上须带有距离标签，机房分水管应安装在托水盘中，托水盘应该覆盖。
  - 机柜排水管规格： 尺寸：排水管外径 12mm~20mm 长度：1500mm。
  - 机房侧排水管安装要求：满足顺畅排水要求，不得堵塞，机房排水管建议采用弯管连接。
  - 机房排水口要求：排水口设计位置需靠机房底部，且与机柜底座面留有较大高度落差；排水口分别在机柜出管范围下方区域，确保排水顺畅；机房排水口需要匹配机柜排水管路需求。

机柜底座与地板开孔标准化

- 机柜底座设计

  整机柜应安装在专用的机柜底座上，底座设计应根据整机柜重量、支脚位置、液冷管路接口要求确定。
  - 机房承重和地面要求：楼板活荷载应≥16kN/m²，机房地面应尽量平整，误差不宜超过±3mm/平米。
  - 底座承重要求：底座需满足整机柜正式运行的动载荷，包括工质液、线缆重量。
  - 底座固定和调平：底座顶部预留机柜安装孔，底座底部设置可调支脚和固定螺栓。
  - 底座高度：底座材料为镀锌钢板焊接（耐腐蚀）；架空地板最终的高度需要根据机柜液冷流量，软管的弯曲半径、支路阀门电动阀、手动关断阀所需的安装尺寸、液冷微模块总功率来最终确认。

    底座高低示例：二次环路设计 4000LPM 总流量，需要 DN200 主管+DN50 支管，设计需要 800mm 地板为例：

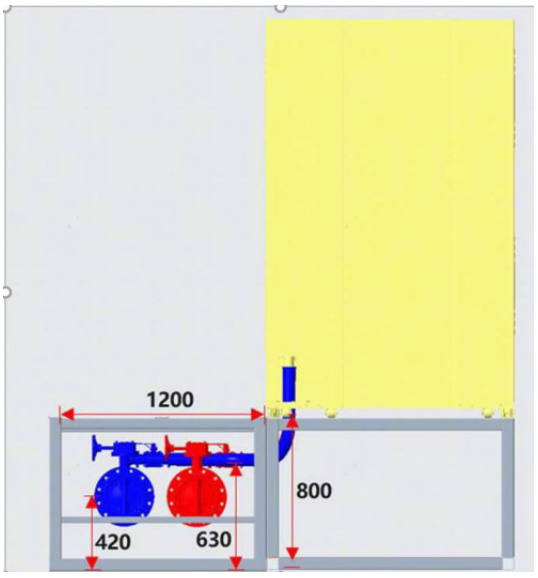


图173 走水管路位置示意图

- 固定方式：建议底座通过 4 个膨胀螺栓孔（M12×120mm），与地板固定；机柜底座螺栓开孔反面若有加强筋（建议螺栓孔反面加强筋布局在螺孔周围），加强筋占用螺栓长度，需核对螺栓长度（标识抗震连接螺栓孔的位置）是否足够。
- 底座上顶板设计要求：为支撑机柜调平，机柜底座位置钢板平面度要求如下：偏差不宜超过±2mm/2m。

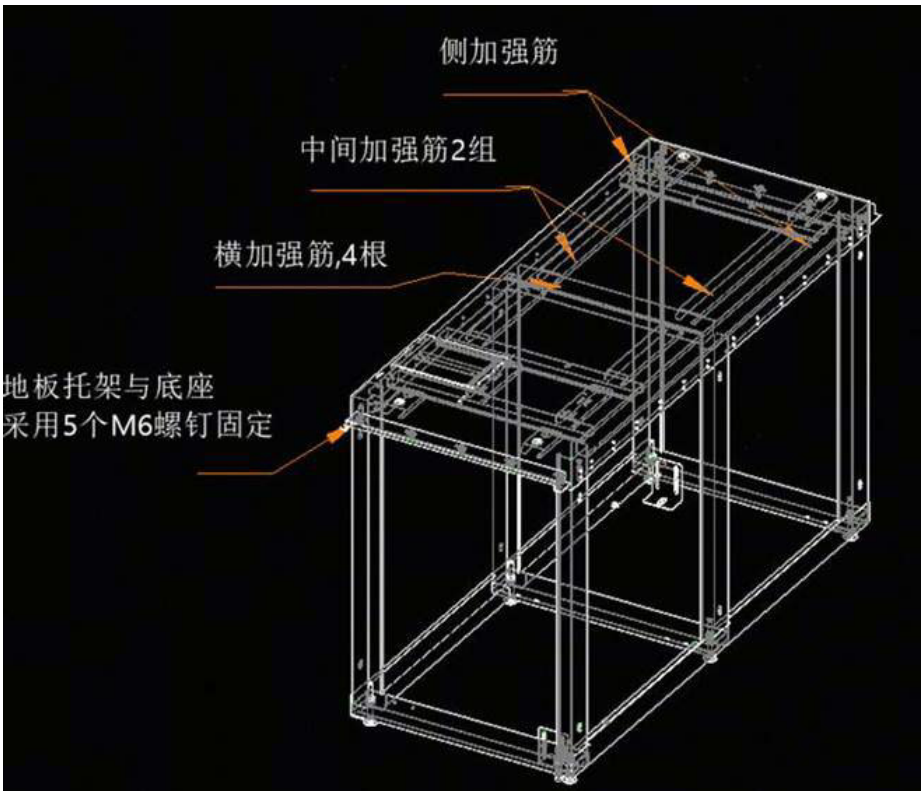


图174 机柜加强筋示意图

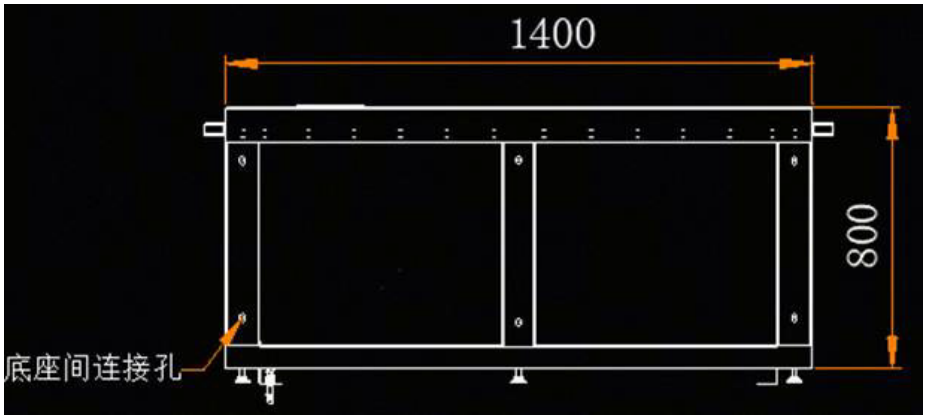


图175 机柜底座位置钢板平面度要求示意图 1

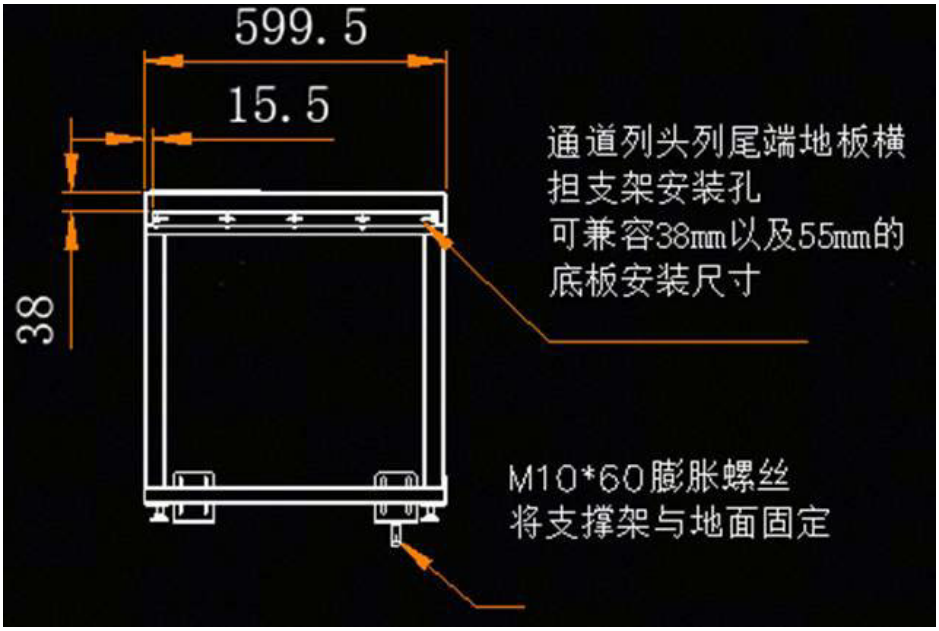


图176 机柜底座位置钢板平面度要求示意图 2

- 底座顶板开孔规范

  底座上顶板开孔设计要求：底座角钢在走管/过线区域开缺避位，机柜地面开孔尺寸要求。

  - 开孔位置：机柜正下方，与底座投影重合，偏差≤±5mm，开孔位置为 manifold 和排水管出管位置。

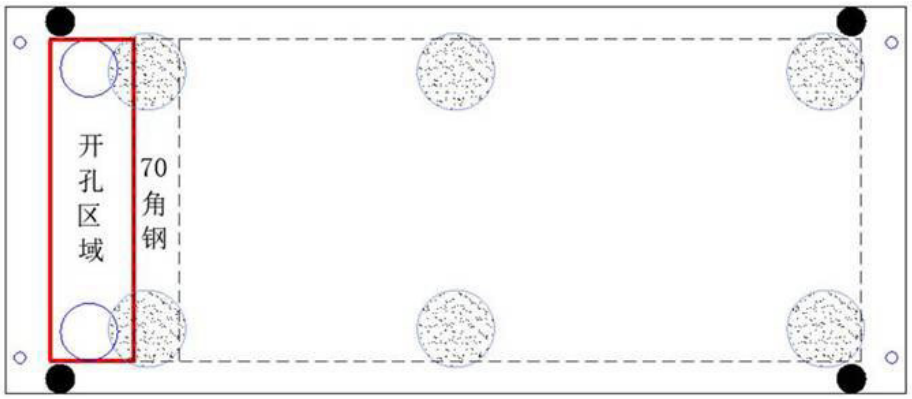


图177 机柜底座顶板开孔示意图

  - 开孔尺寸：≥机柜底座外框尺寸（如 600mm×1400mm 机柜，开孔≥620mm×1420mm）。
  - 开孔边缘：倒角处理（R≥5mm），防止划伤线缆/管路。
  - 地板承重：开孔区域地板需加固（如钢质支撑梁），满足机柜整体承重需求。
  - 底座开孔区域安装金属盖板：用于遮挡多余开孔，同时盖板具有折弯边，能防止机柜过推导致滚轮掉入进线孔。

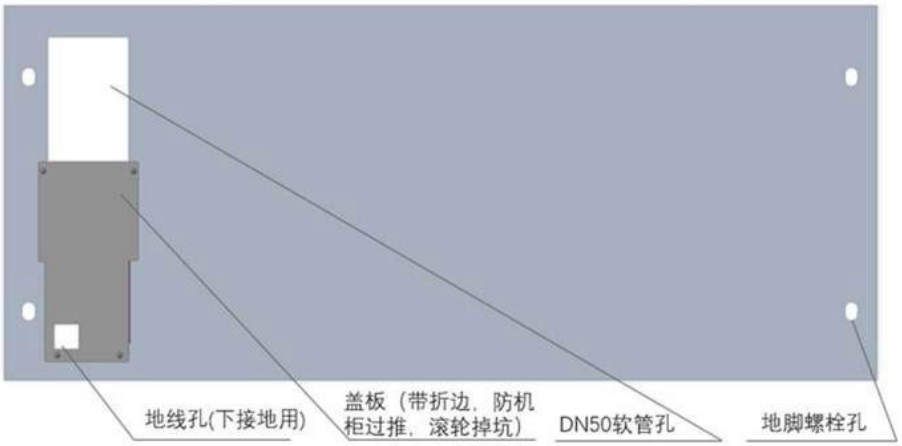


图178 机柜底座顶板盖板安装示意图

整机柜风冷接口标准化

考虑到当前业界部分超节点通常采用风液混合散热模式，需保留辅助风冷能力。整机柜风冷散热：风冷空调末端可采用远端风墙、近端风墙、列间空调等形式，对应的气流组织方式为数据机房内气流组织方式分为弥漫送风、地板下送风、列级送风方式。均可满足液冷整机柜的风冷部分散热需求。机柜区域应设置冷热通道隔离措施，宜设置封闭热通道实现整机柜风冷高效散热。 风冷散热需要考虑机柜演进和风冷散热能力需求。

- 风冷散热方案
  - 机柜气流组织：前进后出，与机房空调送风方向一致。
  - 机柜进风要求：5~35℃，20%~80% 不凝露。
  - 风量要求：回送风温差≥12℃，送风风量基于整机柜风冷散热功率确定。
  - 海拔要求：环境允许的最大海拔高度为 1000m；对于建设在海拔高度超过 950m 的服务器机房，允许的最高干球温度应降低 1℃/300m。

2.2 集群搭建流程

硬件部署

超节点集群在数据中心的部署主要涉及三种网络，分别是 Scale Up 网络，负责高带宽域的 AI 算力组网；Scale Out 网络，负责 AI 算力集群大规模扩展组网；通用数据中心网络 Frontend，负责通算业务多租户隔离提供主机侧管理和控制平面流量，同时为超节点提供高性能存储网络。集群带内/带外管理网络不在此赘述。

表22 超节点集群网络部署说明

网络类型	别称	硬件组网	物理互联特征	业务形态与流量映射
Type 3: Scale-Up	Local(南向网络) 构建超节点内部的高带宽域(HBD)	CLOS组网，无阻塞通信	超节点集群内，视集群规模,支持电互联，也支持光电混合	张量并行（TP） 专家并行(EP)流量
Type 2: Scale-Out	Backend(北向网络) 跨HBD域集群分布式训练加速	多级以太交换，视集群规模，支持多轨组网（Multi-rail）	高性能交换机，光模块互联	流水线并行（PP） 数据并行（DP）及全局梯度同步
Type 1: Frontend	DCN(数据中心网络) 数据吞吐、管理与存储访问	标准DCN以太网架构，通过智能网卡接入	通用光电混合互联	VPC网络支持多租户隔离,提供主机侧管理和控制平面流量 Checkpoint读写，提供高性能存储网络

在纵向扩展层面，Scale-Up（Type 3）网络被定义为算力互联网络，其核心业务映射为承载张量并行及专家并行流量，维持大带宽低时延组网，Scale-Up 网络的扩展范围通常约束在超节点集群内部。

在横向扩展层面，Scale-Out（Type 2）网络（即参数面网络）负责将超节点间组成万卡至更大规模集群。该网络广泛采用 CLOS 架构，根据实际业务流量，多级交换中支持带宽收敛，也支持多轨组网（Multi-Rail）设计。在多轨模式下，支持大规模集群组网中同轨流量仅需一级交换，极大减少多级交换引发的 Hash 冲突与额外时延。硬件层面上，Scale-Out 网络高度依赖高性能大基数交换机，以实现跨 HBD 域互联。在业务映射方面，负责数据并行和流水线并行流量。

在业务端接入层面,Frontend(Type 1)网络即传统的数据中心网络(DCN),通常搭配智能网卡(Smart NIC)以支持安全加密与多租户隔离功能。Frontend 网络在 AI 训练中，主要负责业务面的大容量存储交互，包括海量训练样本文件的上载任务，以及训练过程中模型参数 Checkpoint 的周期性持久化存储。

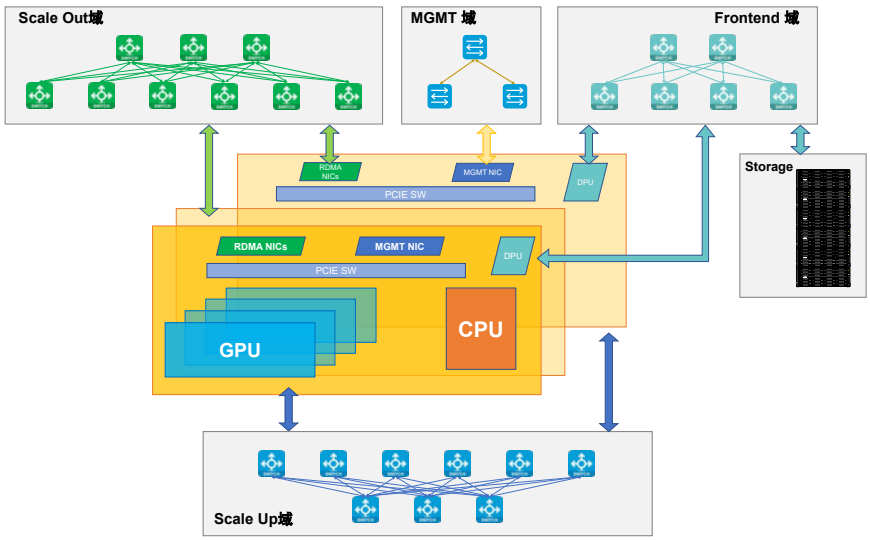


图179 超节点集群组网架构示意图

软件部署/配置

基础组件应用部署

1. 安装环境准备

版本配套关注重点：智算环境的版本配套核心遵循全栈软硬件深度适配、层级组件强绑定原则，需兼顾服务器硬件、操作系统及内核、算力加速卡的驱动与固件、网络适配卡的驱动与固件、分布式通信类组件、存储系统及管理平台组件等全链路环节，确保各层级、各类型组件间版本适配，同时匹配智算场景的组网架构与业务运行要求。

配套发布与测试：由于超节点形态目前尚未形成标准，因此智算相关配套版本由超节点整机厂商完成全链路多阶段测试验证后正式发布，测试涵盖单机层面的软硬件兼容性、基础功能可用性，集群层面的多节点协同能力、高负载下的稳定性，以及实际智算业务场景下的训练、推理等核心能力适配性验证，经多轮迭代优化，确认无兼容性、稳定性问题后，才会发布标准化的配套体系。注意事项：需严格遵循厂商发布的标准化配套清单进行软硬件搭配，禁止跨版本随意混用，避免因版本不兼容引发功能异常、业务运行故障。

超节点安装部署内容主要包括：

- 操作系统安装与初始配置

  智算环境版本配套需遵循全栈适配原则，兼顾各链路环节以匹配组网和业务要求。

- 系统内核与运行环境固化

  在目标硬件上完成操作系统安装、初始化及基础参数优化。

- 硬件驱动与固件适配

  安装适配的系统内核并调优固化，配置基础运行时环境。

- 分布式通信与算力库栈部署

  部署分布式通信等库栈，与底层版本对齐并统一配置相关参数，避免影响集群通信和任务效率。

- 集群网络规划与配置

  规划网络隔离与转发策略，满足智算环境对网络的高要求。

- 存储接入与挂载配置

  确保存储与计算节点版本协议匹配，规范挂载策略避免访问冲突。

- 容器 / 虚拟化与运行时环境部署

  部署容器 / 虚拟化平台及相关接口，实现算力资源统一调度训前环境与集群健康检查。

  执行集群健康检查，形成可复现的检查体系。

- 业务任务部署与集群维护体系

  基于使能平台部署大模型推理实例，提供标准化调用接口并支持分布式推理与高速网络适配。

- 基础应用部署

  对应用平台部署关键节点进行通用框架性描述。

- 应用平台部署关键节点

  上传模型及配置文件、完成注册版本管理，创建并启动推理实例且监控状态。

- 模型部署与实例配置

  保障模型文件完整和计算资源适配，确保分布式推理正常。

- Benchmark 性能测试部署

  搭建测试环境、配置参数并关联推理服务，执行测试并分析性能指标。

- 部署后验证与问题处理

  验证推理服务可用性，监控测试异常并处理常见问题。

从上面看安装部分过程非常复杂且繁琐，对于超节点由于每个计算 Tray 均要完成这些工作，过程会更加负责。因此一般需要有管理控制平台，实现对超节点的软件栈进行统一、自动的安装部署。

3 运维与监控

3.1 概述

随着大模型训练、多模态计算、高性能科学计算等智算场景快速落地，高密度、紧耦合的超节点已成为新一代智算中心核心算力载体。业界通用定义，超节点是基于专有 Scale-Up 高速互联网络，将多颗 AI 加速芯片集成的紧耦合算力系统，分为单机柜高密度、跨机柜大规模协同两种形态，支持域内统一内存编址、芯片间 TB 级带宽互联与微秒级时延，可突破传统 Scale-out 集群通信瓶颈，支撑大模型长周期训练与大规模并行计算。超节点整合计算、高速交换、高密度供电、液冷散热、带外集中管控五大模块，全 Mesh 紧耦合架构与高密度部署模式，对运维体系提出了区别于传统集群的全新技术要求。

超节点运维核心挑战

超节点的高密度、高耦合、高敏感特点，核心痛点集中在硬件、技术、业务、运维四大维度，各类痛点相互关联、层层传导，形成了区别于传统运维的独特挑战。

1. 硬件层面：高密高负载，单点故障风险极高

超节点单柜因高功耗、高集成设计，对硬件可靠性和运维精度要求极高，一是供电压力大，单柜功耗达数十至上百 kW，电压波动需严格控制在±5%以内，否则易引发芯片离线或训练任务中断，容错率极低。二是液冷运维严苛，需精准控制温度、流量与压力，一旦漏液或温度超标，可能直接导致硬件损坏，属最高等级故障。三是故障影响全局，单卡或互联链路故障会迅速扩散，造成整组算力性能下降甚至“雪崩”，对故障预防要求远超传统集群。

2. 技术层面：全栈协同要求高，一致性管控难度大

超节点全链路技术栈耦合度极高，任何一环微小偏差都会引发连锁技术问题：一是全栈固件/驱动一致性要求严苛，BMC、BIOS、加速芯片、交换机固件版本不统一，会直接导致通信异常、性能大幅衰减，甚至无法组网运行，全集群同类型固件一致性需达到 100%；二是高速互联网络零容错，必须保证低时延、零误码、无拥塞，普通网络运维手段无法适配专用 Mesh 互联与 IB/RoCE 高速网络的管控需求，误码率需控制在 10^{-12}以内；三是异构资源统一调度难，多芯片、多模块协同调度，对运维平台的智能化、自动化能力要求极高，传统调度工具无法适配全局池化调度需求。

3. 业务层面：适配大模型场景，稳运压力突出

超节点核心承载万亿参数大模型训练、高端科学计算等高端算力业务，业务特性决定了运维特殊需求：一是大模型训练周期长达数周甚至数月，任务中断重跑成本极高，不仅浪费算力资源，还会延误项目进度，对运维稳定性要求近乎苛刻；二是算力利用率保障难度大，需兼顾多任务调度、多租户资源隔离，避免资源抢占、任务冲突导致效率大幅下滑；三是断点续训、异常恢复能力刚需，需配套专项运维机制，保障训练成果不丢失、任务可快速接续，降低业务损耗。

4. 运维层面：专业门槛高，传统模式无法适配

现有传统 IT 运维体系难以直接适配超节点需求：一是故障定位难，全栈深度耦合导致故障根因复杂隐蔽，传统逐点排查效率极低，难以快速定位根源，需依托全栈可观测技术实现精准溯源；二是运维复杂度高，需同时掌握智算硬件、液冷散热、高速网络、AI 平台、大模型基础等多领域专业技能，行业专业人才稀缺；三是自动化运维刚需，人工运维无法满足秒级故障感知、快速自愈需求，智能化运维能力缺失会直接导致运维效率低下、故障频发，必须搭建自动化闭环运维体系。

超节点运维核心定位与目标

超节点运维，需面向高密度紧耦合算力单元的全生命周期技术保障，聚焦专有 Scale-Up 网络、一体化硬件、长周期并行业务运维，承担硬件全生命周期管控、链路可靠性保障、故障闭环处置、算力与能效优化，确保超节点可用性与算力释放效率。

超节点运维核心目标明确为四点：一是构建全维度监测体系，提前预判隐性故障、隔离全域风险，减少非计划停机；二是建立标准化故障处置流程，实现精准定位、快速自愈、无扰动恢复，保障业务连续；三是严控非必要变更，最小化运维扰动，适配长周期高稳业务需求；四是完善产品的运维框架，强化全链路遥测、自动化与智能诊断能力，实现高可靠与高算力利用率兼顾，推动运维从被动抢修向主动预判转型，实现目标量化指标：

- **7×24 高可用保底：**保障超节点整体可用性达标，最大限度压缩计划外停机时间，杜绝核心算力业务中断，满足大模型长周期训练的连续性要求，行业基准整体可用性≥99.9%；
- **算力效率最大化：**维持算力有效利用率稳定在行业基线以上，避免性能衰减、资源闲置与算力浪费，实现算力投资价值最大化，核心算力有效利用率≥90%；
- **大模型业务不间断：**保障长周期大模型训练任务平稳运行，支持断点续训、异常快速恢复，避免任务中断重跑带来的时间与成本损耗，断点续训成功率 100%；

- **全链路合规可控：**实现硬件、网络、系统、数据全维度可监控、可管理、可追溯，满足智算中心安全合规与运维审计要求，保障算力运行安全，符合网络安全等级保护相关规范。

3.2 超节点运维核心范围

超节点作为软硬件深度耦合的一体化算力单元，其运维范围并非单一硬件或软件层面的零散管控，而是遵循“从物理底座到业务输出”的全链路逻辑，结合主流超节点产品技术架构与国内智算中心实操运维经验，划分为硬件基础设施层、高速互联网络层、系统与软件栈层、业务与算力服务层四大核心层级。

硬件基础设施层运维

硬件基础设施运维是超节点算力输出的物理根基，也是全链路运维体系的底层核心支撑，其运维逻辑完全适配超节点高密度集成、百 kW 级高功耗、芯片级紧耦合的专属特性，彻底摒弃传统服务器硬件运维“单机巡检、粗放管控、被动抢修”的落后模式，推行全生命周期闭环、全参数精准管控、全隐患前置预防的精细化运维体系，核心目标是筑牢物理层安全防线，杜绝单点硬件异常引发全局算力雪崩，保障超节点长期满负载稳定运行，为上层网络、软件、业务运维奠定坚实物理基础。

硬件基础设施的核心运维对象，主要包含硬件算力核心、高密供电、液冷散热等，如下：

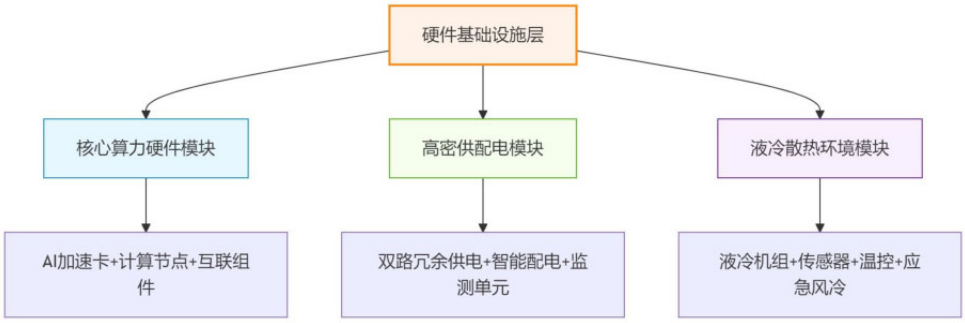


图180 硬件基础设施层运维模块架构图

模块名称	核心组成部件	核心监控指标
核心算力硬件模块	AI 加速芯片（GPU/NPU）、计算 tray、交换 tray、正交无背板高密机箱、200G/400G/800G 专用高速互联光模块、芯片级直连专用线缆、硬件传感器模组	芯片温度、显存温度、实时功耗、功耗偏差、ECC-UE/CE错误、算力降频时长、利用率、在线状态、固件版本 光功率、模块温度、偏置电流、链路误码率、时延抖动、端口利用率
高密供配电系统模块	高密定制化 PDU、双路 UPS 不间断电源、高压直流母线、智能精密配电模块、冗余备用电源、供电电压电流监测单元	输出电压波动、模块负载率、双路均衡度、支路电流、接口温度、冗余切换时长
液冷散热与环境硬件模块	冷板 / 浸没液冷散热模块、供液回液闭环管路、分布式漏液检测传感器、精密温控单元、循环泵组、机柜环境温湿度传感器、应急风冷备份风机、液冷参数实时监测仪表	供水温度、供回水温差、流量波动、电导率、水泵状态、漏液信号、露点温差

高速互联网络层运维

高速互联网络是超节点全栈架构的核心技术命脉，分为 scale-up 芯片级直连扩容网络（单集群内部）与 scale-out 跨机柜交换网络（跨集群级联）两大板块，其中 scale-up 网络是大模型分布式训练的核心通信底层，采用 NVLink/Ulink/ 自研 Mesh 专属互联协议，运维优先级、管控精度、容错要求远高于传统网络，其运维有如下特征：

- **紧耦合无边界，全域风险强关联性，**Scale-Up 网络采用全 Mesh 点对点互联架构，单节点、单端口、单条链路异常，不仅影响局部通信，极易引发全域 PFC 死锁、流量拥塞、算力协同中断等问题，甚至导致整域算力瘫痪，不存在传统网络的边界隔离缓冲。运维软件必须具备毫秒级故障感知+全域快速隔离能力，单点异常即刻阻断扩散，无法依靠传统网络的分段排查、渐进式处置模式，对软件的实时性和联动管控能力要求极高。
- **高带宽低时延，运维指标极致精细化，**承载 AI 大模型训练的 TB 级互联带宽、微秒级时延需求，常规网络的带宽、连通性监控完全无法满足，运维需聚焦微秒级时延抖动、纳秒级同步偏差、1e-15 量级低误码率、光功率微漂移等极致精细指标，微小指标波动就会导致算力降频、训练任务掉点。运维软件必须支持秒级高频采集、基线动态校准，具备微小异常的预判识别能力，而非传统网络的分钟级巡检、粗放式阈值告警。
- **专属硬件协议，无通用运维兼容性，**Scale-Up 网络采用厂商专属互联协议与专用芯片（NVSwitch、华为灵衢、阿里 CIPU 交换单元），不兼容传统以太网、RoCE 常规协议，无法用通用网络运维工具监控管理，属于超节点专属封闭管控域。运维软件必须适配专属遥测接口、私有通信协议、专用芯片诊断指令，通过 RMC 统一归集数据，无法依托通用 SNMP、常规网络监控平台实现管控，对软件的专属协议适配能力要求严苛。
- **拓扑刚性固定，严禁动态变更与随意改动，**为保证全域算力统一编址、内存协同，Scale-Up 网络采用刚性全 Mesh 拓扑，部署完成后拓扑不可随意变更、链路不可随意插拔，拓扑一致性直接决定算力集群可用性，轻微拓扑偏移就会导致整域无法组网。运维软件需具备拓扑自动发现、规划基线比对、每日强制巡检能力，异常即刻告警，不支持传统网络的动态扩容、链路热插拔无感替换，运维变更必须经过软件前置校验与全域合规核查。
- **算力网络深度耦合，运维需算力-网络协同联动，**Scale-Up 网络与 AI 加速卡算力深度绑定，网络异常直接引发算力降频、ECC 错误增多，算力负载过高也会反向导致网络拥塞、死锁，二者运维无法割裂。运维软件必须实现算力状态与网络指标联动监控、协同调优，网络拥塞时联动算力做负载限流，算力异常时同步核查网络链路质量，打破算力与网络的运维数据孤岛，实现一体化闭环运维，而非传统网络与服务器的分离运维模式。

针对 Scale-up 网络特点，提供如下运维能力

- **7×24 小时实时链路监控：**依托全栈可观测平台，对所有 Mesh 直连链路实行毫秒级监控，实时采集光功率（如果有）、时延、误码率、带宽负载等核心数据，重点监控芯片间直连链路状态，一旦指标超标立即触发分级告警，杜绝隐性链路异常累积；
- **拓扑完整性定期巡检：**每日执行自动化网络拓扑校验，核查 Mesh 全网、IB 交换网拓扑结构完整性，排查链路松动、交换芯片离线、路由配置异常等问题，每周开展全链路深度巡检，生成拓扑健康度报告，确保无单点链路盲区；

- **流量动态调度与拥塞防控：**针对大模型训练高频数据交互场景，通过智能调度平台动态调控全网流量，避免单条链路带宽过载引发拥塞，优化通信路由配置，保障多任务并行时通信效率均衡，算力与通信资源协同匹配；
- **故障链路快速自愈与隔离：**内置链路自愈模块，检测到误码超标、链路中断等异常时，30 秒内自动切换备用链路，隔离故障链路，避免故障扩散引发全局通信瘫痪；硬件故障类链路异常，4 小时内完成光模块或线缆更换，更换后复测指标达标；
- **配置一致性刚性管控：**全网交换芯片、Mesh 互联模块固件版本、路由配置参数 100%统一，严禁私自修改配置，所有配置变更需经过测试验证、审批流程，变更后自动同步全网络节点，杜绝配置偏差导致通信异常。

系统与软件栈层运维

系统与软件栈层是衔接物理硬件底座与上层业务算力输出的**核心枢纽桥梁**，也是保障超节点全栈软硬件深度协同、消除算力隐性损耗、规避通信故障的关键管控层级。相较于传统服务器集群零散化、单机化的软件运维模式，超节点软硬件高度耦合的特性，决定了本层运维必须摒弃单点调试、人工逐台操作的粗放模式，全程围绕**“统一标准、全局一致、批量管控、版本刚性合规”**核心逻辑，构建全生命周期闭环运维体系，从根源杜绝因版本错乱、参数漂移、兼容性失衡引发的算力衰减、多芯片协同失效、训练任务中断等隐性问题，为高速网络层、硬件层的稳定运行提供软件层面的刚性支撑。

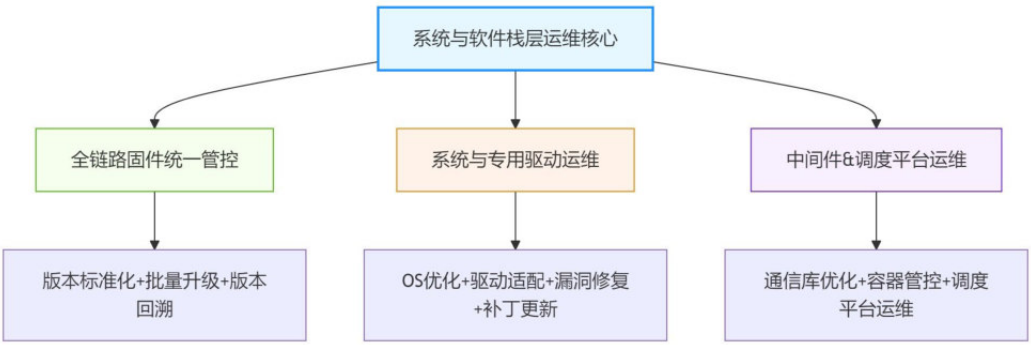


图181 超节点系统与软件栈层运维架构示意图

模块名称	核心组成部件	核心监控指标
全链路固件统一管理模块	BMC基板管理固件、BIOS系统固件、高速交换芯片固件、AI加速卡（GPU/NPU）固件、液冷/供电模块控制固件	版本一致性指标：固件版本一致性
系统与专用驱动运维模块	算力专属Linux优化系统、AI加速卡专用驱动、RDMA通信驱动、硬件外设驱动	系统稳定性指标：系统 uptime、进程存活率、漏洞率 版本一致性指标：驱动、系统参数一致性
中间件与调度平台运维模块	NCCL/HCCL集合通信库、Docker/K8s容器平台、全局资源调度平台、集中监控运维平台、虚拟化隔离层	软件效率指标：通信库延时、调度响应时长

业务与算力服务层运维

业务与算力运维是智算超节点运维价值的最终体现，它承接着底层硬件与系统软件栈的稳定状态，直接面向大模型训练、科学计算等高价值智算业务，构成了“资源就绪 -> 任务交付 -> 价值实现”的关键闭环。

与传统数据中心以设备监控和个体服务器可用性为中心的运维模式不同，超节点运维面临全域算力池化管理、长周期无中断业务保障、高并发细粒度资源调度等核心挑战。因此，其核心目标演进为：保障复杂业务流的稳定运行，实现全局算力资源的高效释放与最优利用，最终确保智算投资能够精准、无损耗地转化为业务成果。

为实现上述目标，超节点业务与算力运维依托智能化的运维管理平台，实现从自动化调度、智能预警到闭环处置的全流程覆盖。其工作范畴主要包括以下几个层面：

- **算力资源精细化运维**，对全域分布的 AI 加速卡（GPU/NPU）、高带宽内存（HBM）、CPU 及存储资源进行统一的池化管理。核心职责包括：资源的动态分配与回收、负载均衡调度、性能基线监控与劣化根因分析，确保算力供给的弹性与高质量。
- **核心业务全生命周期护航**，针对大模型预训练/微调/推理、高性能计算等核心作业，提供从任务编排、依赖部署、分布式启动、运行监控到断点续跑、异常恢复的全流程管理。重点解决任务排队、资源死锁、意外中断等影响业务连续性的问题。
- **跨域协同联动调控**，建立算力调度、业务流、高性能网络（Scale-Up/Scale-Out）之间的联动机制。通过业务优先级策略、资源抢占仲裁、变更协同（如运维操作与计算任务的错峰）等手段，化解资源竞争冲突，保障关键业务的性能与稳定性。
- **运维数据驱动优化**，基于对算力利用率、任务成功率、作业完成时间等核心指标的持续度量，构建资源瓶颈分析与调度策略仿真能力。通过数据反馈驱动资源调度算法的迭代、运维策略的优化，形成持续改进的智能运维闭环。

为精准评估上述运维职能的有效性，建立以下核心指标体系，用于实时监控、异常告警与趋势分析：

运维维度	核心指标	监测要点
资源健康度	算力在线率、算力利用率、算力负载偏差、冗余算力占比、HBM内存利用率	硬件可用状态、关键组件可靠性
资源效率	算力利用率(均/峰值)、负载均衡度、冗余资源占比	资源使用效率、分配合理性、池化效益
业务效能	任务提交成功率、作业完成率、断点续跑成功率、任务完成时间	业务流畅通度、连续性保障能力、性能达标情况
调度与协同效能	调度决策时长、资源抢占冲突率、跨域故障定位平均时间(MTTI)	调度超时、根因定位失败、自愈失败

3.3 超节点核心运维技术

结合超节点高密度、高耦合、高敏感的特性，以及四大层级运维需求，整套运维体系围绕“**感知-研判-执行-保障-复盘**”全流程，搭建四大核心技术支撑体系，四大技术相互联动、形成闭环，是破解超节点运维痛点、实现高效稳定运维的核心抓手。

全栈可观测技术

全栈可观测技术是超节点运维体系的**基础感知核心**，区别于传统数据中心零散式监控，针对超节点软硬件深度耦合、故障传导快、根因隐蔽的特性，打造“全域覆盖、全维度采集、秒级告警、根因溯源”的一体化可观测平台，打破硬件、网络、系统、业务各层级的数据壁垒，实现从单一部件到全局集群的全方位可视、可管、可溯，彻底解决传统运维“看不见、查不清、定位慢”的核心痛点。

1. 全域一体化可观测框架

采用“**数据采集层-数据归集层-分析决策层-可视化展示层**”四层标准架构，自上而下对接运维管控平台，向下穿透至物理硬件传感器与芯片级监控接口，实现秒级数据采集、实时分析、多级告警，架构逻辑清晰且适配各类超节点方案，具体架构如下：

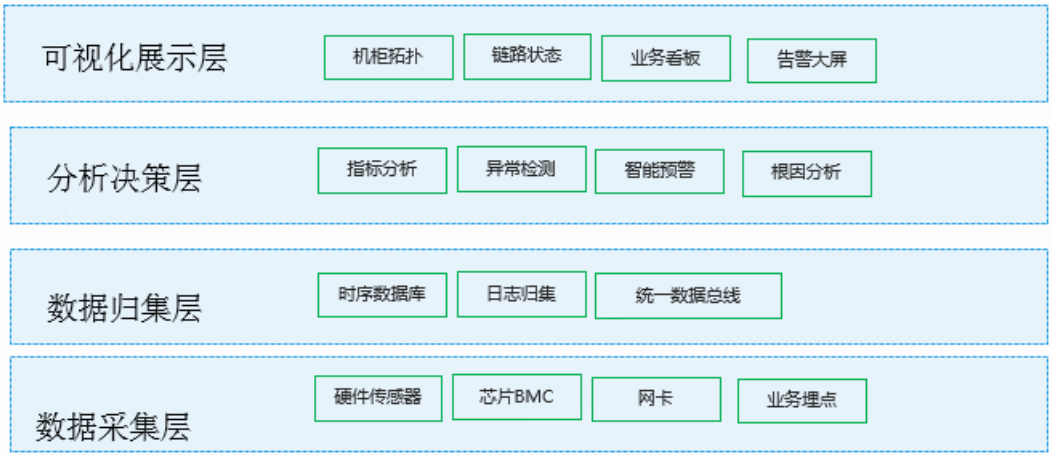


图182 超节点全栈可观测技术架构示意图

- **数据采集层：**作为全栈可观测的神经末梢，覆盖全场景采集入口，硬件层面通过 BMC、传感器直采温度、功耗、液冷流量、供电电压等参数；网络层面通过交换机、光模块、专用互联芯片采集带宽、时延、误码率、链路状态；系统层面采集固件版本、进程状态、资源占用、日志信息；业务层面通过埋点采集大模型训练进度、算力利用率、Loss 曲线、任务状态，采集频率最低达到秒级，关键核心指标支持毫秒级采集，杜绝数据滞后。
- **数据归集层：**采用专业时序数据库（InfluxDB/Prometheus）存储海量时序指标，统一日志归集平台汇总全链路日志，通过标准化数据总线完成数据清洗、格式统一、去重过滤，解决多源数据异构问题，为上层分析提供高质量数据支撑，支持海量指标长期存储与快速查询。
- **分析决策层：**核心算力大脑，内置阈值告警、趋势分析、异常检测算法，支持自定义指标基线，自动对比实时数据与行业标准值，识别隐性异常；联动智能化运维模块，实现告警自动分级、故障初步预判，避免无效告警与漏告警，提升告警精准度。
- **可视化展示层：**通过运维大屏、管控后台实现多维度可视化，包含物理机柜拓扑图、高速链路互联图、算力利用率热力图、业务运行看板、告警统计面板，直观呈现超节点全局运行状态，支持下钻查看单部件、单链路、单任务详情，大幅降低运维排查难度。

2. 核心能力与分级告警机制

结合超节点运维痛点与业界通用规范，将核心监控指标分为四大类，每类指标明确阈值范围，实现可量化、可考核，彻底告别模糊化运维，具体指标明细如下：

表23 超节点全栈可观测核心指标明细表

指标分类	核心监控指标	行业标准阈值	监控意义
硬件基础设施指标	算力芯片结温、液冷供回液温差、机柜进风温度、单柜功耗、供电电压波动、液冷流量、漏液传感器状态、硬件ECC报错数	芯片结温≤85℃，液冷温差≤5℃，电压波动≤±5%，漏液零告警	保障硬件稳定运行，提前预警硬件老化、过热、漏液等高危故障
高速互联网络指标	链路带宽利用率、端到端时延、链路误码率、丢包率、拓扑完整性、光模块功率	时延≤百纳秒级，误码率≤10^{-12}，丢包率=0，带宽利用率合理区间	保障高速通信畅通，杜绝网络瓶颈与链路故障引发算力雪崩
系统软件指标	固件版本一致性、CPU/内存占用率、驱动运行状态、进程存活状态、日志异常条数	版本一致性100%，内存占用≤85%，无异常报错日志	保障全栈软件兼容稳定，避免版本混乱、资源占用过高引发异常
业务算力指标	算力有效利用率、单卡算力输出、训练任务进度、Loss波动值、任务运行时长、断点保存状态	算力利用率≥90%，Loss无异常跳变，任务无无故中断	保障算力高效输出，支撑大模型长周期训练平稳运行

3. 多级告警机制与异常处置逻辑

全栈可观测平台配套三级分级告警机制，按照故障影响范围、严重程度划分为紧急告警（P1）、重要告警（P2）、一般告警（P3），匹配不同的通知方式与处置时效，避免告警泛滥导致运维人员忽视核心风险：

- **P1 紧急告警：**涵盖液冷漏液、整机断电、全网链路中断、大面积算力离线等高危场景，通过电话、短信、大屏声光同步告警，要求 15 分钟内响应，优先处置；
- **P2 重要告警：**涵盖单卡故障、局部链路误码超标、温度接近阈值、算力利用率骤降等场景，通过短信、运维平台弹窗告警，30 分钟内响应处置；
- **P3 一般告警：**涵盖日志异常、非核心部件状态波动、备件余量不足等场景，通过运维平台消息推送，24 小时内处置即可。

同时平台支持告警抑制、告警合并功能，避免单点故障引发批量重复告警，帮助运维人员快速定位根因，提升处置效率。

自动化与自愈运维技术

自动化与自愈运维技术是超节点全栈运维体系的**核心执行引擎**，也是承接全栈可观测技术感知结果、实现运维闭环落地的关键环节，更是破解超节点运维复杂度高、人工依赖度大、故障传导快、操作风险高的核心技术支撑。超节点集群规模动辄数百至数千张加速卡，软硬件组件数量庞大、耦合度极高，传统人工逐台巡检、单点调试、手动故障处置的粗放模式，不仅效率低下、耗时耗力，还极易因操作失误引发配置偏差、算力中断、任务崩溃等次生故障，完全无法适配超节点 7×24 小时高可用、秒级故障响应的刚性运维需求。

1. 整体架构

整体采用“三层感知+两层决策+一层执行”架构，自上而下对接业务调度层，向下穿透至硬件物理层，实现全栈数据打通、统一管控，适配各类国产与国际通用超节点方案，架构如下：

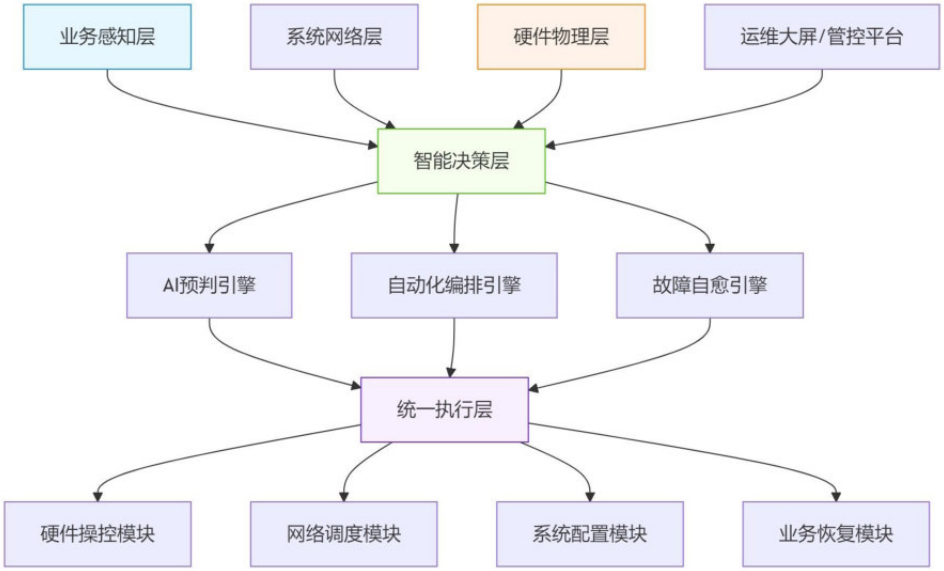


图183 超节点智能运维整体技术架构图

- **感知层：**实时采集硬件（温度、功耗、算力卡状态）、网络（带宽、误码、时延）、系统（固件版本、进程、日志）、业务（任务状态、算力利用率、训练 loss）全维度指标，采集频率达到秒级，覆盖超节点全部关键运维点位，无数据盲区；
- **决策层：**运维核心大脑模块，包含 AI 故障预测、自动化任务编排、故障自愈三大引擎，基于历史运维数据、行业基线标准、实时运行状态做出精准管控决策；
- **执行层：**承接决策层指令，完成硬件上下电、链路切换、配置刷新、任务重启等实操动作，全程闭环无需人工干预，实现秒级–分钟级快速响应。

2. 核心技术模块

- **自动化巡检与标准化运维能力**

  摒弃传统人工逐点排查模式，依托自动化编排工具，实现超节点全维度定时+定点+触发式巡检，覆盖日常运维全场景，是基础运维兜底能力，也是行业通用标配功能：

- **全栈自动化巡检：**支持按日/周/月定制精细化巡检策略，自动完成 AI 加速卡健康度校验、液冷/供电参数核查、高速互联链路误码检测、固件与驱动版本一致性比对、资源池空闲度扫描，生成标准化巡检报告，异常项自动标记分级告警，杜绝人工漏检、错检；
- **批量配置与版本标准化：**针对超节点全栈固件（BMC/BIOS/交换芯片）、驱动、系统参数、通信库（NCCL/HCCL），实现批量部署、统一升级、版本回溯，避免单节点版本偏差引发的全局算力雪崩，业界主流平台可实现数百卡集群批量配置耗时≤30 分钟；
- **日志与指标自动归集分析：**自动采集硬件日志、网络日志、系统日志、训练业务日志，统一归集清洗，过滤无效日志，提炼关键异常信息，替代人工逐台日志排查，大幅提升故障定位效率。

- **故障自愈技术**

针对超节点“单点故障影响全局”的特性，业界主流超节点运维平台均内置故障自愈模块，实现故障秒级感知、分钟级隔离恢复，避免大模型训练任务中断、重跑损耗，核心自愈场景与流程如下：

图184 超节点故障自愈全流程示意图

故障自愈场景

- **算力卡故障自愈：**检测到掉卡、ECC 报错、降频等异常，自动屏蔽故障卡，同步调度空闲算力资源补位，无需人工更换即可恢复算力池容量，硬件更换完成后自动重新纳管；
- **高速网络链路自愈：**监测到 Mesh 直连链路或 IB 网络误码超标、链路中断，自动切换备用链路，重构网络拓扑，保障通信带宽与时延达标，杜绝网络瓶颈导致训练停滞；
- **液冷与供电自愈：**温度、流量、电压异常时，自动调节散热功率、切换冗余供电模块，触发预警式保护，避免硬件损坏与业务中断，实现液冷漏液、供电异常等高危场景的快速应急响应。
- **软件类故障自愈：**针对进程假死、容器崩溃、驱动加载异常、固件版本偏差、调度服务卡顿等软件故障，自动执行进程重启、容器重建、驱动重新加载、配置一键回溯等操作，快速恢复软件正常运行，避免软件故障引发算力衰减；
- **业务类故障自愈：**针对任务静默失败、算力利用率骤降、断点保存异常等业务异常，自动暂停任务并保存断点数据，排查底层软硬件故障，修复完成后自动触发断点续训，保障长周期训练任务不中断、成果不丢失。

故障隔离机制

故障隔离的目标是防止故障扩散，将故障组件与正常组件隔离，保障超节点整体功能的可用性：

- **硬件故障隔离：**通过硬件管理接口（IPMI/Redfish）远程关闭故障硬件组件（如故障硬盘、故障网卡），或通过服务器热插拔功能移除故障组件；对于分布式计算节点，通过集群管理工具（如 Kubernetes、Slurm）将故障节点移出集群，停止向其分配任务；
- **网络故障隔离：**通过 隔离故障网络端口、隔离故障链路，或调整路由表将流量切换至备用链路；对于交换机故障，启动冗余交换机接管流量；

- **软件故障隔离：**通过容器化技术（Docker、Kubernetes）隔离故障应用，重启故障容器；对于中间件故障，关闭故障服务实例，启动备用实例；

故障恢复策略

故障恢复策略根据故障类型与严重程度，采用“自动恢复 + 手动恢复”相结合的方式，优先保障核心业务的连续性：

- **自动恢复：**针对轻微故障（如进程崩溃、网络连接中断），通过监控工具触发自动恢复脚本，如重启进程、重新建立网络连接、切换至备用存储介质；对于分布式集群，通过集群自愈机制（如 Kubernetes 的自愈功能、Ceph 的副本修复）自动恢复故障节点 / 数据；
- **手动恢复：**针对严重故障（如核心硬件损坏、数据丢失），由运维人员执行手动恢复操作，包括更换故障硬件、从备份中恢复数据、重新配置系统 / 网络；
- **恢复验证：**故障恢复后，通过性能测试、功能验证工具确认超节点运行状态正常，如检查硬件指标是否在正常范围、网络连通性是否恢复、应用服务是否正常提供服务。

故障分级

故障分级	定义	在运行任务
硬件不可自愈	不在位；硬件功能不可用，如GPU掉卡，设备掉电；反复重启等状态反复变化	任务主动或被动中断。故障件替换后手工恢复；或重新调度资源，重启任务
软件不可自愈	软件应用退出、进程终止、任务挂死等	任务主动或被动中断。手工恢复任务或通过管理软件自动恢复任务
硬件降级	硬件功能可用，但是故障造成性能降低，比如Lane故障	任务不主动中断。计划维护，故障件替换后，手工恢复
软件可自愈	通过软件重试可以恢复的故障，可能是硬件引起但表现在软件	软件可靠性保护，单次故障对任务无影响或短暂抖动，连续多次故障有时会带来系统性能降价
硬件可自愈	硬件能够自动恢复的短时（<秒级）故障，且可以通过软硬件可靠性设计，进行自愈。如端口闪断、网络少量丢包等	硬件自行恢复，软件可靠性保护，对任务无影响或短暂抖动

- **AI 驱动的智能预测与预防性运维**

依托机器学习、大数据分析技术，基于超节点长期运行的历史指标数据，构建故障预测模型，实现潜在风险提前预判，是头部智算中心与超节点厂商主推的进阶能力，核心应用场景：

- **硬件寿命与故障预测：**针对 AI 加速卡、电源模块、硬盘、光模块、液冷泵等核心部件，分析温度波动、功耗异常、运行时长等指标，提前预判硬件老化、潜在故障，推送预防性更换建议，整体故障率降低可达 60%以上；
- **算力性能衰减预警：**实时监测单卡算力、集群整体利用率、通信效率，预判性能衰减诱因（散热不佳、链路拥堵、参数异常），自动推送调优方案，保障算力有效利用率稳定在 90%以上；

- **业务风险预判：**结合大模型训练任务的 loss 波动、算力占用、通信开销，预判任务异常中断风险，提前触发断点保存，避免训练成果丢失。

- **数字孪生可视化运维**

  采用数字孪生技术，1:1 还原超节点物理机柜、硬件布局、链路拓扑、液冷管路、供电走向，实现物理世界与数字世界同步映射，运维人员可通过管控大屏直观查看全节点运行状态，精准定位故障物理位置，替代传统纯数据监控模式，大幅降低运维操作难度，是当前高端超节点运维的标配可视化能力。

3. 性能指标要求

- **效率提升：**日常运维人工工作量降低 80%，故障处置时长缩短 70%以上。
- **稳定性保障：**计划外故障停机时间减少 90%，大模型任务异常重启率≤1%。
- **力保障：**算力有效利用率稳定维持≥90%，避免资源闲置与性能损耗。

全局一致性管控技术

全局一致性管控技术是超节点运维体系的**刚性合规中枢**，也是破解超节点软硬件深度耦合场景下，版本错乱、参数漂移、配置差异等核心痛点的关键支撑。超节点内部数千颗算力芯片、上百套网络设备、全栈系统软件与固件高度绑定，哪怕单个节点的固件版本、驱动参数、网络配置出现微小偏差，都会引发软硬件不兼容、多芯片通信异常、算力利用率骤降，甚至诱发全集群算力雪崩，这类隐性故障隐蔽性极强、排查难度极大，传统分散式、人工校验的管控模式完全无法适配。

本技术核心聚焦“**全栈统一、基线管控、偏差回溯、全程合规**”四大核心目标，搭建覆盖固件、驱动、系统、配置、参数的全维度一致性管控体系，将所有运维对象纳入统一标准基线，实现从上线预装、日常运行到变更迭代的全生命周期闭环管控，彻底杜绝各类偏差隐患，保障全集群“万卡如一机、全栈同一规”，和全栈可观测、自动化自愈技术深度联动，形成“偏差感知-自动校验-批量回溯-合规闭环”的完整管控流程，全程不重复前文系统软件层的基础运维内容，聚焦专属管控架构、核心机制、落地流程与量化指标。

1. 核心管控范围与基线标准

全局一致性管控覆盖超节点全栈软硬件，针对不同层级管控对象制定专属标准基线，所有基线均参照厂商原厂规范、信通院行业标准与智算中心实操要求，严禁私自偏离，核心管控范围分为四大模块，每个模块均设定刚性合规要求：

- **固件版本一致性：**作为最核心管控模块，涵盖 BMC 基板管理固件、BIOS 固件、算力芯片固件、Mesh 交换芯片、IB 交换机固件、液冷/供电控制固件等全链路硬件固件，要求同类型、同批次硬件固件版本 100%统一，严禁跨节点版本混用，所有固件必须采用厂商适配超节点集群的专属正版版本，禁止通用版本替代；
- **驱动与系统一致性：**涵盖算力加速卡专用驱动、RDMA 通信驱动、硬件外设驱动、定制化操作系统内核参数、系统服务配置，要求全集群驱动版本与固件、硬件深度适配，系统内核参数、后台服务启停状态、安全配置完全统一，驱动加载成功率 100%，无版本兼容冲突；
- **网络配置一致性：**涵盖 Mesh 直连网络路由参数、IB 交换机交换策略、光模块参数、带宽分配规则、QoS 配置、防火墙策略，全集群网络配置无单点偏差，确保通信时延、误码率指标全程达标，杜绝因配置差异导致的链路瓶颈；

- **业务与调度配置一致性：**涵盖全局资源调度规则、容器隔离参数、断点保存策略、业务优先级配置、算力分配规则，保障多任务、多租户场景下调度逻辑统一，避免资源抢占、任务冲突，维持算力输出稳定。

2. 核心技术架构与管控流程

采用“**基线定义层-全域校验层-偏差处置层-合规审计层**”四层闭环架构，联动自动化运维平台与全栈可观测平台，实现一致性管控全流程自动化，无需人工逐节点核查，彻底解放运维人力，具体架构与流程拆解如下：

- **基线定义层：**搭建集群级标准基线库，由厂商技术专家与运维团队联合制定，固化全栈最优版本、标准参数与合规配置，形成唯一基准；基线库支持版本管理，更新前必须经过小范围试点测试，验证无兼容问题后再全量推行，同时留存历史基线，支持一键回溯；
- **全域校验层：**依托自动化巡检引擎，执行 7×24 小时不间断全量一致性校验，分为实时后台校验、每日定时全量校验、变更后专项校验三类模式，对比各节点实际配置与标准基线差异，覆盖版本号、参数值、配置项、运行状态全维度，校验覆盖率 100%，偏差识别准确率≥99.5%；
- **偏差处置层：**针对校验发现的偏差项，按照偏差等级分级处置，轻微参数漂移由平台自动批量回溯至标准基线；中度版本偏差触发自动化批量升级/回退；重大合规偏差暂停相关节点算力输出，锁定变更权限，推送告警至运维团队人工复核，处置全程联动自动化执行平台，批量同步操作，无单点遗漏；
- **合规审计层：**全程记录基线变更、校验结果、偏差处置全流程日志，留存周期不低于 1 年，形成一致性管控审计报表，标注偏差节点、偏差类型、处置结果、操作人，满足智算中心合规审计要求，同时通过偏差数据复盘，优化基线标准与巡检策略，减少后续偏差发生。

3. 性能指标要求

- 全栈固件、驱动、配置一致性达标率：100%
- 一致性偏差识别时长：≤1 分钟，偏差识别准确率≥99.5%
- 轻微偏差自动回溯成功率：≥99%，回溯时长≤5 分钟
- 256 卡集群全量一致性校验时长：≤8 分钟
- 变更操作合规率：100%，全程审计可追溯

高可靠灾备与业务连续性技术

高可靠灾备与业务连续性技术是超节点运维体系的最后兜底防线，核心应对极端故障、重大灾害、计划外停机等突发场景，保障核心算力业务不中断、关键数据不丢失、长周期训练成果不损耗，彻底解决超节点业务中断成本高、恢复难度大的行业痛点。超节点核心承载大模型长周期训练、高端科学计算等关键业务，单次中断可能造成数天甚至数周的算力与时间损耗，单纯依靠故障自愈无法覆盖极端场景，必须搭建多层级冗余、多维度备份、快速切换恢复的灾备与连续性保障体系，全程贴合超节点业务特性，区别于传统数据中心基础灾备模式，聚焦“冗余前置、备份无感、切换快速、业务连续”核心目标。

1. 核心技术逻辑与分层冗余架构

遵循“**多层冗余、分级备份、就近恢复、业务无感**”核心逻辑，针对超节点硬件、网络、数据、业务四大维度，搭建四级冗余保障架构，逐层提升可靠性，杜绝单点故障引发的全局性瘫痪，具体架构拆解如下：

- **一级冗余：**硬件级冗余：核心算力、供电、散热模块采用 2N 及以上冗余配置，算力芯片预留冗余算力单元，供电采用双路独立市电+UPS 不间断电源+备用发电机三重冗余，液冷系统配备备用泵组与管路，单个硬件模块故障后，冗余模块秒级接管，无算力中断感知，硬件可用性≥99.99%；
- **二级冗余：**网络级冗余：Mesh 直连网络、IB 高速网络均配备备用链路与冗余交换节点，核心链路采用主备双链路设计，主链路故障后，30 秒内自动切换至备用链路，网络零丢包、业务无感知，网络可用性≥99.99%；
- **三级冗余：**数据与配置备份：建立全量备份+增量备份+实时快照三重备份机制，核心训练数据、模型权重、系统配置、基线参数定时全量备份，日常变更增量备份，关键节点实时快照，备份数据异地存储，防止本地灾害导致数据丢失，备份数据恢复成功率 100%；
- **四级冗余：**业务级连续性保障：针对长周期大模型训练业务，部署全局断点续训机制，定时自动保存训练断点与进度数据，业务中断后，无论软硬件故障是否修复，均可依托断点数据快速接续任务，无需从头重跑，保障业务连续性。

2. 核心灾备与连续性保障能力

- **全维度数据备份与恢复机制**

  建立差异化备份策略，兼顾备份效率与数据安全性，避免冗余备份占用过多算力资源：

  - **系统与配置备份：**全栈标准基线、系统配置、网络参数每周全量备份，每日增量备份，配置变更后实时快照备份，系统崩溃或配置错乱时，支持一键批量恢复至备份状态，恢复时长≤30 分钟；
  - **业务数据备份：**大模型训练权重、中间结果、科学计算数据实行实时增量备份，关键进度节点自动全量备份，备份数据加密存储，异地灾备机房同步留存，数据丢失后可快速恢复，恢复成功率 100%；
  - **备份有效性校验：**每月自动执行备份数据有效性校验，核查备份文件完整性、可恢复性，淘汰失效备份，更新最新备份数据，确保突发场景下备份数据可用。

- **断点续训与业务快速恢复**

  针对超节点核心长周期业务，打造专属断点续训体系，这是业务连续性保障的核心能力：

  - **自动断点保存：**依托业务运维平台，按照预设周期（通常为每小时或每完成一个训练批次）自动保存训练断点、进度数据、模型状态，保存过程无感，不占用核心算力资源，不影响训练效率；
  - **故障后无缝接续：**无论软硬件故障、断电还是机房切换，故障修复后，平台自动读取最新断点数据，快速加载模型与进度，无需人工干预，直接接续任务运行，断点续训成功率 100%；
  - **跨节点任务迁移：**针对局部硬件损毁无法快速修复的场景，支持将任务断点数据迁移至冗余算力单元，自动完成环境适配与配置同步，快速重启任务，最大限度压缩业务中断时长。

3. 性能指标要求

- 超节点整体可用性：≥99.9%，年计划外停机时间≤8.76 小时
- 断点续训成功率：100%，业务接续时长≤10 分钟
- 数据备份恢复成功率：100%，备份数据有效性≥99.9%
- 硬件/网络冗余切换时长：≤30 秒，业务无感知
- 跨机房灾备切换时长：≤2 小时，核心业务优先恢复

全链路能效管理与绿色运维技术

超节点单柜功耗高达 30-100kW，属于数据中心高功耗核心载体，传统固定功耗、粗放散热的运行模式极易造成算力空载损耗、能耗冗余、PUE 超标等问题，既增加智算中心运营成本，也不符合绿色智算、低碳数据中心的行业发展导向。结合高密度算力集群特性、液冷散热专属架构及国家双碳战略要求，超节点能效管理需构建**“全维度监控-智能动态调节-量化评估-闭环优化”**的全链路绿色运维体系，打破硬件、散热、供电、业务调度的能效割裂管控模式，实现算力输出与能耗消耗的精准匹配，在保障算力高可用、高性能的前提下，最大限度降低综合能耗、优化 PUE 值、提升能源利用效率，打造高效低碳的智算算力运维新模式。本章节从功耗监控体系、动态智能调节、能效评估优化、绿色节能落地细则四大维度，搭建完整的能效运维技术框架，配套行业量化指标与实操规范，兼顾技术可行性与落地实用性。

1. 全维度功耗监控体系

能效管理的前提是精准感知，超节点高密度、高耦合的特性决定了功耗监控必须突破传统机柜级、机房级粗放监控模式，搭建**芯片级-模块级-机柜级-集群级**四级联动功耗监控体系，实现全链路功耗数据可采集、可溯源、可分析，杜绝能耗盲区，为后续动态调节与优化提供数据支撑。

- **芯片级精细化监控：**针对 AI 加速芯片（GPU/NPU）、CPU、交换芯片等核心算力部件，通过 BMC 接口、芯片内置功耗传感器，实时采集单芯片瞬时功耗、平均功耗、功耗峰值、idle 空载功耗，精准定位单卡功耗异常、空载耗电、超频冗余损耗等问题，监控粒度细化至秒级，同步关联芯片负载率、结温数据，分析功耗与性能的对应关系。
- **模块级分区监控：**分模块采集液冷散热模块、供配电模块、高速互联模块、管理节点模块的独立功耗，区分算力有效功耗与辅助系统损耗，重点监控液冷循环泵组、温控风机、配电转换损耗等辅助能耗占比，定位辅助系统能效短板。
- **机柜级与集群级汇总监控：**单机柜部署智能精密配电监测单元，实时采集机柜总输入功耗、功率因数、电压电流谐波；集群层面搭建全局功耗监控大屏，汇总多机柜、全超节点集群总功耗，统计单位算力功耗（PFlops/W）、日均总能耗、峰谷能耗差值，同步对接机房动环系统，实现跨层级功耗数据联动分析。
- **功耗数据闭环归集：**所有层级功耗数据实时上传至全栈可观测平台，建立能耗专项数据库，留存周期不少于 1 年，支持按小时、日、周、月生成能耗报表，自动标记功耗突增、空载超时、能效超标等异常项，触发分级能效告警。

2. 智能动态功耗调节技术

依托四级功耗监控数据，联动硬件特性、业务负载与散热供电系统，推行算力负载联动、散热协同适配、供电智能优化三大动态调节策略，摒弃固定功耗、满负荷运行的传统模式，实现“**算力负载高则功耗拉满、负载低则功耗下调、空载则深度节能**”的精准匹配，杜绝无效能耗浪费，同时保障业务性能无衰减。

- **算力芯片智能降载与调频调压：**基于业务负载实时动态调节 AI 加速芯片功耗，大模型训练、高性能计算等高负载场景，维持芯片额定功耗满负荷输出，保障算力性能；推理任务、离线预处理等低负载场景，自动下调芯片核心频率与工作电压，关闭闲置算力核心，降低空载功耗；无业务运行时段，启动芯片深度节能模式，将功耗降至待机基线，业务唤醒时延控制在秒级，不影响后续任务调度。
- **液冷散热与功耗协同调节：**打破散热系统固定转速、固定流量的运行模式，根据芯片功耗、结温实时联动调节液冷供液流量、泵组转速与温控功率，芯片高功耗、高结温时提升散热功率，低负载、低结温时降低散热能耗，实现“按需散热”，大幅减少液冷系统冗余能耗；同时优化供回液温差控制，将温差稳定在 4–5℃行业最优区间，提升散热换热效率。
- **供配电系统动态优化：**针对双路冗余供电架构，根据集群总功耗动态调节供电回路负载均衡，避免单回路过载损耗；利用机房峰谷电价差异，结合业务优先级推行错峰算力调度，低谷电价时段优先运行高功耗、高算力消耗任务，高峰电价时段以低功耗运维、轻负载任务为主，降低用电成本；优化功率因数补偿，将功率因数稳定在 0.95 以上，减少配电线路无功损耗。
- **算力硬件智能休眠与唤醒：**针对业务低谷期、任务间隙的闲置算力节点，通过自动化平台执行分级休眠策略，关停闲置加速卡、非必要管理模块，保留核心监控与唤醒功能；业务高峰期或新任务上线时，秒级唤醒节点，快速恢复算力输出，将空载功耗占比严格控制在 5%以内。
- **业务与算力错峰调度：**结合峰谷电价与算力负载波动，合理排布大模型训练、科学计算、推理任务的运行时段，核心高负载任务优先安排在谷电时段，常规推理任务错峰运行；同时优化任务调度逻辑，均衡全集群算力负载，避免局部机柜高负载满功耗运行、局部机柜空载闲置的两极分化，提升整体能效。
- **全栈软件能效调优：**轻量化操作系统部署，关闭无关后台服务与端口，减少系统冗余功耗；优化固件与驱动参数，提升算力硬件能效比；定期清理冗余进程与日志，避免系统后台占用资源导致额外能耗，从软件层面降低算力转化损耗。

3. 核心能效监控指标

依托全栈可观测平台，搭建专属能效监控模块，实现 7×24 小时不间断能耗数据采集与分析，核心监控指标全部量化，具体指标与阈值如下：

表24 超节点核心能效评估指标表

指标名称	计算公式	行业标准阈值	管控意义
数据中心能源效率（PUE）	总设备能耗/IT设备能耗	≤1.1	衡量整体散热、供电能效水平
IT设备能源效率（ITUE）	IT设备能耗/总算力输出	≤1.05	分区管控，消除算力能耗短板
单位算力功耗	总功耗/总算力输出（PFlops）	≤25W/PFlops	提升算力有效利用率，降低单位能耗

指标名称	计算公式	行业标准阈值	管控意义
空载功耗占比	空载能耗/总能耗	≤5%	杜绝长期空载、低负载运行浪费
功率因数	有功功率/视在功率	≥0.95	减少无功损耗，提升供电能效

3.4 超节点运维软件实践案-AD-DC 智算版超节点运维

AD-DC 智算版是专为智算场景打造的专属运维管控平台，针对超节点的运维摒弃通用服务器监控软件的粗放式架构，深度适配 Scale-Up 全互联架构、高密度液冷供电、大模型长周期业务场景，核心聚焦超节点自动化部署、全域纳管、Scale-Up 专属网络监控、智能故障根因分析、故障自愈五大核心能力，联动底层 RMC/BMC、专用交换芯片与算力硬件，实现全流程自动化、智能化、无扰动运维，并且通过与上层智算作业调度平台深度联动，构建“故障感知-根因定位-资源隔离-任务自愈-进度接续”全闭环自愈体系，实现断训无感知、任务不中断、进度零丢失，彻底解决长周期业务运维兜底难题。

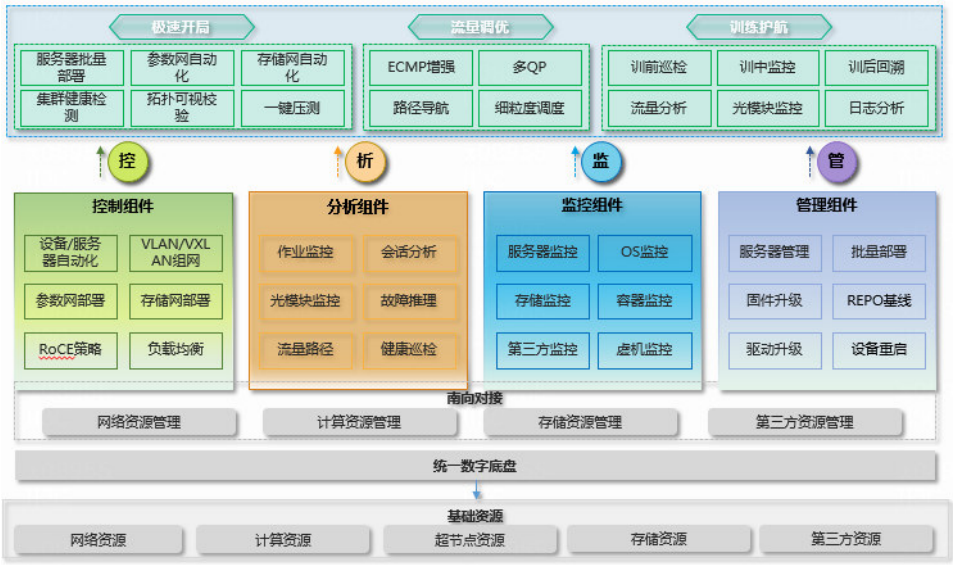


图185 AD-DC 智算版软件架构

通过对超节点的纳管，实现超节点 Scale-up 网络的拓扑可视化能力。

图186 超节点 Scale-up 拓扑运维

通过训前巡检能力，可以有效降低训练任务因环境问题引发的故障，提升训练效率。支持普通与深度巡检，针对超节点支持 Scaleup 网络单独巡检，支持 ScaleUp 叠加 ScaleOut 网络整体巡检。

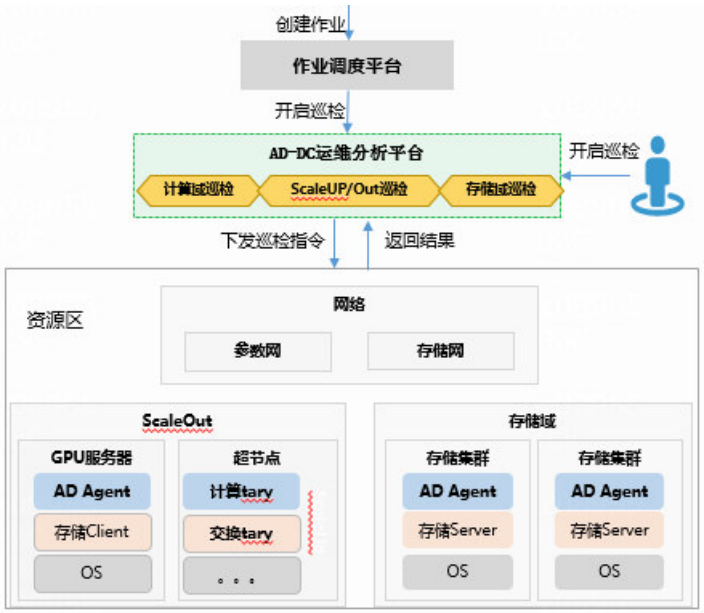
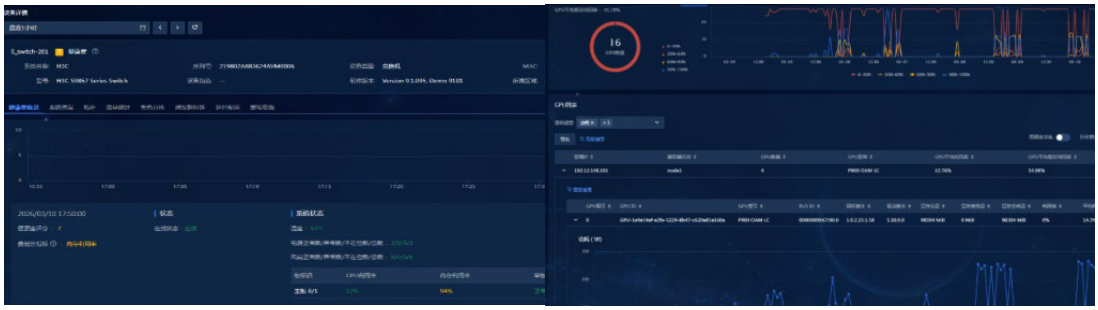


图187 训前巡检架构

图188 训前巡检指标

训中作业监控和故障分析，包含交换机、服务器、存储集群的端到端的整网全栈数据采集、调优能力及拥塞分析等能力，保障训练任务的稳定运行。



监控指标：

监控模块	监控指标
交换机	电源、风扇、温度、CPU、内存等
网络	收发包、错报，pool缓存及水线，headroom缓存及水线，队列收发包、拥塞丢包、缓存及水线
GPU	型号、功耗、温度、显存、利用率、ECC、xpulink状态
服务器	系统版本、驱动版本、固件版本、CPU、内存、网卡收发包等指标

3.5 超节点运维未来发展趋势

全域 AI 自治运维成为核心方向

未来运维的核心将从“工具辅助”转向“AI 主导”。通过构建超节点专用的运维大模型，融合实时数据与历史知识，实现从异常检测、根因分析、策略生成到执行反馈的完整自治闭环。其高级阶段将体现为：

- 预测性自愈：AI 提前预判硬件劣化、网络拥塞，在业务受影响前主动调度或隔离。
- 动态性能调优：根据业务特征实时调整网络参数、作业调度策略，全局优化资源利用率。
- 知识自我进化：运维模型能从每次处置中学习，不断迭代优化决策逻辑。

全链路数字孪生运维落地普及

全链路数字孪生将成为运维智能体的“试验场”和“指挥沙盘”。1:1 高保真模型不仅实现状态同步，更关键的是支持：

- 无损演练：在孪生体中模拟故障注入、扩容方案，验证处置策略，实现“先仿真，后实操”。
- 因果追溯：故障发生时，在孪生体中进行快速回放与根因推演，极大缩短定位时间。
- 容量规划：基于业务预测，在孪生体中进行负载推演，指导物理集群的精准扩容。

绿色低碳与高效运维深度融合

后续行业将出台更细化的超节点运维国家标准、技术规范，统一运维指标、操作流程、SLA 体系，打破厂商壁垒；同时形成“厂商+智算中心+第三方运维”协同生态，共享运维技术、案例、备件资源，降低行业整体运维成本，推动超节点运维规模化、规范化发展。

3.6 总结与展望

本白皮书系统构建了面向超节点的下一代运维范式。这一范式以“**全栈可观测**”为数据基石，以“**智能闭环**”为核心引擎，以 “**全局协同**”为关键架构，旨在将复杂耦合的超节点基础设施，转化为稳定、高效、透明的确定性算力服务。

展望未来，运维体系本身将进化为超节点的 “**AI 赋能的价值中枢**” 。通过内核智能化、对象数字化与目标价值化的深度融合，运维不仅是大模型训练与科学计算的无损保障，更将成为优化算力成本、践行绿色责任、驱动智算业务持续创新的核心支撑力量，护航智算产业迈向高质量发展新阶段。

新华三超节点介绍

1 概述

1.1 产品简介

H3C UniPoD S80000 液冷机柜（以下简称 S80000 或液冷整机柜）是 H3C 针对通信开销大的 AI 场景打造的 AI 整机柜，具有高密度、整机柜交付、液冷散热等特点，并深度集成了覆盖 AI 业务全生命周期的超节点软件栈，是万亿级 MoE 模型训练与推理、以及下一代智能体（Agent）应用的理想基础设施。

S80000 通过硬件层面的柜内卡间全互联与高速网络拓展，构建了极致性能的算力基座；同时，其内置的智能软件栈实现了对海量异构算力的统一抽象、智能调度与高效运维。这种软硬协同设计，旨在系统性解决大规模智算中心在资源管理、性能调优与运维保障方面的核心挑战，为用户提供一站式、高性能、高可用的 AI 算力服务。

液冷系统分别有 32 卡、64 卡和 256 卡，外观分别见以下图示。

图189 S80000 液冷机柜 32 卡外观示意图

图190 S80000 液冷机柜 64 卡外观示意图

图191 S80000 液冷机柜 256 卡外观示意图

1.2 产品特点

极致性能

新华三 S80000 AI 超节点采用整机柜设计，单柜可部署 32 张或 64 张高性能 AI 加速卡，并支持柜内卡间全互联通信，卡间通信带宽相比传统 8 卡 OAM 服务器提升高达 4 倍。在此基础上，节点支持拓展至 256 卡全互联，构建大规模算力集群。单柜形成的算力系统计算效率远超传统形态 8 卡服务器。针对 MoE 模型，单卡推理性能可提升 100%以上，每 token 成本降低至少 50%。同时，系统支持基于负载感知的 PD 实例数动态调度，进一步将推理效率提升 40%以上。

超高可靠

采用 Cable tray 互联方式，可靠性较传统方案提高 10 倍，硬件层面通过关键器件级冗余设计与生产期极限压测，大幅增强设备韧性；软件层面支持 PD 实例的自动切换，全面提升系统级 RAS 能力，确保业务连续性与稳定性。

极简运维

支持水、电、网三总线盲插设计，极大简化现场部署与维护流程。整柜提供算、存、scale up/out 双网的统一运维界面，实现资源部署、状态监控与故障处理的集中化管理。系统具备故障自动感知、隔离及恢复能力，结合工厂预集成与整柜运输交付模式，显著降低部署周期与运维复杂度。

高效散热

采用先进液冷散热方案，冷板覆盖 CPU、GPU 等核心热源，在保障极致算力密度的同时实现高效、静音的热管理。

2 产品规格-64/256卡整机柜

S80000 液冷整机柜（64 卡和 256 卡）产品规格和技术参数。

2.1 规格参数

产品规格的计算，以产品支持的所有部件为基准。

表25 液冷整机柜产品规格

功能特性	说明
机柜空间分布（46SU）	可用空间46SU： 1SU～2SU用于安装假面板 3SU～4SU、39SU～40SU用于安装Power Shelf，其中安装电源模块及RMC模 6SU~7SU、10SU~13SU、16SU~17SU、26SU~27SU、30SU~33SU、36SU~37SU用于安装计算节点 8SU~9SU、14SU~15SU、28SU~29SU、34SU~35SU用于安装交换节点 41SU~46SU用于安装TOR交换机（3SU）和理线架（3SU） 18SU~19SU、24SU~25SU用于2U毛刷+理线架
机柜单元	机柜宽600mmx深1200mmx高2300mm(可用槽位46SU) 冷板液冷为主，风冷辅助，支持漏液检测 Busbar集中供电方案，最大支持120kw供电能力 最大支持16个计算节点、8个交换节点
计算节点	最多可支持16台S8400 G7计算节点（具体规格请参见S8400 G7技术白皮书）
交换节点	最多可支持8个交换节点
CDU	支持风液式CDU
冷却液	去离子水、PG25丙二醇水溶液、EG25乙二醇水溶液
冷板	CPU冷板采用钎焊工艺，密封可靠性高
Manifold	Manifold设计顶部具备自动排气阀，泄漏液体可通过软管导入排液管、喷溅盒 节点鹰嘴泄漏液体，通过排液槽一同汇入机柜底部积液盘
快接头	快接头为计算节点中液冷模块与分水管（Manifold）的连接头。S8400 G7计算节点选用的快接头，型号为UQDB06，其他型号快接头可根据需求适配
漏液检测	冷板模组集成漏液检测绳，通过服务器HDM管理漏液信号 Manifold连接漏液检测绳
管路设计	软管方案，方便布局
高冗余	单个Power Shelf的PSU支持11+1冗余
供电功能	Busbar集中供电方案，最大支持120kW供电能力： 计算节点采用Busbar供电 每计算节点单槽位支持不低于8kW，每交换节点槽位支持不低于3kW 支持机柜内管理交换机AC 220V输入
管理功能	机柜级RMC管理模块，实现对整机和电源的状态监控、日志记录和故障报警，节点的资产管理等功能
一体化交付	支持整柜运输，一体化交付，即插即用

2.2 技术参数

表26 环境参数

类别	项目	说明
物理参数	尺寸（高x宽x深）	2300mm*600mm*1200mm（含脚轮）
	重量	• 空机柜重量：350kg • 64 卡满配最大重量 1400kg（含包材 150kg）
环境参数	温度	工作环境温度：5°C～40°C
		贮存环境温度：-40°C～70°C
	湿度	• 工作环境湿度：8%～90%（无冷凝） • 贮存环境湿度：5%～95%（无冷凝）
	海拔高度	• 工作环境高度：-60 m～3000m（海拔高于 1500m 时，每升高 100m，规格最高温度降低 0.33°C） • 贮存环境高度：-60 m～3000m
	运输坡度	≤10 度，建议 6 度

2.3 散热规格参数

表27 液冷系统散热规格表

节点名称	整柜进液温度℃	整柜流量 L/min	机柜工作压力 Bar	整柜压差 Kpa
S8400 G7	5~40	• CPU+GPU 液冷： • 64 卡，16 计算节点，整柜总流量≥96	≤3.0（推荐≤2.5）	≥60
		• CPU+GPU+DIMM 液冷： • 64 卡，16 计算节点，整柜总流量≥112		

3 超节点软件栈

3.1 概述

S80000 超节点软件栈是一个面向 AI 业务全生命周期的综合性管理平台，旨在实现对大规模智算资源的统一纳管、智能调度与高效运维。该软件栈以“高易用、高可靠、高性能”为核心设计理念，通过深度软硬件协同优化，为模型训练、微调及推理等核心场景提供从底层基础设施到上层业务应用的全栈支持。

3.2 软件栈分层架构

超节点软件栈采用层次化、平台化的设计思想，整体架构自下而上可分为基础资源层、加速计算与通信层、框架与平台层以及智能管控与运维层，如图所示。各层之间通过标准接口协同工作，共同构成一个完整、自治的智算操作系统。

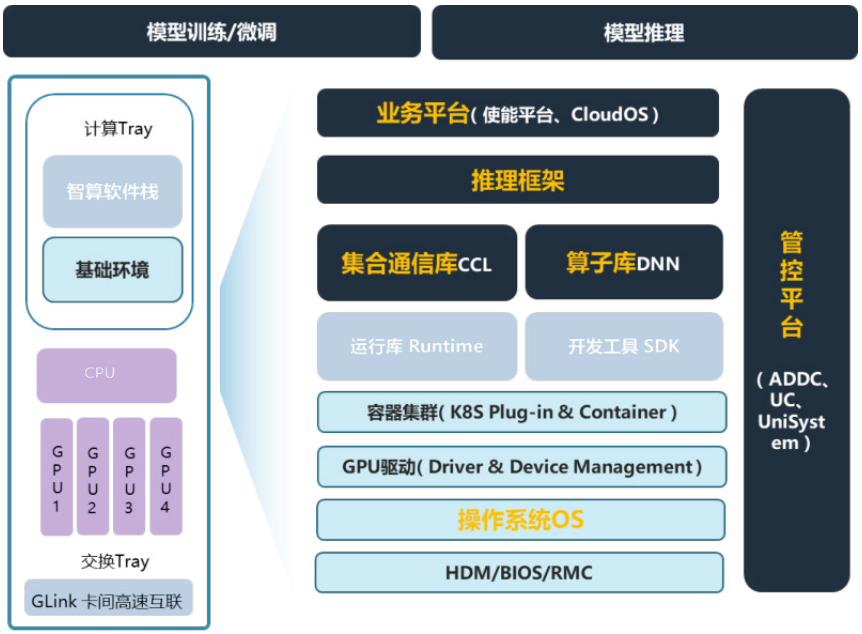


图192 分层架构

1. 基础资源层

作为软件栈的基石，本层负责对物理硬件进行抽象与管理，确保计算资源的稳定供给与高效利用。

- **操作系统：**采用针对智算场景深度定制的操作系统（如 NingOS），提供 NUMA 亲和调度、内存 Pin 驻留等特性，减少跨节点访问延迟，加速模型加载与执行。
- **硬件驱动与管理：**集成优化的 GPU 驱动、设备管理模块以及 HDM/BIOS/RMC，实现对计算 Tray、交换 Tray 及高速互联硬件的精细控制。
- **容器化与编排基础：**基于 Kubernetes 及其插件，构建稳定、高效的容器集群环境，为上层应用提供灵活、隔离的资源运行单元。

2. 加速计算与通信层

本层聚焦于释放硬件算力潜能，为大规模分布式 AI 任务提供核心的计算、通信与数据访问能力。

- **算子库（DNN）：**提供针对特定硬件架构（尤其是国产 GPU）极致优化的基础计算内核。通过多粒度并行、矩阵乘加速、算子融合与数据布局优化等技术，显著提升模型核心运算（如 Attention、MLP）的并发性能与效率。

- **集合通信库（CCL）：**专为超节点拓扑优化，支持卡间高速互联总线。具备拓扑自动探测、最优通信路径选择（如 Ring、Mesh 算法）能力，并集成 Primary-Backup QP、链路故障自动切换等高可靠机制，有效降低集群通信延迟与故障率。
- **运行时与开发工具：**包含高效运行库（Runtime）及配套 SDK，为上层框架和应用提供稳定的执行环境与便捷的开发接口。

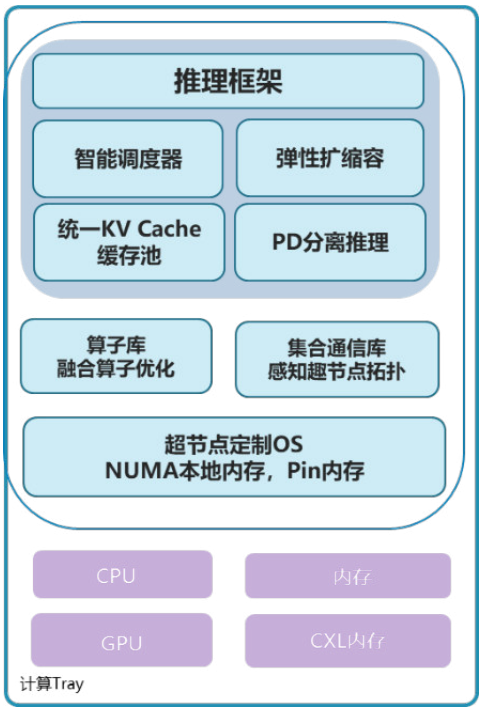


图193 通信平台

3. 框架与平台层

本层承接具体 AI 业务，提供模型开发、训练与推理所需的框架支持和平台服务。

- **训练与微调框架支持：**全面兼容主流并行训练框架，如 Megatron、DeepSpeed 及 PyTorch 等，支撑大规模模型的高效训练。
- **推理与服务框架：**深度优化推理引擎（如 vLLM, SGLang），提供智能调度器、统一 KV Cache 缓存池、PD（Prefill/Decode）分离推理等高级功能，满足海量 Token 并发与长上下文应用的生产级需求。
- **业务使能平台：**作为 AI 业务的统一调度与管理平台，提供模型服务、应用编排与解决方案交付能力，实现对计算、存储、网络资源的整合管理与调度。

4. 智能管控与运维层

该层实现对整个超节点集群乃至多个超节点的全域统管，保障系统的高可靠与高可用。

- **统一纳管与调度（业务平台）：**

- **超节点统一纳管：**将计算、驱动、OS 等封装为统一资源池，实现像调度一台计算机一样调度跨超节点的算力，大幅降低接入与调度门槛。
- **智能调度策略：**
  - **拓扑感知调度：**感知硬件拓扑，实现负载亲和性调度，最小化跨节点通信延迟，显著提升大模型训练效率。
  - **训推一体调度：**支持训练、推理、在线、离线任务的混合部署与分时复用，全面提升资源利用率。
  - **故障感知调度：**实现故障自动发现、业务重调度与恢复，防止故障扩散，保障业务连续性。
  - **逻辑超节点调度：**支持弹性切分超节点资源，灵活适配小模型推理与碎片化任务，提升资源利用精度。

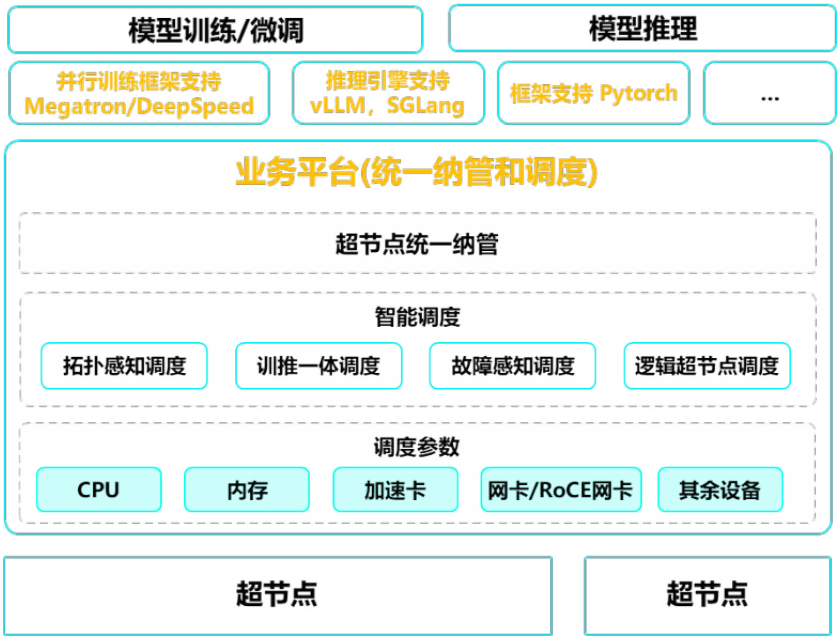


图194 业务平台

- **全栈运维分析（管控平台）：**
  - 控制组件（SeerEngine-DC）：负责智算网络、计算节点、存储设备的自动化配置、部署及训练流量调优。
  - 分析组件（SeerAnalyzer-DC）：聚焦作业级的算网存全域运维分析、流量路径分析与智能故障推理。
  - 监控组件（UC-IOM）：负责服务器、OS、存储、容器及第三方设备的基础设施监控与指标采集。
  - 管理组件（UniSystem）：实现服务器的批量部署、固件/驱动升级、基线管理及超节点计算 Tray 安装等生命周期管理。

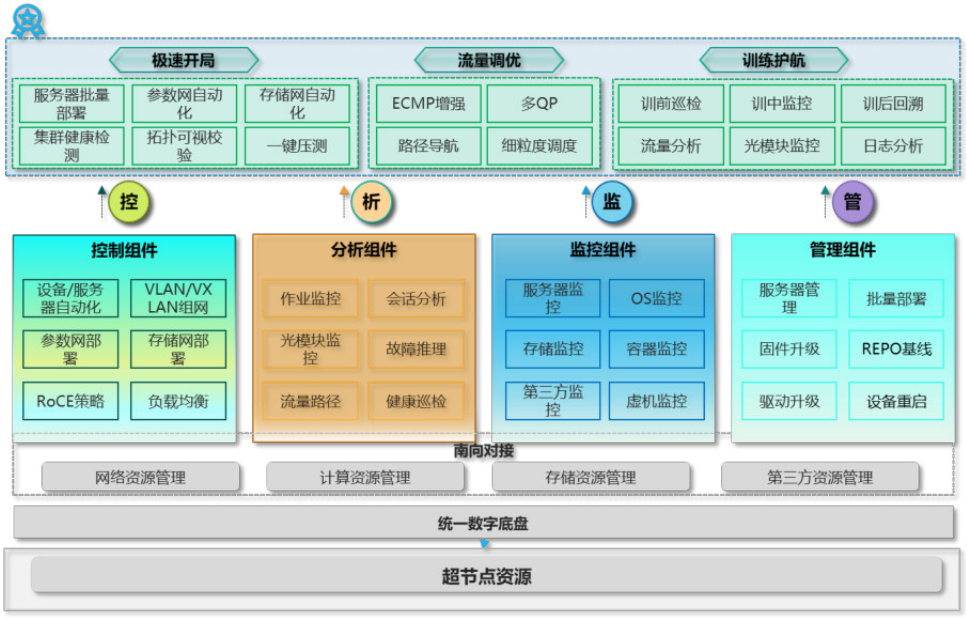


图195 管控平台

3.3 功能特性

通过全栈、分层、协同的软件创新，构建了面向 Agentic 时代的智算操作系统。其核心特点可归纳为以下三个维度：

1. 高易用：化繁为简，赋能业务敏捷

- **统一抽象，降低门槛：**通过“逻辑超节点”和统一纳管视图，将超大规模分布式系统抽象为可直观管理的单一资源对象。用户无需感知底层复杂的硬件互联细节，即可像调度一台计算机一样调度跨节点的海量算力，大幅降低了大规模智算资源的接入与使用门槛。
- **智能调度，提升效率：**集成拓扑感知、训推一体、故障感知、逻辑切分等智能调度策略。训练与推理任务可使用同一调度器，并支持 vGPU 等细粒度调度，实现负载亲和性部署与资源分时复用，预计可提升整体资源利用率 10%以上。
- **灵活适配，应对多变：**支持将物理超节点弹性切分为多个逻辑超节点，既能支撑万亿参数模型的集中训练，也能高效服务海量小模型推理与碎片化任务，灵活适配快速变化的业务场景。

2. 高可靠：全程护航，保障业务连续

- **可视化运维，破除黑盒：**提供从芯片、高速互联链路到作业层的多层可视化能力，图形化展示超节点内部资源关联关系与状态，彻底解决框内“黑盒”问题。
- **深度巡检，防患未然：**提供训前一键式深度健康巡检与网络性能压测，快速检测节点可用性及网络性能，在关键任务开始前排除潜在隐患。

- **全栈高可用与智能容错：**通过计算域与网络域的深度指标采集与自动故障感知，联动业务平台实现故障节点的快速发现、业务重调度与自动隔离，有效防止故障扩散，保障长周期训练与在线服务的业务连续性。

3. 高性能：软硬协同，释放极致算力

- **端到端优化，突破瓶颈：**从定制操作系统（NUMA 亲和、内存 Pin 驻留）、融合算子库（多粒度并行、硬件加速）、拓扑感知通信库到推理框架（PD 分离、智能 KV Cache）进行深度垂直优化，形成协同效应，共同打破传统性能瓶颈。
- **释放硬件潜能：**软件栈的深度优化确保 S80000 超节点的强大硬件算力得到充分释放，满足千亿/万亿参数模型的高效训练与低延时、高并发推理的严苛要求。
- **赋能 Agentic 应用：**针对智能体（Agent）应用特点，提供分级 KV Cache 智能缓存、会话管理等优化，显著降低复杂链式推理与多轮对话的重算开销与响应时间，为下一代 AI 应用提供高性能底座。

全球标准/生态格局和进展

超节点作为 ScaleUp 纵向扩展的系统，目前其标准和生态格局存在多样化发展的情况。英伟达目前已经构建完整且封闭的生态系统，其产品在市场上被广泛接受。而其他厂商也在建立自有或开放联合的生态系统。在每个生态系统中，我们都能看到三个关键要素，即算力、联接、软件栈。

- **算力：**以 GPU/AI 加速芯片为代表的算力，是超节点被市场接受的直接要素，因此拥有最新最强的算力芯片，并持续快速迭代，是当前驱动超节点生态成长的动力；
- **联接：**以 ScaleUp 网络为代表的联接技术，是构建超节点系统的骨架。除了 GPU/AI 芯片之外，还有内存、硬盘等设备也需要加入到联接之中。联接的范围越广，参与的厂商越多，生态的多样性才能发展得更好；
- **软件栈：**在硬件提供的算力，与用户的业务应用之间，需要软件栈提供另一种形态的“联接”。对于追赶英伟达的众多竞争厂商而言，软件栈可能是最难的生态构建环节，它不是靠一时的技术突破，而是需要长期的用户培养和积累。

1 专有生态

1.1 英伟达生态

英伟达 GPU

在英伟达的产品技术生态中，尤其是在超节点的范畴中，GPU、CUDA 和 NVLink 是英伟达三个关键技术。从 2006 年至今，英伟达已经发布了几十款 GPU（见下表），其 GPU 已经占据了领先的市场份额，积累了最大规模的用户生态。

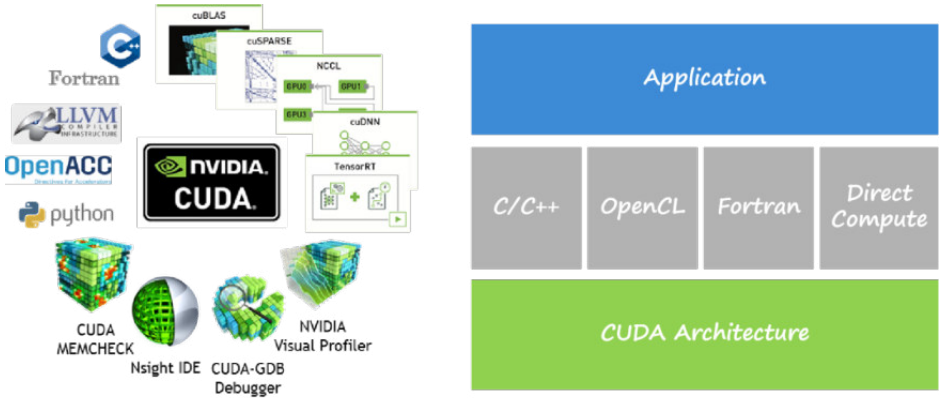
截至 2026 年 2 月，英伟达 Blackwell Ultra 架构的 B300 是当前全球单芯片算力最高的 GPU，其 15 PetaFLOPS FP4 算力、288GB HBM3e 显存，以及在 AI 性能基准测试 MLPerf v5.1 中的全面领先，均获得权威实证支持。

英伟达 GPU 架构	发布时间	产品型号清单	NVLink 对应关系
Tesla	2006年	C870, C1060, C1080, M1060	–
Fermi	2010年	C2050, C2070, C2075, C2090, M2050, M2070, M2070Q, M2075, M2090, X2070, X2090	–
Kepler	2012年	K8, K10, K20c, K20m, K20s, K20X, K20Xm, K40c, K40d, K40m, K40s, K40st, K40t, K80	–
Maxwell	2014年	M4, M10, M40, M60	–
Pascal	2016年	P4, P10, P40, P100, P100 DGXS	NVLink 1.0
Volta	2017年	PG500-216, PG503-216, V100, V100 DGXS, V100 FHHL, V100S	NVLink 2.0
Turing	2018年	T10, T4, T4G, T40	
Ampere	2020年	A2, A10, A10G, A10M, A16, A30, A30X, A40, A100, A100X, A800, AX800, PG506-207, PG506-217, PG506-232, PG506-242	NVLink 3.0
Ada	2022年	L2, L4, L20, L40, L40G, L40S, L40 CNX	
Hopper	2022年	H20, H100, H100 CNX, H100 NVL, H200, H200 NVL, H800	NVLink 4.0
Blackwell	2024年	B100, B200, RTX PRO 6000 Server	NVLink 5.0
Blackwell Ultra	2025年	B300	

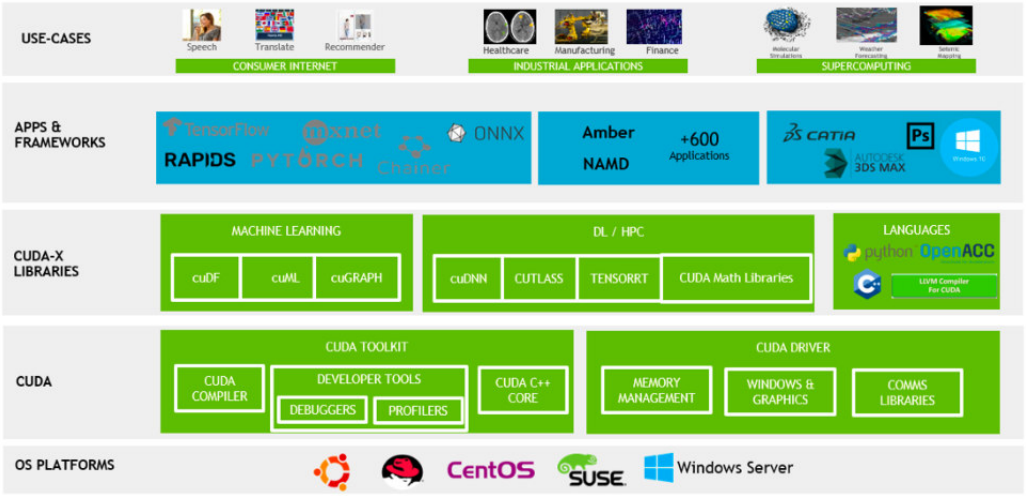
CUDA 生态

1. CUDA 平台介绍

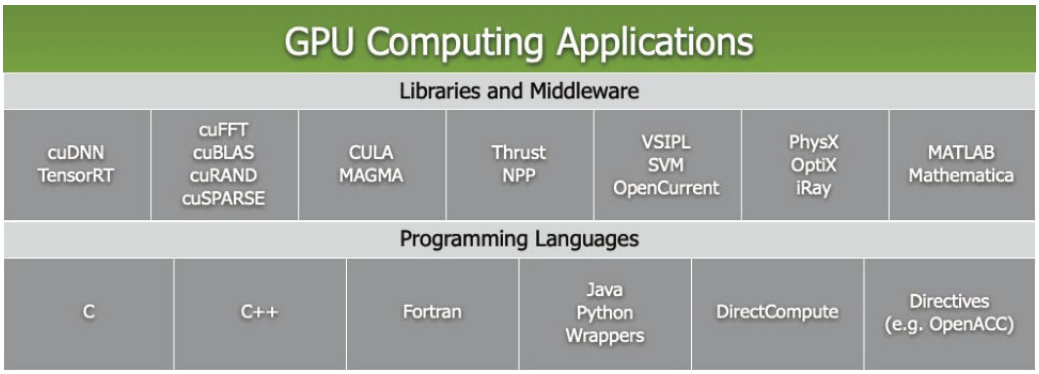
CUDA，即计算统一设备架构（Compute Unified Device Architecture），是英伟达在 2006 年推出的一种并行计算平台和编程模型。CUDA 是 NVIDIA 的加速计算平台，为应用程序提供利用英伟达 GPU 算力的软件层。开发者可以使用 C++、Python、Fortran 等语言编程，或借助如 PyTorch 等 GPU 加速库和框架，将 GPU 计算灵活集成到软件栈的各个层面，以获得更佳的功能与性能表现。



CUDA 使得开发者能够利用 GPU 的强大计算能力，来加速各种科学计算、工程模拟、数据分析等任务。在传统的计算模式中，CPU 是处理复杂逻辑和顺序任务的主力，而 GPU 则擅长处理大规模的并行计算任务。CUDA 的出现，极大地简化了在 GPU 上进行编程的复杂性，使得开发者能够更加高效地利用 GPU 资源，实现计算性能的飞跃。



CUDA 在软件方面由一个 CUDA 库、一个应用程序编程接口（API）及其运行库（Runtime）、两个较高级别的通用数学库，即 CUFFT 和 CUBLAS 组成。CUDA TOOLKIT 包括编译和 C++核，CUDA DRIVER 驱动 GPU 负责内存和图像管理。CUDA-X LIBRARIES 主要提供了机器学习（Meachine Learning）、深度学习（Deep Learning）和高性能（High Performance Computing）计算方面的加速库，APPS & FRAMEWORKS 主要对接 Tensorflow 和 Pytorch 等框架。

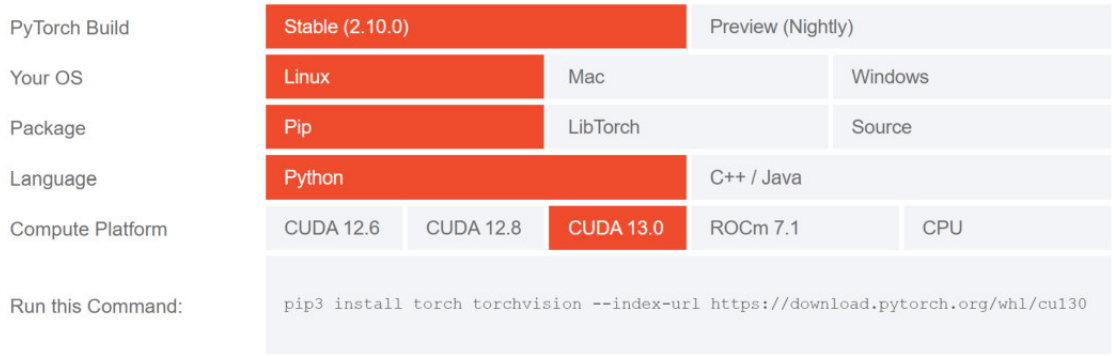


CUDA 提供了几个较为成熟的高效函数库，可以直接调用这些库函数进行计算，常见的包括：

- cuFFT：利用 CUDA 进行傅立叶变换的函数库
- cuBLAS：利用 CUDA 进行加速的完整标准矩阵与向量的运算库
- cuDPP：并行操作函数库
- cuDNN：利用 CUDA 进行深度卷积神经网络

2. CUDA 在 AI 训练和推理场景占据绝对生态优势

截至 2025 年的各类公开统计信息，在 AI 计算场景中，CUDA 生态覆盖了 80%~90%的市场份额，处于市场的绝对领先地位。从 AI 训练和 AI 推理这两类主要的细分场景来看，在 AI 训练场景中超过 85%的模型训练任务依赖于 CUDA 底座，以目前最流行的 AI 训练框架 PyTorch 为例，其默认的计算平台就是 CUDA，并且其版本的演进与 CUDA 的有密切关系。



在另一个主流的 AI 训练框架为 TensorFlow，其计算层也以 CUDA 提供的 cuBLAS、cuRAND、cuDNN 为默认的数值计算层组件。

层	功能	组件
视图层	计算图可视化	TensorBoard
工作流层	数据集准备、存储、加载	Keras/TF Slim
计算图层	计算图构造与优化 前向计算/后向计算	TensorFlow Core
高维计算层	高维数组处理	Eigen
数值计算层	矩阵计算/卷积计算	BLAS/cuBLAS/ cuRAND/cuDNN
网络层	通信	gRPC/RDMA
设备层	硬件	CPU/GPU

在目前全球的 AI 训练场景中，PyTorch 和 Tensorflow 合计已经超过 7 成的市场占比，紧随其后的是百度的 PaddlePaddle，也同样将 CUDA 作为首选的计算层，仅华为的 MindSpore 出于商业的考虑，从 2025 年中期发布的 MindSpore 2.6.0 版本开始，才取消了对 CUDA 的支持。

在推理场景，虽然推理引擎体现出多元化发展的趋势，但除了桌面和移动端等有限的场景，多数推理引擎将 CUDA 作为默认的选项。

推理引擎	CUDA 依赖性	适用硬件	部署难度	典型应用场景
vLLM	强依赖	NVIDIA GPU	中高	高并发API服务、企业级大模型部署、RAG系统
TensorRT-LLM	强依赖	NVIDIA GPU	高	低延迟生产环境、金融/医疗实时推理、边缘AI加速
Hugging Face TGI	强依赖	NVIDIA GPU	中	OpenAI兼容接口服务、多轮对话机器人、云原生部署
SGLang	强依赖	NVIDIA GPU	中	Agent系统、工具调用密集型任务、结构化生成
LMDeploy / TurboMind	可选	NVIDIA GPU / 昇腾 NPU	中	混合异构部署、私有云环境
llama.cpp	非必需	CPU / NVIDIA GPU（可选）	低	本地开发、笔记本推理、树莓派等边缘设备
MNN	否	ARM CPU / 骁龙/NPU	极低	移动端App、嵌入式设备、手机本地大模型运行

根据主流媒体的报导，截至 2025 年底，CUDA 生态目前已经拥有超过 450 万注册开发者的生态系统，值得关注的是，上述数据仅统计了英伟达官网的注册开发者数量，而如果将非注册用户以及通过 PyTorch/TensorFlow 调用 CUDA 的用户数量统计进来，则开发者的规模极可能提升一个数量级。

3. CUDA 针对超节点提供多项优化能力

2025 年底，在 CUDA 的最新版本 CUDA 13.1 中，多个新特性与能力直接针对英伟达超节点（Super Pod）架构进行了优化和适配，以充分发挥超节点中大规模 GPU 集群的并行计算潜力。这些特性不仅提升了编程效率，也优化了资源调度与性能分析，确保在超节点环境下实现高性能、高效率的计算。

- **CUDA Tile 编程模型：简化超节点并行编程**

  CUDA Tile 是 CUDA 13.1 的核心创新，其设计初衷之一就是简化大规模并行计算的编程复杂度，特别适用于超节点中成百上千个 GPU 协同工作的场景。

  - 抽象层级提升：开发者无需再手动管理线程块划分、共享内存分配等底层细节，而是以“数据块”（Tile）为单位组织计算，更贴近算法设计思维。
  - 跨架构兼容性：针对 Blackwell 架构（计算能力 10.x 和 12.x）优化，未来只需重新编译即可适配下一代硬件，显著降低超节点系统升级成本。
  - 性能优化：在超节点环境下，Tile 编程模型能更高效地利用 Tensor Core 和高带宽内存（HBM），提升大规模矩阵运算性能。

- **Green Contexts（绿色上下文）：精细化资源管理**

  超节点通常包含大量 GPU 资源，如何在多任务并发中合理分配资源是关键挑战。CUDA 13.1 引入的 Green Contexts 为这一问题提供了有效解决方案。

  - 资源隔离：通过将 GPU 的 SM（流式多处理器）划分为多个逻辑分区，确保高优先级任务（如低延迟推理）能够独占部分资源，避免资源争抢。
  - 多租户支持：在超节点的多用户或多任务环境中，Green Contexts 可实现资源的精确分配，提升系统整体利用率与服务质量。
  - 运行时 API 支持：开发者可通过运行时接口直接控制资源分配，提升超节点中任务调度的灵活性。

- **MLOPart（内存局部性优化分区）：提升超节点内存效率**

  在超节点中，多个 GPU 协同工作时，内存访问模式的优化至关重要。CUDA 13.1 引入的 MLOPart 技术，专为 Blackwell 架构优化，旨在提升内存访问效率。

  - 虚拟化与分区：将物理 GPU 虚拟化为多个逻辑分区，每个分区具有独立的内存访问路径，减少跨分区访问带来的延迟。
  - 内存局部性增强：优化数据在共享内存和缓存中的分布，减少重复加载，提升超节点中大规模数据处理的吞吐量。
  - 云原生场景优化：在多租户云环境中，MLOPart 可确保每个用户任务的内存访问模式独立优化，提升整体服务质量。

- **Nsight Compute 2025.4：超节点性能分析支持**

  为了帮助开发者在超节点环境中优化性能，CUDA 13.1 配套的 Nsight Compute 2025.4 工具也进行了增强，支持对 CUDA Tile 内核的深度分析。

  - Tile 内核分析：新增“Tile Statistics”（Tile 统计）部分，展示 Tile 维度、关键管线利用率等信息，帮助开发者理解超节点中的并行计算效率。
  - 设备端图节点分析：支持对设备端启动的 CUDA 图节点进行性能追踪，适用于超节点中复杂的任务调度与依赖关系分析。
  - 源码映射：性能指标可映射到 cuTile 内核源代码，便于开发者定位性能瓶颈。

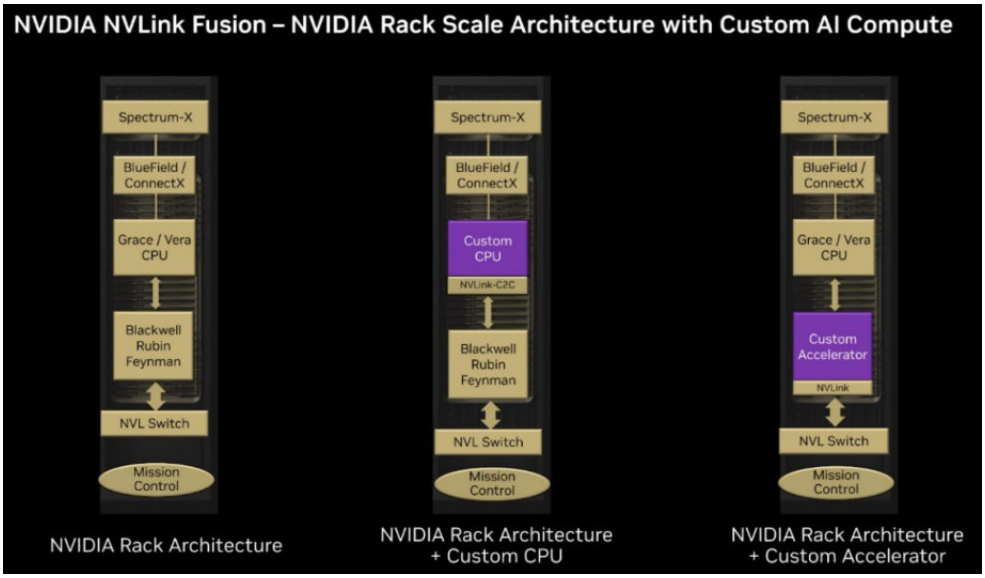
- **数学库与确定性执行：保障超节点稳定性**

  在超节点中，大规模模型训练和推理对计算结果的确定性要求极高。CUDA 13.1 的数学库（如 cuBLAS）也针对此进行了优化。

  - 双精度仿真：在 Blackwell GPU 上支持 FP32/FP64 的仿真 GEMM 运算，提升大模型训练的精度与稳定性。
  - 确定性执行：确保在不同 GPU 上运行相同任务时，结果保持一致，这对超节点中多节点协同计算至关重要。

NVLink Fusion 生态

2025 年 5 月 19 日，英伟达正式发布了 NVLink Fusion，这是英伟达针对其专有的 NVLink 技术的重大转变。在过去的英伟达算力集群中，只能使用英伟达的芯片，但现在英伟达出让了一部分权限，将 NVLink 开放给行业，客户可以把其他品牌的芯片也加入到算力集群里。需要关注的是，NVLink Fusion 仅开放了计算部分，即 CPU 和加速芯片，而网络部分仍然是采用英伟达的全家桶方案。



NVLink Fusion 目前在 CPU 方面已经有 ARM、Intel、富士通、高通、SiFive 加入其中。CPU 的技术路线覆盖了主流的 x86、ARM 和 RISC-V，但目前的 CPU 型号还相对有限。

厂商	CPU 型号	架构路线
ARM	定制版Arm Neoverse处理器	ARM
Intel	定制版至强Xeon处理器	X86
Fujitsu	FUJITSU-MONAKA	ARM
Qualcomm	Oryon架构的定制CPU	ARM
SiFive	待发布	RISC-V

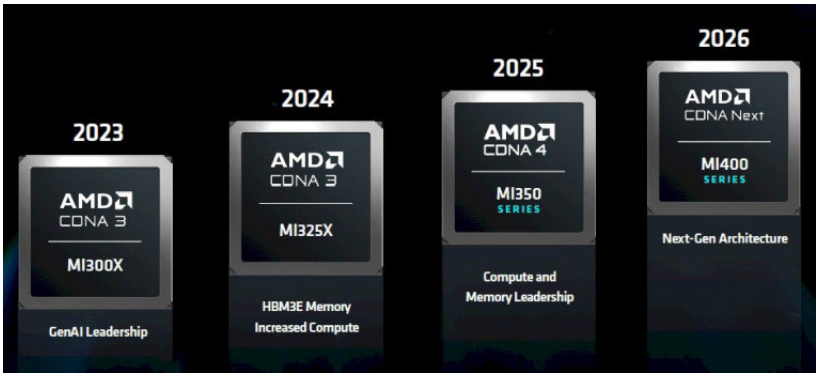
至于 NVLink Fusion 所支持的 GPU/加速芯片，则包括了 Marvell、联发科、三星等厂商，对应的 GPU 芯片的市场份额相对偏小。

从上述生态支持的情况看，英伟达的 NVLink Fusion 目前仍处于起步阶段，CPU、GPU 或加速芯片的选项相对有限，目前市场上也尚未发布此类的超节点系统或集群方案，因此其生态的整体影响力目前还有待市场观察。

1.2 AMD/Intel 生态

GPU 及其 ROCm 生态

AMD 是英伟达 GPU 领域的长期竞争对手，AMD 在 2017 和 2018 年曾经两次推出 Instinct MI 系列数据中心 GPU，但当时的几款型号 MI6、MI8、MI25 在性能上距离同期的 NVIDIA GPU 差距较大，这三款 GPU 的 FP64 算力均未超过 1TFLOPS，直到 AMD 在 2023 年发布了 MI300 系列才有所改变。



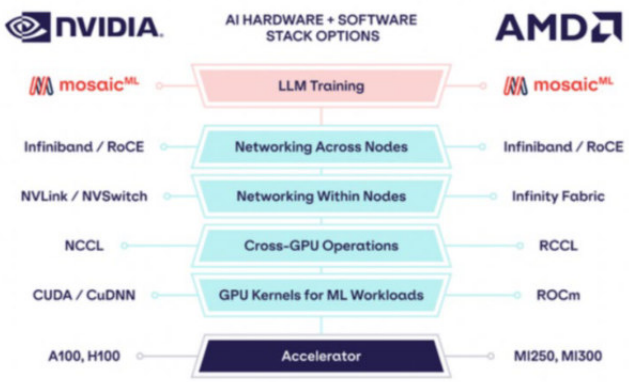
架构	发布时间	AMD 产品型号清单	英伟达对标型号
GCN3/GCN4	2014年	MI6, MI8	–
GCN5.0/ GCN5.1	2017年	MI25, MI50	–
CDNA	2020年	MI100	–
CDNA 2	2021年	MI210, MI250, MI250X	–
CDNA 3	2023年	MI300A, MI300X, MI325X	H100/H200
CDNA 4	2025年	MI350, MI355X	B200
CDNA 5	2026年待发布	MI400, MI430X, MI440X	B200/B300

在 GPU 的 ScaleUp 互联技术上，Infinity Fabric（IF）是 AMD 提出并长期演进的一体化高速互连架构，用于连接 CPU、GPU 以及 SoC 内部的多个计算与功能模块。

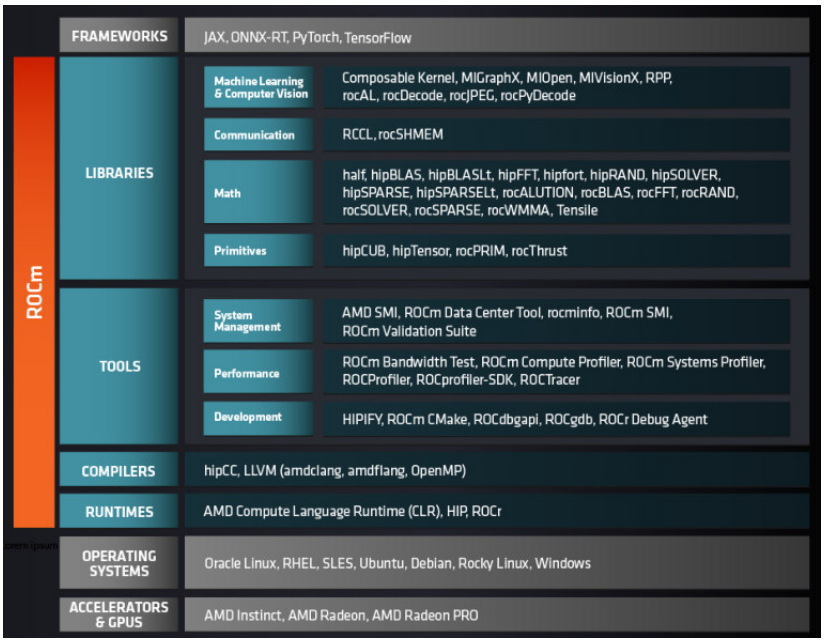
与 NVLink、UB、ICI 等主要面向系统级或超节点级 Scale Up 场景的互连技术不同，Infinity Fabric 最初的设计目标并非专注于超大规模加速器互联，而是作为贯穿芯片内部、封装级以及节点内系统的统一互连骨干，支撑 AMD 多核、多 Die 与异构计算架构的发展。在 AI 与高性能计算系统中，Infinity Fabric 被用于连接 AMD EPYC CPU、Instinct GPU 以及相关 I/O 与内存控制模块，为节点内多加速器协同提供较高带宽和一致的通信语义。

从生态角度需要关注的是，Infinity Fabric 起初是作为 AMD 专有的互联协议，为了挑战英伟达的生态主导地位，AMD 将 Infinity Fabric 技术部分贡献给了 UALink 联盟。

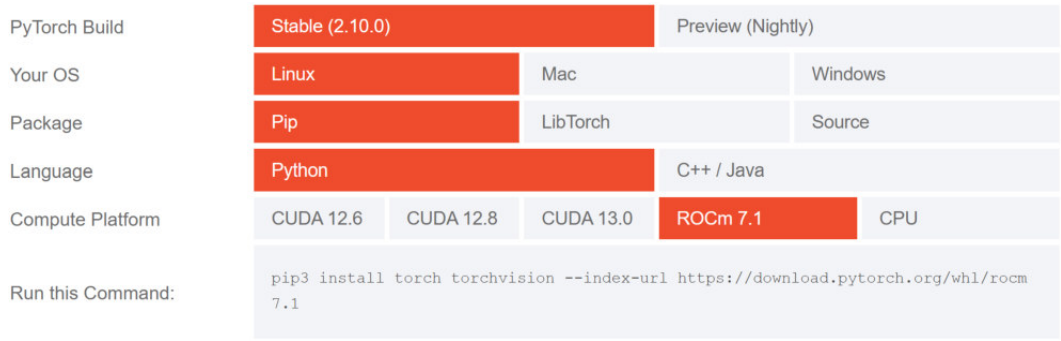
为了构建 AMD 自有的算力生态，AMD 也推出并开源了 ROCm（Radeon Open Compute），作为其对标 CUDA 的软件平台和方案。ROCm 在 2016 年发布，目前已经演进到 ROCm7 版本。



AMD ROCm 作为追赶者，具有与 CUDA 类似的定位和功能，也提供了一系列的计算库，为开发者提供统一、抽象化的编程接口。



AMD ROCm 与 AI 训练框架的生态适配情况相比英伟达 CUDA 存在一定的差距。其中的主要差异是 PyTorch 和 TensorFlow 目前仅支持 Linux 版本的 ROCm，相比而言 CUDA 支持可支持 Linux 和 Windows 系统。这个差异主要影响个人用户，而企业环境因为主要使用 Linux 因而不受影响。



至于推理引擎的支持情况，ROCm 所支持的推理引擎也相对偏少。

推理引擎	是否支持 ROCm
vLLM	否
TensorRT-LLM	否
Hugging Face TGI	否
SGLang	否
LMDeploy / TurboMind	否
llama.cpp	支持
MNN	否
Megatron-LM	支持
VERL	支持

截至 2025 年，AMD 作为全球第二大 GPU 厂商，其 ROCm 生态相对英伟达的 CUDA 生态仍有较大的差距，用户规模数量尚无确切的统计数量，根据公开信息显示，其 GPU 主要集中在微软等少量云巨头客户，而缺少中小客户和个人客户，因为后者对于生态成熟度的依赖性远高于云巨头。

Intel 生态

Intel 作为 GPU 市场的另一个主要追赶者，其对标英伟达的产品线为 Gaudi 系列 AI 加速芯片，它是由英特尔 2019 年收购的以色列 AI 初创公司 Habana Labs 设计的，采用 ASIC 架构，有别于 Intel 原有的通用 GPU 产品。目前英特尔最新一代的 AI 加速器是 Gaudi 3，该产品已于 2024 年发布，但截至 2026 年初，Intel 仍无 Gaudi 4 的发布计划。

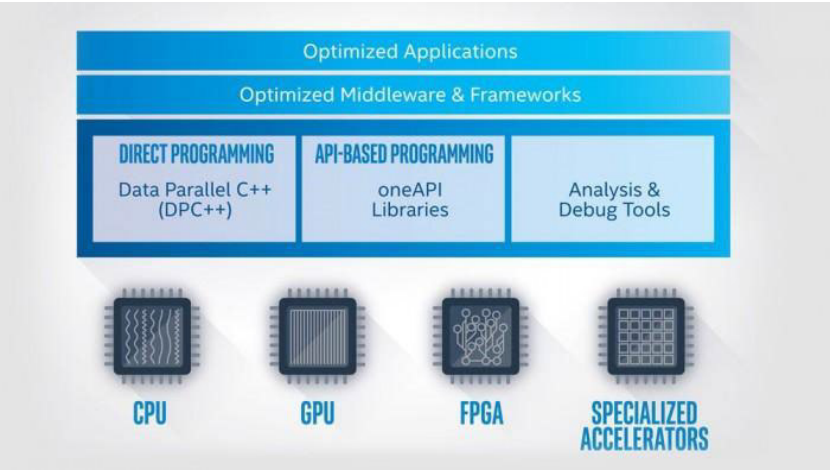
Intel Gaudi 型号	发布时间	英伟达对标型号
Gaudi	2019年	T4
Gaudi 2	2022年	A100
Gaudi 3	2024年	H100/H200

Intel Gaudi 芯片之间主要通过**基于以太网的 RDMA（Remote Direct Memory Access）技术**实现互联，具体采用 RoCE（RDMA over Converged Ethernet）协议，这是英特尔在 AI 加速器互连架构中的核心设计选择。

与 NVIDIA NVLink 或 AMD Infinity Fabric 等专有高速互连不同，Gaudi 系列（包括 Gaudi 3）坚持使用标准以太网基础设施进行芯片间通信，具备良好的开放性和可扩展性。以下是其互联机制的关键特点：

- Gaudi 3 集成了24 个 200Gb/s 的 RoCE 网卡（NIC），总对外带宽高达1.2TB/s，相较 Gaudi 2 的 100Gb/s 接口实现翻倍提升。
- 这些端口被划分为两类用途：
    - 21 个端口用于节点内互联（Scale-Up）：连接同一服务器内的其他 Gaudi 加速器，实现低延迟、高带宽的芯片间通信；
    - 3 个端口用于节点间扩展（Scale-Out）：连接其他计算节点，支持大规模集群部署。
- 由于使用标准以太网协议，Gaudi 集群可以利用现有的数据中心网络架构（如 ToR 交换机、Spine-Leaf 拓扑）进行横向扩展，无需额外投资专有网络硬件。

为了打造 Intel 的算力生态，Intel 也在 2019 年发布了 oneAPI 平台，同样对标英伟达的 CUDA。为了体现出 Intel 的差异化优势，Intel 将 oneAPI 打造成了一个横跨 CPU、GPU、ASIC 和 FPGA 等其他专用加速硬件的平台。



oneAPI 的核心理念是“一次编写，到处运行”。它提供了一套统一的编程接口和工具，使开发人员能够为 CPU、GPU、FPGA 等不同类型的处理器编写单一代码库，而不需要为每种架构单独维护代码。因此 oneAPI 不仅是一套工具或平台，更是一项基于开放规范的行业计划。

但由于 Intel 在 GPU 算力市场的弱势地位，oneAPI 的理念虽好，但仍需要时间的考验，目前在市场的生态规模仍相当有限。

2 国际开放生态

2.1 UALink 联盟

1. UALink 介绍

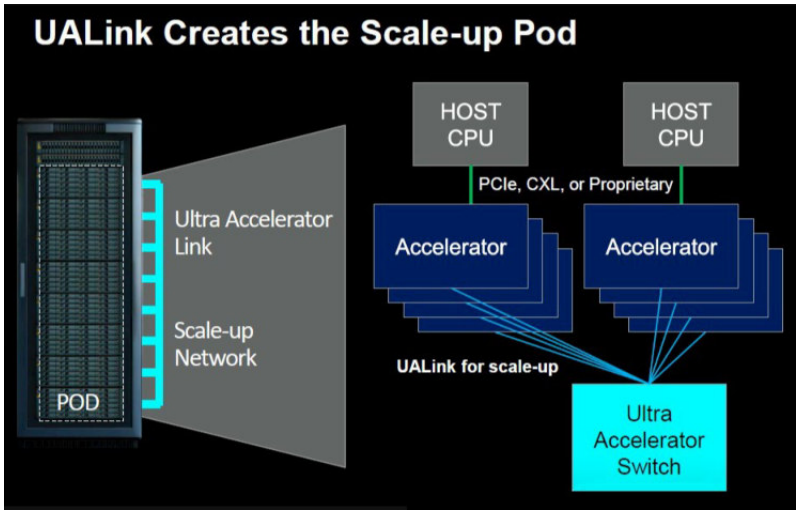
UALink（Ultra Accelerator Link）包含了 AMD、Astera Labs、AWS、思科、谷歌、HPE、英特尔、Meta、微软、字节、腾讯、中兴、新华三、ARM 等行业主要厂商的技术联盟。

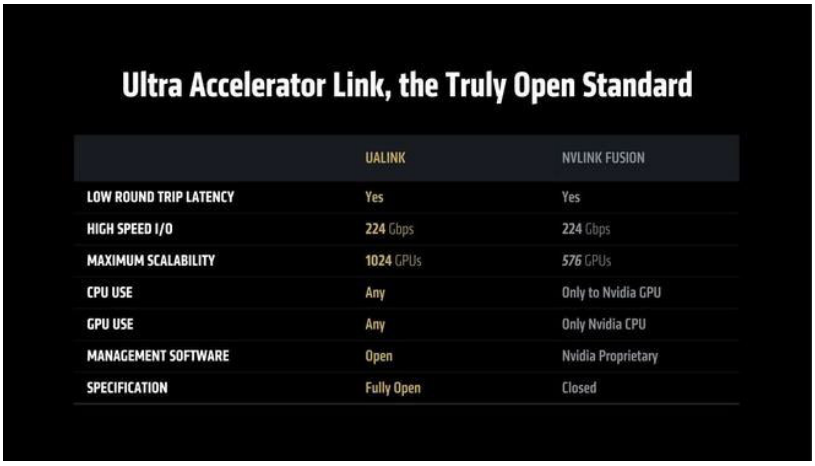
UALink 核心战略是“通过利用现有的以太网基础设施，包括线缆、连接器、中继器和管理软件，来降低总体拥有成本（TCO）”。目的是通过开放的生态，对抗英伟达的专有生态。

对比维度	NVIDIA“专有生态”	UALink“开放联盟”
核心哲学/模式	垂直整合的专有模式	专为AI优化的开放标准内存网络
优势	通过控制每一个环节，实现了无与伦比的性能和优化。	生态协作带来的选择多样性和价格竞争力。
劣势/挑战	高昂的成本、深度的厂商锁定和单一的创新来源。	管理一个多厂商联盟所固有的复杂性和潜在的决策滞后。

2. UALink 当前进展

2025 年，UALink 发布了 1.0 版本的规范，可连接 1024 个 GPU、单通道 200GT/s 带宽，四通道配置下可达 800GT/s。UALink 的目标规格相对于英伟达的 NVLink Fusion 在有明显的优势。





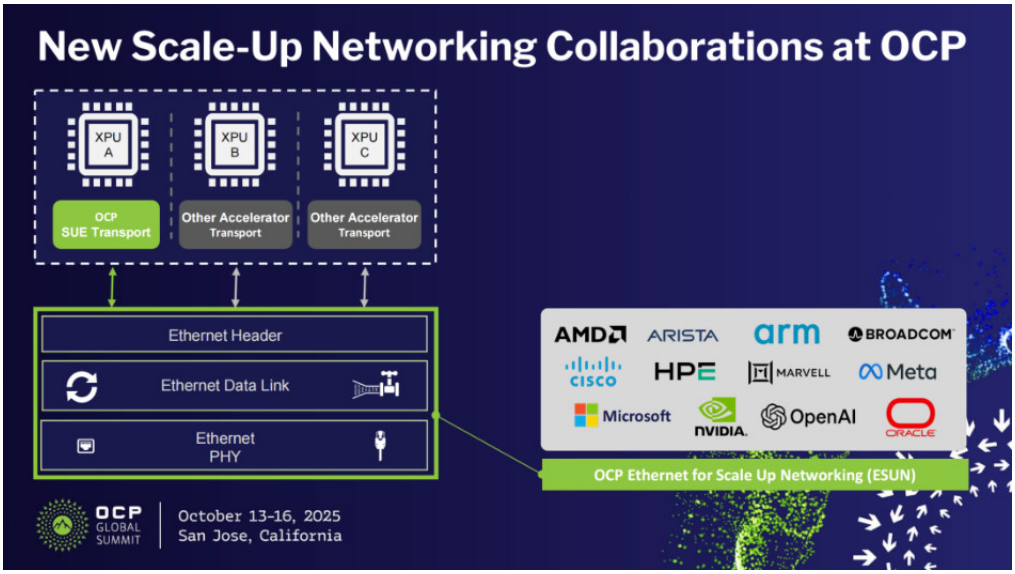
虽然 UALink 的规划目标非常具有吸引力，但目前还没有采用 UALink 的整机系统上市，至少要等到 2026 年下半年甚至 2027 年，因此 UALink 在算力市场快速变化的背景下，需要跨过生态初期的重重困难。

2.2 ESUN 联盟

1. ESUN 介绍

2025 OCP（开放计算项目）全球峰会上，OCP 正式宣布全新网络项目工作组 —— ESUN（Ethernet for Scale-Up Networking，以太网纵向扩展网络）。ESUN 为 OCP 框架下的开放式技术协作平台，目标为：“利用标准以太网（Ethernet）的生态与成本优势，构建可扩展、低延迟、高带宽的 AI 加速器互联架构。”

ESUN 的成员包括 AMD、Arista、博通、思科、HPE、英特尔、Meta、微软、英伟达（NVIDIA）、OpenAI、Oracle 等，与 UALink 联盟相比最大的差异，在于英伟达也参与了 ESUN 联盟，并且增加了博通 Broadcom，因此相对于 UALink 的“反英伟达”特点，ESUN 体现出了更为广泛包容性。



UALink 与 ESUN 虽然都是为了改变英伟达一家独大的市场局面，但在彼此的生态发展过程中，UALink 和 ESUN 成为彼此最直接的竞争对手，也就是说两者决出的胜者，才有可能撼动英伟达的市场地位。

对比维度	UALink	ESUN
战略定位	合纵联盟，开放标准	统一以太网江湖
主要目标	打破NVLink锁定	统一Scale-up/out网络
主要推动者	汇集了NVIDIA主要竞争对手和客户（AMD、谷歌、英特尔、微软、Meta等）；OCP 2025更新：博通已退出董事会。	包括了几乎所有关键参与者：AMD、Arista、ARM、博通、思科、HPE、Marvell、Meta、微软、NVIDIA、OpenAI和Oracle 。

对比维度	UALink	ESUN
架构哲学	基于标准PHY的全新协议：在标准的以太网物理层之上，设计一个全新的、为特定目的构建的协议栈（DL/TL）	模块化与标准化分工： ESUN（网络交换结构）和SUE-T（端点传输协议）双线并战
OCP 2025关键举措	/	ESUN工作组正式成立

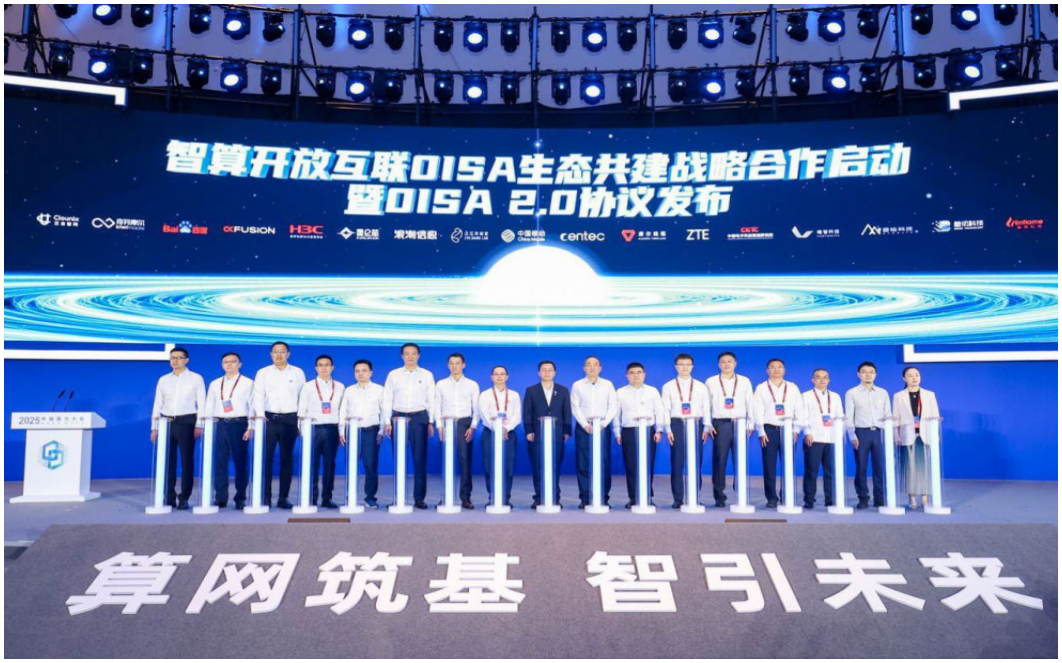
3 国内开放生态

3.1 OISA 联盟

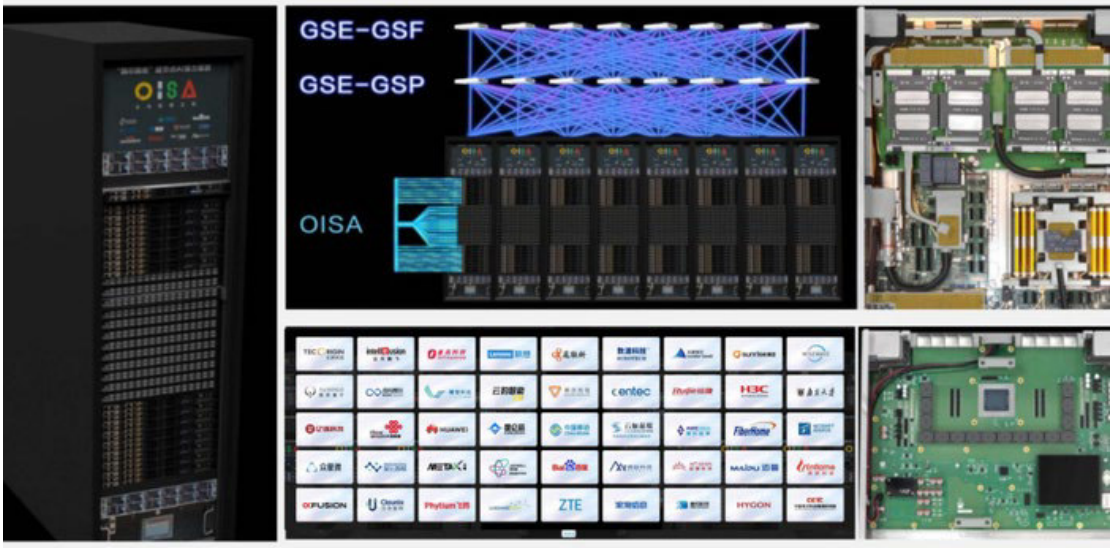
OISA（Omni-directional Intelligent Sensing Express Architecture，全向智感互联）是中国移动提出的GPU 卡间互联协议体系，旨在提供 GPU 卡间高速互联标准，包括大规模 GPU 对等互联、极致报文格式、数据层流控和重传以及高效物理传输等解决方案。

OISA 1.0 协议在 2024 年发布，支持 128 张 GPU 通过 8 个 Switch 芯片互联，任意卡间互联带宽达到800GB/s，每个 Switch 芯片支持 128 个端口、交换容量达到 51.2 Tb/s。

OISA 2.0 协议在 2025 年的中国算力大会上发布，将支持的 AI 芯片数量提升至 1024 张，带宽突破 TB / s 级别，AI 芯片互联时延缩短至数百纳秒，具备支持原生内存语义、创新 TLP 报文重构技术、支持智能在途感知、集合通信硬 件加速等多个核心技术特征。



目前 OISA 联盟已经包含了 GPU 芯片、Switch 芯片、整机厂商等几十个成员单位，相关的产品和系统也有做部分早期发布，但距离市场的实际场景落地仍待观察。



3.2 华为生态

华为以昇腾芯片为基础，采取了与英伟达竞争的生态策略，在硬件、软件栈和系统级（含超节点）均进行了较为深入的生态建设。

1. 华为 AI 芯片

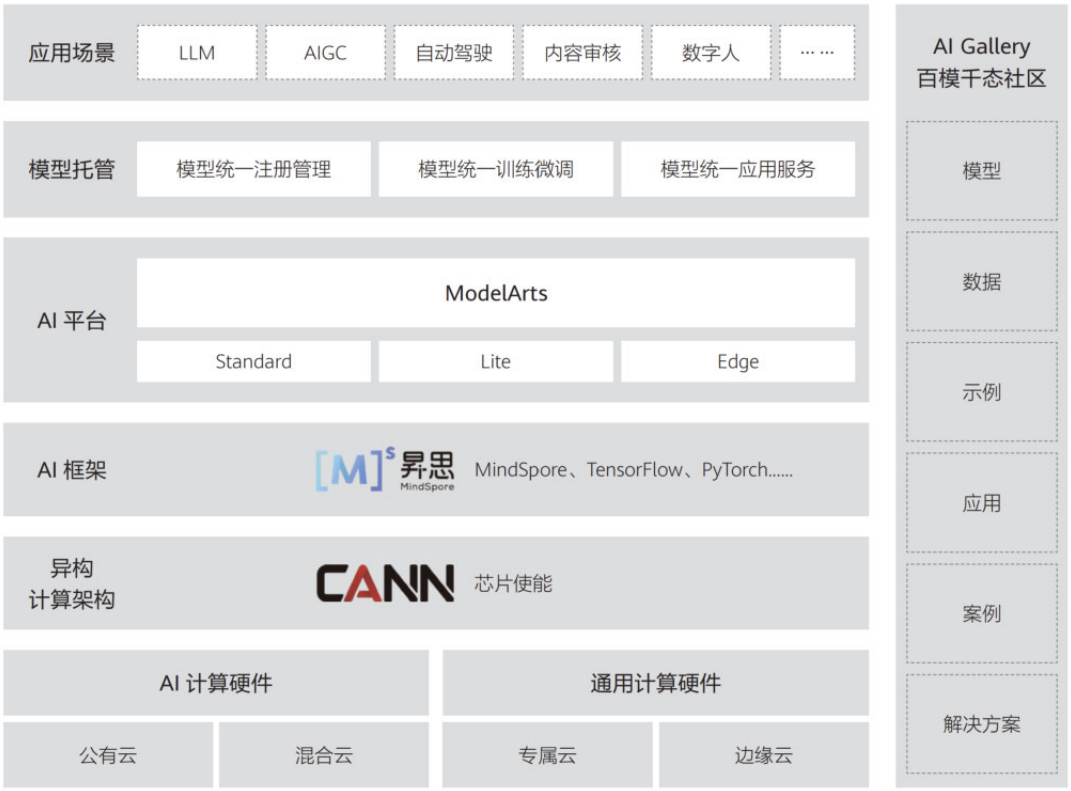
华为生态的基础是其昇腾系列 AI 芯片。在数据中心领域，华为与英伟达对标的产品线为昇腾 910 系列芯片，目前已经发布到 910C 产品。

华为昇腾型号	发布时间	英伟达典型对标型号
910	2019年	–
910B	2023年	A100
910C	2025年	H100

由于多种客观原因导致，华为昇腾芯片的发布时间，基本上相对其对标的英伟达要晚上几年，因此华为需要在生态建设上要做更多的投入，包括通过华为云、生态伙伴网络等多种模式和渠道。

- 基于华为云的生态建设模式

  华为作为国内主流的云计算厂商，具备公有云和私有云的方案，可以快速、规模化的进行昇腾芯片和配套软件的推广。



截止 2025 年底，使用华为云昇腾 AI 云服务的全球客户增长到上千家，涉及昇腾芯片之上的大规模算力集群、计算引擎 CANN、AI 开发框架 MindSpore、ModelArts AI 平台等多样化的云服务。

- 昇腾生态伙伴网络

除了云计算，昇腾在应用软件、基础模型、基础软件、生态运营方面，也在构建其生态网络。根据华为的官方信息，截至 2025 年底，昇腾已经培养较大规模的开发者和合作伙伴群体，推出了多个开源项目、硬件产品和行业解决方案。

2. 华为 AI 软件栈

华为的 AI 软件栈包含了 CANN、MindSpore、AI 平台（ModelArts 系列产品）三类主要的组成部分，共同构建了从底层的计算加速、AI 框架到平台使能的多层次软件栈。

- CANN

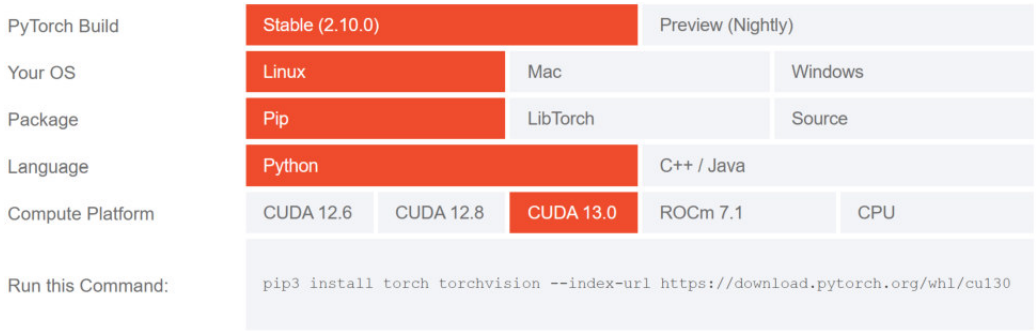
  CANN 是华为针对 AI 场景推出的异构计算架构，对上支持多种 AI 框架，对下服务 AI 处理器与编程，发挥承上启下的关键作用，是提升昇腾 AI 处理器计算效率的关键平台。在华为的软件栈中，CANN 的定位与英伟达软件栈的 CUDA 类似，是最基础的计算层软件。

作为计算层的基础软件，CANN 与 PyTorch 等 AI 框架的适配是首要的生态工作。截至 2026 年 1 月，CANN 与 PyTorch 的 2.90 版本完成了正式的适配。

CANN 版本	PyTorch 版本
商用版：8.5.0 社区版：8.5.0	2.6.0
商用版：8.5.0 社区版：8.5.0	2.7.1
商用版：8.5.0 社区版：8.5.0	2.8.0

CANN 版本	PyTorch 版本
商用版：8.5.0 社区版：8.5.0	2.9.0

值得关注的是，PyTorch 2.9.0 的正式发布时间为 2025 年 10 月，而 CANN 的在 2026 年 1 月就发布了正式的适配版本，仅耗费约一个季度的时间就完成了快速适配，这对于生态而言是十分重要的能力。



同时我们也应当看到，在 PyTorch 的官方配套中，仍只有英伟达的 CUDA 和 AMD 的 ROCm，这意味着 CANN 对 PyTorch 的适配工作主要由华为完成，从生态的长期发展而言尚未进入良性循环。

- MindSpore

MindSpore(昇思)是华为公司于 2019 年发布并于 2020 年开源的 AI 框架，与 PyTorch、TensorFlow 对标。

根据行业研究机构 Omdia（Informa tech 集团旗下国际信息与通信技术研究机构）发布了《中国人工智能框架市场调研报告》(2023 年)，华为昇思 MindSpore 与 PyTorch、TensorFlow、PaddlePaddle 等人工智能框架在知名度与使用率市场份额上处于第一梯队。

而根据华为官方信息，截至 2025 年底，MindSpore 的社区用户已经达到 350 万+，已经具备了一定的生态基础。

MindSpore 所支持的硬件平台和计算层，除了华为自己的昇腾 CANN 之外，还支持英伟达 CUDA，即 MindSpore 具备独立发展生态的技术条件。

硬件平台和计算加速	操作系统	MindSpore 支持状态
Ascend CANN	Linux-x86	✔
	Linux-aarch64	✔
Nvidia CUDA	Linux-x86	✔

- AI 平台软件

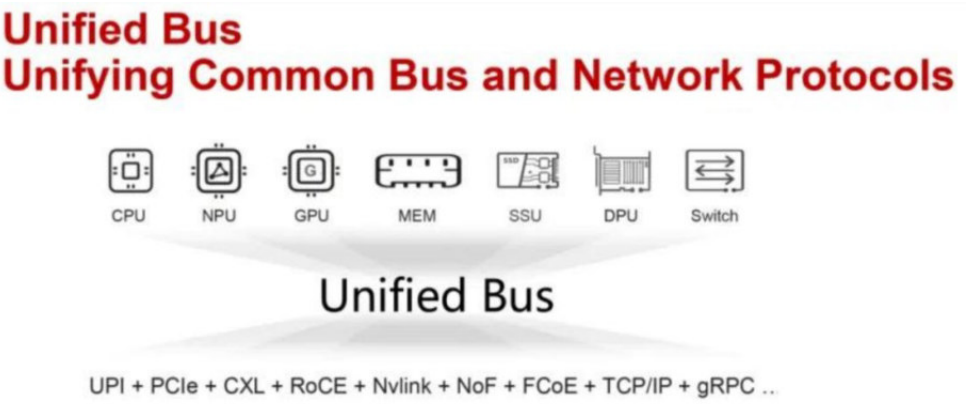
根据面向的用户群体差异，华为在云上和云下提供了不同的 AI 平台方案。其中 ModelArts 平台是华为云提供的面向开发者的一站式人工智能 AI 开发平台，以云服务的方式在公有云和私有云中提供，具有快捷易用的特点，是华为云上 AI 业务的重要入口。

而在云下，华为提供了一套更偏向于基础的开发套件和工具，包含集群的管理、端边协同、训练加速、推理引擎等部分，面向更为弹性变化的使用场景。

3. UB 协议介绍

华为于 2019 年开始研究灵衢(UnifiedBus),随后发布灵衢 1.0 商用 验证,于 2025 年 9 月发布并开放灵衢 2.0 技术规范。

UB 协议栈由物理层、数据链 路层、网络层、传输层、事务层、功能层以及 UMMU、UBFM(UB Fabric Manager) 组成，对于内存交互，UB 支持 UBPU 中的计算单元直接发起同步和异步访存指令，减少控制命令交互，实现百 ns~us 级低时延；对于集群大规模组网，UB 除了支持采 用多级 UB Switch 扩展组网之外，还支持通过 UBoE 与以太 Switch 对接，实现融合 组网，以及通过 OCS 组网，实现可变拓扑，助力集群规模扩张。



4. UB 标准进展

UB 协议已经完成 2.0 标准的设计，并通过开放标准和建设灵衢社区，硬件开放、软件开源，支持伙伴打造面向行业的超节点场景化解决方案，加速开发者高效自主创新，共建繁荣生态。

灵衢®基础规范	定义灵衢基础规范2.0版本，包括灵衢系统组成、协议、以及编程模型等。本文档帮助读者理解符合灵衢基础规范的设备和系统的交互行为、互操作要求、编程模型、资源管理等
灵衢®固件规范	配套灵衢协议，定义灵衢设备固件的架构设计、交互逻辑、功能及接口，确保固件在全生命周期内满足可靠性、安全性和兼容性要求灵衢固件规范
灵衢®使能操作系统参考设计	介绍操作系统灵衢组件（UB OS Component）的架构、功能及外部接口，方便应用开发者、驱动开发者以及OS开发者等了解操作系统灵衢组件和基于操作系统灵衢组件进行开发
灵衢®系统高阶服务软件架构参考设计	介绍支持面向智算和通算场景灵衢产品的系统高阶服务架构、所需的主要软件功能模块及接口功能。便于软件开发工程师、技术支持、企业技术负责人等使用、开发、管理和维护灵衢产品
灵衢®系统管控运维软件架构与接口参考设计	介绍基于灵衢的超节点智算和通算场景管控运维所使用的软件架构和接口功能参考。方便运维系统开发者、管控系统开发者、运维工程师和系统管理员等使用、开发、管理和维护灵衢产品
基于灵衢®的超节点参考架构白皮书	定义基于灵衢的超节点参考架构，其拥有总线级互联、协议归一、平等协同、全量池化、大规模组网和高可用性六大特征，帮助您迎接智能化时代算力基础设施挑战

UB 相关软件已经开源，包括操作系统软件组件，以及高阶服务软件等。

5. UB 软件生态进展

华为 UB 目前已经开源，并且绑定了华为自家的欧拉操作系统，以“木兰宽松协议 2.0”进行授权。因此华为软件生态的范围，仍然还是华为原有的生态合作伙伴，软件生态仍处于萌芽状态。

6. UB 硬件生态进展

虽然华为 UB 的协议标准已经发布，但在产品化落地中，仍高度依赖于华为的昇腾、鲲鹏、网络和存储等 ICT 产品，因此目前 UB 硬件生态仍高度集中于华为自身，尚无生态伙伴提供第三方硬件。

UB 目前仍处于从实验室走向商用的关键阶段，其核心价值在于为 AI 训练提供了一种高性价比、高可靠性的专有网络方案。尽管尚未大规模落地，但华为在昇腾生态中的垂直整合能力可能加速其应用进程。未来 1–2 年将是观察其实际效能和商业化潜力的关键窗口期。

3.3 其他国产 GPU 生态

寒武纪、昆仑芯、摩尔线程、沐曦、壁仞、阿里平头哥等国产 GPU 厂商，随着国产算力产业的快速发展，在生态建设上也在参考英伟达模式进行投入。

首先在软件栈上，以寒武纪为例，寒武纪重点提供计算层的软件，即与 CUDA 对标的计算软件，同时也提供了兼容 CUDA 的解决方案。

在计算层之上的 AI 框架部分，寒武纪并未像华为提供 MindSpore 那样做 AI 框架的自研，而是以对接开源生态为主要方案，形成了寒武纪自身发展的差异化生态路线，各有优点所在。

在 ScaleUp 互联领域，国产厂商均积极投身与开放技术联盟中，包括 UALink、ESUN、OISA 等技术方案，借助联盟的力量共同打造互联生态。

4 异构GPU生态集成

由于多个国产厂商的崛起，以及大模型对算力的需求规模变化，异构 GPU 的生态集成变成了当前的重要需求之一。

理想的情况下，基于 UALink 等 ScaleUp 技术联盟实现异构 GPU 的集成，在算力、内存等方面实现完全的融合，可以构建出异构 GPU 组成的超节点系统。但目前上述工作仍有待于联盟各成员的共同推进，生态成熟仍待时日。

因此当前主要的生态集成模式，是采用 AI 云的方式，即基于云计算的调度能力实现异构 GPU 之间的算力调度，在云的层面实现异构集成。

以上述典型架构为例，每一类算力在微观上组成相对独立的集群，在集群内采用 ScaleUp 和 ScaleOut 技术进行互联；多个算力集群之间通过云网络进行连接，并且使用云平台进行算力的资源池化和调度。这是当前较为主流的生态集成模式，各大云厂商、垂直领域的 AI 云服务商基本上都采用思路相似的方案。

基于云实现的异构算力多集群，在大模型训练等场景时，如果算力需求超过单种算力计算的规模，则涉及单一任务跨异构集群运行的情况。目前少部分云厂商提出异构混训的集成方案，即通过集合通信库改造等方案，将训练任务进行拆解后分担到异构 GPU 上完成，目前在有限场景下已经取得一定的成果。但由于异构集成与单个 GPU 厂商的利益存在一定矛盾，而集成厂商改造集合通信库又离不开 GPU 厂商的支持，因此从生态角度看，上述方案的长期生态可持续性仍需观察。

总的来看，未来的异构 CPU 生态集成将两极化发展。一种是基于 AI 云的模型，进行松耦合的方式集成，这是目前已经可落地的模式，未来会在异构迁移、统一管理等技术点上进行重点发展。另一种是基于类超节点的紧耦合模式，依赖于各类 ScaleUp 互联技术和联盟的成熟和生态完善，仍需要一个较为长期的发展过程。

未来技术趋势与展望

超节点作为大规模异构计算（GPU/CPU/XPU 协同）的核心载体，依托本书前面介绍的各类 Scale-up 互联协议（NVLink、UALink、HSL、ETH-X、CXL、华为 UB、新华三 GLink 等）实现算力聚合与效率跃升，已成为支撑人工智能大模型训练、推理及高性能计算等数据密集型场景的关键基础设施。当前，超节点正逐步摆脱“单一算力堆叠”的发展模式，向“高效协同、灵活扩展、生态兼容”的高阶形态演进。结合现有技术实践与行业发展需求，对其发展趋势与未来进行展望和总结。

1 性能演进方向

超节点的性能演进核心在于“突破技术瓶颈、实现极致协同”，主要围绕带宽、时延、一致性、扩展性四大关键指标，通过互联协议升级与架构优化，持续提升算力利用效率，具体呈现如下四大演进方向。

带宽持续倍增，传输效率不断优化

带宽是制约超节点算力聚合效能的核心瓶颈，当前各类 Scale-up 协议均以高带宽及高效率为核心演进目标。其中，NVLink 作为 GPU 直连领域的标杆协议，GPU 互联带宽已从早期 V100 芯片的 300GB/s，提升至新一代 Rubin GPU 的 3.6TB/s 双向互联带宽，其构成的 NVL72 可提供 260TB/s 的节点内互联带宽；CXL 4.0 协议依托 PCIe 7.0 技术基础，将数据传输速率翻倍至 128GT/s，通过捆绑端口（Bundled Ports）实现多物理端口聚合，打破 16 Lane 传输限制，大幅提升设备间互联总带宽；ETH-X 协议针对超节点 10Tbps 级互联需求，通过头部压缩、事务聚合等技术优化，有效提升 Load 指令传输效率，缓解以太网包头开销带来的带宽损耗。未来，超节点内互联带宽将向单节点数百 TB/s 及更高的方向演进，同时通过协议层面的持续优化，不断降低传输开销，推动带宽利用效率的突破。

时延极致压缩，打破异构协同壁垒

低时延是超节点实现“算力协同如同单一设备”的关键前提，各类 Scale-up 协议通过缩短传输链路、优化软件栈设计、引入事务代理机制等多种方式，持续压缩端到端传输时延。CXL 协议的物理延迟较传统 PCIe 协议时延降低 50%，同时通过自定义极简软件栈，进一步压缩应用层时延，挖掘出远超物理链路时延的优化空间；NVLink 通过 GPU-XBAR 直连架构，实现 GPU 间 3.6TBps 双向通信的百纳秒级时延控制；ETH-X 协议引入 AXI 事务代理机制，针对 Push 类型事务实现本地快速响应，将跨 Scale-up 域的事务响应时延降低近一个数量级；UALink、华为 UB 等协议也以百纳秒级时延为优化目标，持续优化链路传输与事务处理流

程。未来超节点时延将向百纳秒甚至亚百纳秒级方向演进，同时通过计算与通信 Overlap 技术，实现通信时延的有效隐藏，提升算力与通信资源的综合利用效率。

一致性分级适配，平衡性能与开销

缓存一致性是超节点实现异构设备（CPU/XPU/GPU）协同工作的关键需求，不同 Scale-up 协议结合具体应用场景，形成“分级适配、灵活选择”的演进趋势。CXL 协议依托硬件 Cache 一致性技术，复用 PCIe 物理层实现异构设备协同，大幅简化编程模型；NVLink 通过物理地址寻址与硬件一致性检查机制，实现 GPU 间缓存数据的高效同步，支撑 NVL72 超节点作为单一巨型 GPU 的协同工作模式；ETH-X 协议支持释放一致性内存模型（Release Consistency Model），通过 Acquire-Release 语义实现线程间同步，在保证系统性能的前提下，有效平衡一致性开销。

因硬件一致性带来的复杂性和时延、成本开销，Scale up 协议也根据场景的需求，对是否支持硬件一致性进行不同的选择。未来，超节点可根据应用需求（如小规模设备直连用硬件保证一致性、大规模集群组网用软件保证一致性）动态切换一致性级别，在保障数据传输正确性的基础上，最大化降低一致性开销。

扩展性持续突破，支撑超大集群部署

超节点的规模演进正从“单节点多设备”向“多节点互联集群”延伸，各类 Scale-up 协议通过架构优化，持续突破规模限制。NVLink 通过 NVSwitch 构建 Fat-Tree 拓扑结构，支撑 72 个 GPU 组建单一 NVL72 超节点，多个超节点可通过 InfiniBand 协议进一步扩展为更大规模集群；ETH-X 协议支持直联与 Switch 互联两种拓扑结构，通过 High Radix 交换机与多层 Clos 拓扑，可支撑 512 卡规模的超节点，同时兼容 Ethernet 生态，实现超节点与 Scale-out 网络的无缝衔接；CXL 4.0 协议支持最多四个重定时器，有效延长链路传输距离，同时通过捆绑端口优化多主机环境下的扩展性，支撑多 CPU 与多设备的协同工作；华为 UB 协议以百万卡级互联为核心定位，打造直连、Scale-up、Scale-out 一体化协议架构，突破现有协议的规模限制；UALink 则通过多厂商联盟协作模式，支撑千卡级加速器的互联需求。未来，超节点将可能实现“单超节点千卡级、集群百万卡级”的规模目标，同时保持低时延与高带宽的协同优势，为万亿参数级超大模型的分布式训练提供支撑。

2 技术创新热点

当前，超节点的技术创新围绕“协议融合、架构优化、语义增强、生态兼容”四大核心方向展开，结合互联协议升级与硬件设计创新，着力破解超节点协同效率不足、部署成本偏高、编程复杂度较高等行业痛点，形成了多个技术创新热点。

协议融合：打破 Scale-up 与 Scale-out 技术边界

传统超节点依赖 Scale-up 协议实现内部高速互联，通过 Scale-out 协议实现集群规模扩展，两者存在明显的技术边界，导致数据传输效率降低、部署成本上升。当前，“协议融合”已成为超节点技术创新一个核心方向，较多 Scale-up 协议兼容成熟的网络生态，实现 Scale-up 与 Scale-out 的无缝衔接。ETH-X 协议的核心创新点在于将 AXI 事务封装为以太网数据包，依托成熟的 Ethernet 生态，实现超节点内部 Scale-up 互联与外部 Scale-out 扩展的一体化，同时引入 L2 LLR（链路层重传）、CBFC（信用流控）等机制，弥补以太网传输可靠性不足的短板；海光 HSL 协议推出 HSLoE（HSL over Ethernet）解决方案，在保持 HSL

事务层不变的前提下，将事务封装至以太网中，充分复用现有网络基础设施，实现超节点与现有网络生态的兼容，同时支持原生 HSL 与 HSLoE 模式并存，可灵活适配不同部署场景；华为 UB 协议直接定位为“直连、Scale-up、Scale-out 统一协议”，无需进行协议转换，实现超节点内部高速协同与集群规模扩展的一体化；CXL 协议依托 PCIe 生态优势，实现 CPU 与异构设备的 Scale-up 直连，同时可通过 PCIe/CXL 网卡连接以太 Switch 扩展支持 Scale-out 集群。此外，SuperNIC 架构的引入，成为协议融合的重要载体，通过硬件加速技术屏蔽 Scale-up 与 Scale-out 的网络边界，简化网络迁移成本，提升数据传输效率，已成为超节点部署的重要创新方向。

架构创新：IO 芯粒化与异构协同优化

随着超节点异构协同（CPU/GPU/XPU/IO）需求的日益复杂，传统 IP 集成式设计模式存在工艺迭代缓慢、研发成本偏高、灵活性不足等问题，IO 芯粒化（IO Die）已成为超节点架构创新的核心方向。ETH-X、GLink 等协议提出基于 IO 芯粒的实现方案，将计算芯粒与 IO 芯粒解耦，通过 UCIe 接口实现互联，计算芯粒采用先进工艺追求极致性能，IO 芯粒采用成熟工艺降低研发与生产成本，同时支持 IO 芯粒数量的灵活配置，实现互联带宽的动态适配；海光 HSL 协议同样依托 IO 芯粒方案，实现计算芯粒与 IO 芯粒的独立演进，减少 IP 移植成本，缩短产品上市周期，同时支持多 IO 芯粒负载均衡，提升超节点运行可靠性；UCIe 协议的普及为 IO 芯粒化发展提供了重要支撑，其分层架构（物理层、D2D 适配器、协议层）可兼容 PCIe、CXL、AXI 等多种协议，实现不同芯粒间的无缝互联，已成为超节点异构协同的核心接口标准。

语义增强：同步与异步协同，适配多元应用场景

超节点的应用场景日益多元化，涵盖人工智能大模型训练、高性能计算、云计算等多个领域，不同场景对数据传输语义（同步/异步）的需求存在显著差异，语义增强已成为协议创新的重要方向，其核心目标为通过同步保障数据正确性，通过异步提升传输效率。ETH-X 协议针对不同计算通信 Overlap 模式，定义两种访存语义：Direct Copy（粗粒度、连续拷贝）依托专用拷贝引擎（SU-TMA），实现本地 HBM 到远端 HBM 的大批量数据拷贝，适配结构化数据传输场景；Direct Access（小粒度、不连续访问）支持计算引擎直接读写远端 HBM 数据，适配非结构化数据传输场景，同时通过内存屏障优化，提升跨卡同步操作性能；华为 UB 协议也支持同步和异步协同，通过灵活的编程接口与运行时调度机制，支持两种模式的动态切换与混合使用。系统可根据任务类型、网络负载与性能需求，选择最优的通信策略。

可靠性与运维优化：硬件级保障与智能化管控

随着超节点规模的扩大与负载的提升，系统可靠性与运维效率面临更高要求，当前技术创新重点集中在硬件级容错与智能化管控两大方面。ETH-X、ESUN 等基于以太物理层的 Scale up 协议，针对 Ethernet 传输可靠性不足的问题，引入多重可靠性保障机制：L2 LLR 链路层重试机制，可针对 FEC 错误或 CRC 失败实现数据包重传；CBFC 信用流控机制，通过信用分配与释放，确保接收方缓冲区资源充足，避免数据包丢弃；聚合引擎与头部压缩技术，在提升传输效率的同时，减少误码带来的影响；此外，还支持多级负载均衡与事务保序机制，避免数据传输错乱。NVSwitch 通过 hop-by-hop 错误检查、ECC 校验、SRAM 缓冲等硬件机制，保障数据传输可靠性。在运维管控方面，超节点正逐步引入智能化监控与调度机制，通过实时采集链路带宽、传输时延、设备负载等核心数据，动态调整负载均衡策略与资源分配方案；ETH-X、HSL 等协议均支持集群控制器与标准化运维接口，简化超节点的部署、配置与故障排查流程，降低运维成本。

开放生态：协议开源与多厂商协同发展

超节点的规模化发展离不开开放生态的支撑，传统封闭协议（如早期 NVLink）已难以适配多厂商异构协同需求，支持协议开放和多厂商协作已成为行业创新热点。海光 HSL 协议已正式开放，面向 GPU、IO、OS、OEM 等产业全链条，联动上下游合作伙伴共建计算生态，同时提供 PCIe 兼容性支持，降低厂商接入成本；CXL 协议由 CXL Consortium 主导，采用开放标准设计，支持多厂商设备互联，已成为异构计算互联领域的主流标准之一；ETH-X 协议由 ODCC 发布，依托开放数据中心生态，推动超节点互联协议的标准化与产业化，可适配不同厂商的 GPU、Switch 设备；UALink 通过多厂商联盟协作模式，打造开放的加速器互联协议，打破单一厂商的技术主导；华为 UB 协议目前虽未完全开放，但其定位为开放兼容，预留了多厂商设备接入接口。英伟达也开展了 NVLink Fusion 项目，对其 NVlink 生态进行了部分开放，开源和开放已成为超节点领域和 NVLink 生态抗衡的唯一路线。

3 行业发展展望

随着人工智能大模型、高性能计算、云计算等场景的持续升级，超节点作为核心算力载体，将迎来规模化部署、多元化适配、绿色化发展、生态化协同的发展新阶段，同时在发展过程中仍面临协议标准化、成本优化、安全保障等多重挑战，展望如下：

规模化部署加速，成为核心算力基础设施

人工智能大模型（尤其是万亿参数以上级别模型）的训练与推理，需要大规模 GPU 协同算力与高速互联技术的支撑，超节点正从高端小众向规模化部署方向演进。未来 3-5 年，单超节点规模将从当前的数十、数百卡级别，提升至千卡级，集群规模将突破百万卡级，可支撑更大规模的分布式训练与推理任务；NVLink、CXL、HSL、ETH-X 等协议将进一步普及，成为超节点互联的主流选择，其中 ETH-X、ESUN 等依托 Ethernet 生态优势，HSL、UB 依托国产生态优势，CXL 依托 PCIe 兼容性优势，将在不同应用场景形成差异化竞争格局；超节点将广泛应用于 AI 算力中心、超算中心、云计算数据中心，成为支撑数字经济高质量发展的核心算力基础设施，同时逐步向边缘算力中心延伸，适配边缘大规模异构计算需求（如自动驾驶、工业互联网等场景）。

场景多元化适配，定制化超节点成为发展趋势

不同行业场景对超节点的性能、规模、成本等需求存在显著差异，通用化超节点已难以满足各行业的极致性能与成本优化需求，定制化成为超节点行业的重要发展趋势。在人工智能大模型场景，核心需求聚焦于高带宽、低时延、大规模协同，可使用用 NVLink、ETH-X 等协议，构建 GPU 主导的超节点，优化计算与通信 Overlap 效率，提升大模型训练速度；在高性能计算（HPC）场景，重点关注 CPU/XPU 协同与缓存一致性，可采用 HSL、CXL、UALink 等协议，构建异构协同超节点，适配气象预报、量子计算等大规模科学计算需求；在云计算场景，注重灵活性、扩展性与成本控制，可使用 ETH-X、HSLoE 等协议，依托 Ethernet 生态实现超节点与现有云网络的无缝衔接，支持资源按需扩容与动态调度；在自主化场景，海光 HSL、华为 UB 等国产协议将加速成熟，结合国产 GPU、CPU、Switch 设备，构建自主可控的超节点生态，满足政务、金融等关键领域的算力需求。此外，超节点将向“模块化、可配置”方向发展，通过 IO 芯粒数量、协议类型、设备规模的灵活配置，适配不同场景的个性化算力需求。

绿色化与成本优化，平衡性能与性价比

超节点的规模化部署带来了高能耗、高成本等问题，绿色化发展与成本优化已成为行业发展的重要考量因素。在能耗优化方面，将通过协议优化与硬件设计创新，持续降低传输能耗：CXL 4.0、ETH-X 等协议采用低功耗物理层设计，优化链路唤醒与休眠机制，减少设备空闲状态下的能耗；IO 芯粒化方案通过工艺差异化选择，降低整体功耗，同时减少芯片面积，提升芯片产量，降低硬件生产成本；负载均衡与智能化调度机制，可有效避免资源闲置，提升算力利用效率，间接降低系统能耗。在成本优化方面，开放协议的普及将有效降低厂商研发成本，CXL、ETH-X、ESUN 等开放协议，允许厂商复用现有 IP 与生态资源，缩短产品上市周期；ETH-X、HSLoE 等协议对 Ethernet 生态的复用，可避免专用网络的重复建设，降低部署成本；模块化设计允许用户根据实际需求配置资源，避免过度投资，提升超节点的性价比。未来，超节点将实现“高性能、低能耗、低成本”的协同平衡，推动算力资源的普惠化发展。

生态协同深化，协议标准化与互联互通加速

超节点的异构协同需求，推动“协议标准化、设备互联互通”成为行业共识，生态协同将进一步深化。在协议标准化方面，CXL Consortium、ODCC 等行业组织将牵头推动 Scale-up 协议的标准化进程，明确事务层、链路层、物理层的接口规范，减少多厂商设备互联的兼容性问题； GLink、HSL、ETH-X 等协议将持续完善，推动国产超节点互联协议与国际标准的兼容对接，提升国产生态的行业影响力；UCIe 协议将成为 IO 芯粒互联的主流标准，推动计算芯粒、IO 芯粒、存储芯粒的标准化与模块化，加速超节点架构创新。在互联互通方面，多厂商设备将逐步支持 CXL、HSL、ETH-X、GLink 等主流开放协议，实现 GPU、CPU、Switch、存储设备的无缝互联，打破厂商技术壁垒；超节点将实现与 Scale-out 网络、云计算平台、人工智能框架的深度融合，简化应用部署与调度流程；开源生态将进一步完善，开源编程模型、开源调度工具、开源运维平台的普及，将降低超节点的应用门槛，吸引更多开发者参与生态建设。此外，产学研协同将进一步加强，高校、科研机构与企业将联合开展超节点核心技术研发，突破协议优化、架构创新、可靠性保障等关键技术，加速技术成果的产业化转化。

挑战与应对：破解核心瓶颈，推动可持续发展

超节点在发展过程中，仍面临四大核心挑战，需通过技术创新与行业协同逐步破解。一是协议兼容性不足，不同 Scale-up 协议（如 NVLink、HSL、ETH-X、GLink、UB）的接口规范存在显著差异，多协议共存导致设备互联互通难度增加，未来需通过行业标准化建设与协议转换技术，破解兼容性瓶颈；二是核心技术仍有短板，国产超节点协议（如海光 HSL、华为 UB、H3C GLink）在带宽、时延优化等方面，与国际主流协议（如 NVLink、CXL）仍存在差距，需加大核心技术研发投入，突破技术瓶颈，提升国产超节点的核心竞争力；三是运维复杂度较高，超节点规模扩大与异构设备增多，导致系统调度、故障排查、性能优化的难度显著提升，未来需通过智能化运维技术（如人工智能调度、故障预测预警），提升运维效率；四是安全风险凸显，超节点作为核心算力载体，承载大量敏感数据，数据传输与存储的安全性面临严峻挑战，需加强硬件级安全设计与软件级安全防护，构建全方位安全防护体系，保障超节点的安全稳定运行。

总体而言，超节点作为大规模异构计算的核心载体，依托 Scale-up 协议的持续升级与技术创新，将逐步突破性能、规模、成本、生态等多重瓶颈，向“高效协同、灵活扩展、绿色低碳、开放兼容”的方向持续演进，成为支撑人工智能、高性能计算、云计算等产业高质量发展的核心算力基础设施，推动数字经济进入算力驱动的新阶段。

附录-关键词

术语	中文名称	解释/定义
AEC（Active Electrical Cable）	有源铜缆	在铜缆中集成Redriver/Retimer的有源高速互连方案，兼具铜缆的低成本与光缆的延伸能力，是高速短距互连的重要选项，也称为Active DAC。
AER	高级错误报告	Advanced Error Reporting，即高级错误报告，是一种用于详细报告硬件或软件错误信息的机制，便于问题定位和排查。
AFH：AI Forwarding/Fabric Header	AI转发头	SUE定义的以太网传输用的头结构，AFH gen2实现头部的深度压缩，用于优化AI场景下的以太网数据传输效率。
AI	人工智能	Artificial Intelligence，智能计算与大模型技术的核心领域，是研究、开发用于模拟、延伸和扩展人的智能的理论、方法、技术及应用系统的一门新技术科学。
AIGC	生成式人工智能	Artificial Intelligence Generated Content，即生成式人工智能，是指利用人工智能技术自动生成文本、图像、音频、视频等内容的技术。
ALS	阿里整机柜超节点架构	ALink System，阿里基于UALink推出的整机柜超节点架构，用于提升数据中心的计算效率和资源利用率。
ALU	算术逻辑单元	Arithmetic Logic Unit，即算术逻辑单元，是计算机中央处理器（CPU）的核心组件之一，负责执行算术运算（如加、减、乘、除）和逻辑运算（如与、或、非）。
APB (Advanced Peripheral Bus)	高级外设总线	针对外设的慢速标准协议，是一种用于连接微控制器和外设的低带宽、低功耗总线协议，适用于对传输速度要求不高的外设。
API	应用程序编程接口	Application Programming Interface，在 Agent 任务中涉及的调度对象，是不同软件组件之间进行交互的接口，规定了如何调用软件功能的方法和规范。
ATS	自动转换开关	Automatic Transfer Switch，主要用于在主电源与备用电源之间进行自动切换，以保证关键负载的连续运行，提高供电可靠性。

术语	中文名称	解释/定义
AXI	高级可扩展接口	Advanced eXtensible Interface，是一种高性能、高带宽、低延迟的片上总线协议，主要用于连接芯片内部的高速组件，如CPU、GPU、内存控制器等。
BAR	基地址寄存器	Base Address Register，即基地址寄存器，用于存储设备在内存映射空间中的基地址，CPU通过该寄存器访问设备的内存或I/O空间。
BDF 地址	BDF 地址	PCIe 设备的标识地址，包含 PCI 域号、总线号、设备号、功能号，NCCL 通过该地址识别 PCIe 设备的物理位置，便于设备间的通信和管理。
BER	误码率	Bit Error Rate，即误码率，是指在数据传输过程中出现错误的比特数与传输总比特数的比值，是衡量数据传输可靠性的重要指标。
BF16	浮点格式 16位	Brain Floating Point 16，截断尾数保留 8 位指数的浮点格式，大模型训练主流标准，兼顾了计算精度和存储效率。
BFC	基于缓冲感知的流量控制	Buffer-aware Flow Control，即基于缓冲感知的流量控制，是一种根据接收端缓冲区状态动态调整发送端数据发送速率的流量控制机制，避免缓冲区溢出。
BIOS	基础输入输出系统	Basic Input/Output System，即基础输入输出系统，是计算机启动时运行的固件程序，负责初始化硬件设备、引导操作系统启动。
BIST	内置自测试	Built-In Self Test，即内置自测试，是芯片或设备内部集成的测试电路，用于自动检测自身的硬件故障，提高测试效率和可靠性。
BMC	基板管理控制器	Baseboard Management Controller，即基板管理控制器，是一种独立于主CPU的嵌入式控制器，用于远程管理和监控服务器硬件状态。
BusBar	汇流排	机柜内提供集中供电的高压直流铜排，用于将电源分配到机柜内的各个设备，具有低电阻、高载流能力的特点。
C2C	芯片间互联	Chip-to-Chip，如 NVLink-C2C，实现芯片间统一内存寻址的高速互联技术，用于提升芯片之间的数据传输速度和效率。
CAS	比较并交换	Compare-and-Swap，即比较并交换，是一种原子操作，用于在多线程或分布式环境中实现数据的同步更新，避免并发冲突。
CBFC（Credit-Based Flow Control）	基于信用的流量控制	一种链路层的核心流控技术，用于解决物理链路拥塞与接收端缓冲区溢出问题，通过信用机制实现发送端与接收端的速率匹配。

术语	中文名称	解释/定义
CCA	集合通信加速报文头	Collective Communication Accelerate，即集合通信加速报文头，用于优化集合通信场景下的数据传输效率，减少通信延迟。
CCA Mode	集合通信加速模式	Collective Communications Acceleration Mode，即集合通信加速模式，是一种专门用于提升集合通信性能的工作模式。
CCL	集合通信库	Collective Communication Library，即集合通信库，是提供集合通信原语（如广播、归约、散射等）的软件库，用于支持分布式计算中的多节点通信。
CDR	时钟数据恢复电路	Clock and Data Recovery Circuit，即时钟数据恢复电路，用于从接收的数据流中提取时钟信号和恢复数据，确保数据的正确接收。
CDU	冷量分配单元	Cooling Distribution Unit，液冷系统中的关键组件，负责冷却液分配，将冷却系统产生的冷量均匀分配到各个发热设备，实现高效散热。
CFD	计算流体动力学	Computational Fluid Dynamics，热设计解决方案的一种理论设计原理，通过数值模拟方法分析流体流动和热量传递过程，用于优化设备的散热设计。
CLOS	CLOS互联拓扑	多级交换架构，提供无阻塞带宽，AI 集群主流网络拓扑，通过多级交换机实现大量节点之间的高速互联，满足AI集群的高带宽需求。
CM	连接管理器	RDMA 中用于在队列对（QP）建立之前，完成 QP 信息交换的组件，是 QP 建立的前提，确保RDMA通信的顺利建立。
Coherence Agent	一致性代理	集成在 NVLink 每个 GPU L2 缓存中的硬件单元，当 GPU 修改远端内存时，通过该单元发送 “无效化 / 更新” 信号，保证硬件层面的缓存一致性。
Collective Communication	集合通信	并行计算 / 分布式系统中多个进程 / 计算节点共同参与的通信操作，解决点到点通信在多节点场景下的编程复杂、性能低下等问题，是分布式训练的核心技术。
CollNet 算法	集合网络算法	NCCL 中基于 SHARP 协议设计的通信算法，专为 IB 网络优化，可将数据聚合任务卸载到网络交换机，提升大规模 GPU 集群的通信效率。
communicator	通信器	NCCL 中由多个 rank 组成的进程集合，定义了可以相互通信的进程范围，一个进程可属于多个通信器，且在不同通信器中 rank ID 可不同。
CoWoS	晶圆级芯片封装	Chip-on-Wafer-on-Substrate，台积电推出的 2.5D 先进封装技术，将芯片直接封装在晶圆上，再与基板连接，提升芯片的集成度和性能。

术语	中文名称	解释/定义
CPC（Co-packaged Copper）	共封装铜缆	将高性能铜缆直接集成到芯片封装附近的互连技术，缩短信号传输距离，降低信号损耗，提升传输速度。
CPLD	复杂可编程逻辑器件	Complex Programmable Logic Device，即可编程逻辑器件的一种，具有较高的集成度和灵活性，可通过编程实现特定的逻辑功能。
CPO（Co-packaged Optics）	共封装光学	将光学引擎（光模块）与 ASIC 芯片（如 GPU、TPU、交换机芯片）共同集成在同一封装基板或中介层（Interposer）上的高速系统的互连架构，提升光互连的效率和集成度。
CPU	中央处理器	Central Processing Unit，传统计算核心，在超节点中角色转向协同者，是计算机系统的核心，负责执行指令、处理数据。
CQ	完成队列	RDMA 中用于存放发送和接收请求处理完成通知的队列，无论传输成功或失败，都会产生完成消息（CQE）并存入该队列，供应用程序查询。
CQE	完成队列元素	完成队列（CQ）中存放的完成消息，包含传输状态等信息，每个工作请求（WR）处理完成后都会生成对应的 CQE，用于告知应用程序请求的处理结果。
CRC	循环冗余校验	Cyclic Redundancy Check，即循环冗余校验，是一种用于检测数据传输过程中错误的校验方法，通过计算数据的校验值来判断数据是否被篡改或出错。
Credit	信用	基于信用的流量控制机制中的核心计量令牌，本质是接收端向发送端授予的缓冲区资源使用权限，用于量化接收端当前可接收数据的存储能力。
CTLE	接收端连续时间线性均衡	Continuous Time Linear Equalizer，即接收端连续时间线性均衡器，用于补偿高速数据传输过程中的信号衰减和失真，提升接收信号的质量。
CUDA	（英伟达）统一计算架构	Compute Unified Device Architecture，NVIDIA 的并行计算平台和编程模型，用于开发利用GPU进行并行计算的应用程序。
CXL	计算快速链路	Compute eXpress Link，基于 PCIe 的缓存一致性互联标准，由 Intel 牵头联合多家厂商推出的高速互联协议，核心定位是解决 CPU 与 GPU、FPGA 等加速器间的互联瓶颈，支持缓存一致性。
DA（Destination Address）	目的地址	ETH-X 传输层定义的逻辑目的地址，用于标识数据传输的目标节点，确保数据准确送达。
DAC	直连铜缆	Direct Attach Cable，用于短距离高速互联的无源铜缆，电接口的高速无源铜缆组件，用于短距离（通常≤7米）设备间高速互联，无需额外的信号放大设备。

术语	中文名称	解释/定义
DCB	数据中心桥接	为数据中心网络设计的技术标准，包含 ETS、PFC 等子技术，支撑以太网实现无损传输和 QoS 保障，提升数据中心网络的可靠性和性能。
DHCP	动态主机配置协议	Dynamic Host Configuration Protocol，即动态主机配置协议，用于自动为网络中的设备分配IP地址、子网掩码、网关等网络配置信息。
DL	数据层	Data Layer，即数据层，是网络协议栈或芯片架构中的一个层次，负责数据的封装、传输和接收。
DLR	数据层重传	Data Layer Retransmission，即数据层重传，是数据层用于确保数据可靠传输的机制，当数据传输失败时，重新发送数据。
DMA	直接内存访问	Direct Memory Access，即直接内存访问，是一种允许外设直接访问系统内存的技术，无需CPU干预，提升数据传输效率。
DMMA	动态组播加速	Dynamic MoE Multicast Acceleration，即动态组播加速，用于提升MoE（混合专家模型）场景下的组播通信效率。
DMR	动态组播请求	Dynamic Multicast Request，即动态组播请求，是用于发起动态组播通信的请求信号。
DMTF	分布式管理任务组	Distributed Management Task Force，即分布式管理任务组，是一个致力于制定分布式系统管理标准的行业组织。
DNS	域名系统	Domain Name System，即域名系统，用于将域名（如 www.example.com）转换为IP地址，实现网络设备的定位和访问。
DP	数据并行	Data Parallelism，一种 AI 模型并行策略，将训练数据分成多个部分，分配到不同的计算节点上并行训练，提升训练速度。
DPC	下行端口遏制	Downstream Port Containment，一种 PCIe/总线可靠性机制，用于隔离下行端口的故障，避免故障扩散到整个系统。
DSA	领域专用架构	Domain Specific Architecture，专为特定领域（如 AI）定制的硬件架构，具有更高的性能和能效比。
DSCP	差分服务代码点	RoCE 中替代 IB 的 TRAFFIC CLASS 的技术，用于对流量进行分类，实现服务质量（QoS）管控，确保高优先级流量的传输质量。
ECC	纠错码	Error Correction Code，内存或传输中的错误检查与纠正机制，能够检测并纠正数据传输或存储过程中出现的错误，提升数据可靠性。

术语	中文名称	解释/定义
ECMP	等价多径路由	Equal-Cost Multi-Path，即等价多径路由，是一种路由技术，当存在多条到达同一目标的等成本路径时，将流量分散到多条路径上传输，提升网络带宽和可靠性。
ECN	显式拥塞通知	RoCE 网络层采用的拥塞控制技术，用于实现端到端的拥塞通知，让发送端及时调整发送速率，避免网络拥塞加剧。
ECPU	嵌入式中央处理器	Embedded Central Processing Unit，即嵌入式中央处理器，是专门为嵌入式系统设计的CPU，具有低功耗、小体积、高可靠性的特点。
EP	专家并行	Expert Parallelism，一种 AI 模型并行策略（常用于 MoE），将模型中的不同专家分配到不同的计算节点上并行计算，提升模型训练和推理效率。
ESUN	以太Scale-Up网络工作组	Ethernet Scale-Up Network，由 Meta、博通等组建的工作组，基于以太网的互联标准，致力于推动以太网在大规模计算场景中的应用。
ETS	增强型传输选择	DCB 标准的一部分，IEEE 802.1Qaz 规范定义，将流量分配到不同队列并为每个队列分配带宽权重，保证高优先级流量的带宽资源。
EXP Mode	极致访存模式	Extreme Performance Mode，即极致访存模式，是一种用于提升内存访问性能的工作模式，优化内存读写效率。
FEC	前向纠错	Forward Error Correction，即前向纠错，是一种在数据传输过程中自动检测并纠正错误的技术，无需接收端反馈请求重传，提升传输效率和可靠性。
Fence	内存栅栏操作	软件层面保证内存读写顺序的操作，可避免乱序导致的 Cache 不一致，保障数据操作的完整性，常用于多线程编程中。
FFE	发送端前馈均衡器	Feed Forward Equalizer，即发送端前馈均衡器，用于补偿高速数据传输过程中的信号衰减，提升发送信号的质量。
FIB	转发表信息库	Forwarding Information Base，即转发表信息库，是路由器或交换机中用于存储路由信息和转发规则的数据库，指导数据报文的转发。
Flit	流控单元	定长的数据片，配合 Credit 流控机制管控缓冲区，可有效降低拥塞导致的延迟抖动，是实现低抖动的关键技术之一；高速总线链路层的固定长度基本传输单元，用于替代传统可变长度帧 / 包，适配高速传输的流控与信号完整性需求。

术语	中文名称	解释/定义
FLOPS	每秒浮点运算次数	Floating-point Operations Per Second，衡量计算机性能的标准，用于表示计算机每秒能够执行的浮点运算次数，数值越高性能越强。
FP16	半精度浮点数	Half-precision floating-point，传统的 16 位浮点格式，用于在精度要求不高的场景下，减少内存占用和提升计算速度。
FP32	单精度浮点数	Single-precision floating-point，32 位浮点格式，科学计算基准，用于对精度要求较高的科学计算和工程应用。
FP4	4位浮点数	4-bit Floating Point，用于极致推理加速的低精度格式，在保证一定精度的前提下，大幅提升推理速度和降低内存占用。
FP8	8位浮点数	8-bit Floating Point，包含 E4M3 和 E5M2 两种格式，用于加速训练，兼顾精度和计算效率。
FTP	文件传输协议	File Transfer Protocol，即文件传输协议，用于在网络中实现文件的上传和下载，是互联网中常用的文件传输标准。
GBN	回退 N 步协议	Go Back N Frames Protocol，即回退N步协议，是一种自动重传请求（ARQ）协议，当接收端检测到数据错误时，要求发送端从错误帧开始重新发送后续所有帧。
GDDR	图形双倍数据速率存储器	Graphics Double Data Rate，传统的板载显存技术，专门为图形处理单元（GPU）设计，具有高带宽、高速度的特点。
GEMM	通用矩阵乘法	General Matrix Multiply，AI 计算中最核心的数学运算，是神经网络训练和推理过程中最主要的计算任务。
gNMI	gRPC 网络管理接口	gRPC Network Management Interface，即gRPC网络管理接口，基于gRPC协议的网络设备管理接口，用于实现网络设备的配置和监控。
GPGPU	通用图形处理器	General-Purpose Computing on GPU，强调通用计算能力的 GPU 架构，不仅用于图形处理，还可用于通用并行计算。
GPM-Req	GPU 投递型组播请求	GPU Posted Multicast Request，即GPU投递型组播请求，是GPU发起的一种无需等待响应的组播请求。
GPU	图形处理器	Graphics Processing Unit，AI 计算的核心产物，超节点的核心单元，最初用于图形处理，现广泛应用于AI训练和推理等并行计算场景。
GPUDirect P2P	GPU 直接点对点通信	英伟达的技术，让 GPU 能够直接访问彼此的显存，无需经由 CPU 系统内存中转，是 NCCL 节点内高速通信的基础。

术语	中文名称	解释/定义
GPUDirect RDMA	GPU 直接远程直接内存访问	英伟达的技术，让 GPU 内存可与 NIC 直接进行 RDMA 数据传输，无需主机内存中转，是 NCCL 节点间高速通信的核心技术。
GR-Req	GPU 归约请求	GPU Reduce Request，即GPU归约请求，是GPU发起的用于执行归约操作（如求和、求最大值）的请求。
GR-Resp	GPU 归约响应	GPU Reduce Response，即GPU归约响应，是对GPU归约请求的响应消息，包含归约操作的结果。
GUID	全局唯一标识符	Globally Unique Identifier，即全局唯一标识符，是一种用于唯一标识对象的字符串，确保在全球范围内不重复。
GVA	客户虚拟地址	跨交换机访问对端内存时发起访问使用的地址，需经本地 MMU 转换为网络物理地址，用于实现跨节点的内存访问。
HBD	高带宽域	High Bandwidth Domain，指超节点内部形成的高带宽互联区域；超节点中由节点内部处理器（GPU/CPU）通过 Scale up 网络互联形成的域，包含软硬件层面，硬件采用专用互连协议构建高带宽低延迟通信域，软件层面支持基于内存语义的直接内存访问。
HBM	高带宽内存	High Bandwidth Memory，通过 3D 堆叠实现极高带宽的存储介质，专门为高性能计算和AI场景设计，提供超高的内存带宽。
HPC	高性能计算	High Performance Computing，即高性能计算，是指利用高性能计算机系统解决大规模、复杂的计算问题，广泛应用于科学研究、工程设计、人工智能等领域。
HSL	海光超节点互联总线协议	Hygon System Link，海光超节点总线协议，用于海光超节点内部各组件之间的高速互联。
IB	英伟达私有以太互联协议	InfiniBand，一种高性能计算机网络通信标准，专用高速网络技术，支持全互联拓扑、高带宽、低延迟，硬件级支持 RDMA，拥有独立的协议栈和专用硬件，内置完善的流量控制和拥塞控制机制。
ICI	芯片间互联接口协议	Inter-Chip Interconnect，Google TPU 的互联接口，用于Google TPU芯片之间的高速互联。
Infinity Fabric	AMD私有互联协议	AMD CPU间、CPU与GPU模组对接的高速互联总线，用于提升AMD处理器和GPU之间的通信效率。
INS Mode	智感访存模式	Intelligent Sensing Mode，即智感访存模式，是一种智能感知内存访问需求的工作模式，优化内存访问效率。
INT4	4位整数格式	4-bit Integer，一种量化格式，用于AI推理场景，通过降低数据精度来提升推理速度和降低内存占用。
IOD	I/O 芯粒	I/O Die，负责输入输出功能的独立芯粒，是芯片模块化设计中的重要组成部分，负责处理芯片的I/O接口功能。

术语	中文名称	解释/定义
IOMMU	输入/输出内存管理单元	Input/Output Memory Management Unit，即输入/输出内存管理单元，用于管理外设对系统内存的访问，实现地址转换和权限控制。
IPFIX	IP 流信息导出	IP Flow Information Export，即IP流信息导出，是一种用于收集和导出网络流量信息的协议，用于网络监控和分析。
IPMI	智能平台管理接口	Intelligent Platform Management Interface，即智能平台管理接口，用于远程管理和监控服务器硬件，独立于操作系统。
LABR	最小可用带宽比	Least Available Bandwidth Ratio，即最小可用带宽比，用于衡量网络中各链路的可用带宽情况，指导流量分配。
LACP	链路聚合控制协议	Link Aggregation Control Protocol，即链路聚合控制协议，用于将多个物理链路聚合为一个逻辑链路，提升链路带宽和可靠性。
LID	本地标识	InfiniBand 子网内节点的唯一标识，交换机基于 LID 实现子网内的包转发，确保数据在子网内准确传输。
LL 协议	低延迟协议	NCCL 的优化通信协议，针对小数据量传输优化，通过减少同步延迟提升性能，依赖 CUDA 8 字节原子存储操作，采用 4B Data+4B Flag 的传输形式。
LL128 协议	128 字节低延迟协议	NCCL 对 LL 协议的扩展优化版本，专为 NVLink 环境设计，以 128 字节为单位进行原子存储操作，在低延迟的同时实现更高的带宽效率，是 NVLink 环境的默认协议。
LLDP	链路层发现协议	Link Layer Discovery Protocol，即链路层发现协议，用于网络设备之间自动发现彼此的信息（如设备类型、端口信息等），便于网络管理。
LLM	大语言模型	Large Language Model，即大语言模型，是一种基于深度学习的大型语言模型，能够理解和生成人类语言，广泛应用于自然语言处理场景。
LLR	链路重传机制	Link Level Retry，链路层的数据重传机制，链路层的核心可靠性保障技术，用于解决 Scale Up 互联域（不可靠传输）中链路错误（如 FEC 错误、CRC 错误）导致的数据丢失问题。
Load/Store	加载 / 存储	计算机中基础的内存访问指令，内存语义下处理器可直接使用该指令访问远端设备的内存数据，简化编程模型。
LPO	线性驱动可插拔光学	一种省去传统光模块中DSP芯片、采用直驱线性电路的低成本低功耗光互连方案，用于降低光模块的成本和功耗，提升光互连的性价比。

术语	中文名称	解释/定义
LSB	最低有效位	Least Significant Bit，即最低有效位，是二进制数中最右边的一位，权重最小。
MCM	多芯片封装模组技术	Multi-Chip Module，将多个芯片集成在同一个封装内的技术，提升芯片的集成度，减少芯片间的连接距离，提升性能。
Memory Region	内存区域	RDMA 中提前注册到 RDMA 网卡的内存缓冲区，拥有对应的本地密钥（local key），是 RDMA 操作内存的基础
MEMS	微机电系统	Micro-Electro-Mechanical Systems，光电路交换机（OCS）的核心技术
MESI	MESI 协议	经典的硬件级缓存一致性协议，通过确认和修改 Cache Line 状态实现缓存一致性，存在多次 Cache 端交互，大规模场景下存在同步开销
MLAP	Maximum Length of Aggregated Packet	聚合包长上限
MMIO	内存映射 I/O	内存映射 I/O（Memory-Mapped I/O），一种将 I/O 设备地址映射到系统内存地址空间的技术，使处理器可通过访问内存的方式操作 I/O 设备
MMU	内存管理单元	负责地址转换的硬件单元，可实现 GVA、NPA、SPA 之间的地址转换，是实现全局内存访问的关键
MNSP	成员报文数量上限	成员报文数量上限（Maximum Number of Sub-packets），指一个聚合数据包中可包含的成员报文的最大数量
MoE	混合专家系统（混合专家模型）	混合专家系统（混合专家模型，Mixture of Experts），一种稀疏大模型架构，通过多个“专家”模块分工处理不同任务，提升模型效率和性能
MPI	消息传递接口	定义集合通信标准接口规范的协议，起源于高性能计算领域，为集合通信提供统一的接口标准
MPS	最大数据包尺寸	最大数据包尺寸（Maximum Payload Size），指一次数据传输中可携带的最大有效数据量
MQDU	最大已使用队列深度	最大已使用队列深度（Maximum Queue Depth Used），指在数据传输过程中，队列中被使用的最大深度值，反映队列的负载情况
MSB	最高有效位	最高有效位（Most Significant Bit），指一个二进制数中最左边的一位，权重最大，代表2的（位数-1）次方
MSHD	最大单跳时延	最大单跳时延（Maximum Single-hop Delay），指数据在网络中经过单个节点跳转时的最大延迟时间

术语	中文名称	解释/定义
MTU	最大传输单元	最大传输单元（Maximum Transmission Unit），指数据链路层中可传输的最大数据包大小（不包含链路层头部），超过该尺寸的数据包需分片传输
NCCL	英伟达集合通信库	NVIDIA 推出的针对 GPU 和网络优化的多 GPU、多节点通信库，实现了高性能的集合通信和点对点通信原语，支撑分布式 GPU 计算
ncclComm	NCCL 通信器对象	NCCL 中参与通信的每个 GPU 维护的通信器对象，是调用 NCCL 所有通信操作的执行载体，需先初始化并定义参与通信的 GPU 集合
NCSI	网络控制器侧带接口	网络控制器侧带接口（Network Controller Sideband Interface），用于在主处理器与网络控制器之间传输管理和控制信息的辅助接口
NIC	网卡	网卡（Network Interface Card），又称网络适配器，是计算机与网络之间进行数据交换的硬件设备，负责将计算机数据转换为网络可传输的信号
NoC	片上网络	片上网络（Network on Chip），是在芯片内部实现多个处理器核、内存控制器等模块之间通信的互连架构，替代传统总线架构提升通信效率
NPA	网络物理地址	客户虚拟地址经本地 MMU 转换后的地址，在对端 MMU 中可进一步转换为本地系统物理地址
NPM-Req	非投递型组播操作	非投递型组播操作（Non-Posted Multicast Request），一种组播操作类型，发起方需要等待操作完成的确认信号后才能继续后续操作
NPM-Resp	非投递型组播响应	非投递型组播响应（Non-Posted Multicast Response），是对非投递型组播操作的响应信号，用于告知发起方操作已完成
NPO	近封装光学	近封装光学（Near packaged optics），一种将光模块与芯片封装靠近放置的光互连技术，可减少信号损耗，提升传输带宽和效率
NPU	神经网络处理器	神经网络处理器（Neural Processing Unit），专为神经网络加速设计的 DSA 芯片，具备高并行度、高能效比，适用于深度学习任务的快速运算
NSP	成员报文数量	成员报文数量（Number of Sub-packets），指一个聚合数据包中实际包含的成员报文数量
NTP	网络时间协议服务器	网络时间协议服务器（Network Time Protocol Server），用于通过 NTP 协议为网络中的设备提供精确的时间同步服务

术语	中文名称	解释/定义
NUMA	非一致内存访问	非一致内存访问（Non-Uniform Memory Access），一种内存架构，处理器访问不同内存区域的延迟不同，靠近处理器的内存访问速度更快
NVLink	NVIDIA 高速互连总线（英伟达私有AI互联总线协议）	NVIDIA 开发的高速互连通信协议，采用点对点结构，用于实现 CPU-GPU 及 GPU-GPU 间的高效互联，提供超高带宽和缓存一致性，同时支持内存语义，是英伟达私有AI互联总线协议
NVLink Fusion	NVLink 融合技术	允许定制芯片（CPU/XPU）与 NVIDIA 的 NVLink scale-up 网络集成的技术，旨在实现半定制化的 AI 基础设施部署，核心是“有控制的开放”。
NVLink-C2C	NVLink 芯片到芯片互连	基于 NVLink 协议的、用于芯片到芯片极近距离高速互连的物理封装和连接技术。常用于构建“超级芯片”（如 Grace Hopper），提供统一缓存一致性内存空间。
NVSwitch	NVLink 交换机	一种专用交换机技术，用于连接多个 NVLink GPU 服务器，构建大型 Fabric 网络（NVLink 网络），解决多 GPU 间的高速通信带宽和效率问题。
OAM	开放加速器模块	开放加速器模块（Open Accelerator Module），一种标准化的加速器模块规范，用于简化加速器与服务器的集成和部署
OCP	开放计算项目	开放计算项目（Open Compute Project），致力于硬件开源和标准化的组织，推动数据中心、服务器等硬件的创新和普及
OCS	光电路交换机	光电路交换机（Optical Circuit Switch），Google TPU 集群使用的基于光路切换的交换设备，通过光信号直接切换实现数据传输，具备低延迟、高带宽的特点
ODCC	开放数据中心委员会	开放数据中心委员会（Open Data Center Committee），推动数据中心标准化的组织，致力于制定数据中心相关的技术标准和规范
ODLS	直接加载/存储	直接加载/存储（OISA Direct Load/Store），基于 OISA 架构的直接内存访问指令，可直接访问内存数据，简化编程流程
ODMA	OISA 直接内存访问	OISA Direct Memory Access，OISA 架构中的直接内存访问机制
OISA	全向智能感知高速互联架构	Omni-directional Intelligent Sensing Express Architecture（原Omnidirectional Intelligent Sensing Express Interconnect Architecture），中移动牵头的互联标准，即全向智感互联相关架构
OISA-SAI	OISA 交换芯片抽象接口	OISA Switch Abstract Interface，OISA 架构中交换芯片的抽象接口规范

术语	中文名称	解释/定义
OM	OISA 管理器	OISA Manager，负责管理 OISA 架构相关设备和协议的管理模块
OOB	带外	Out-of-Band，指不通过主数据通道进行的管理或通信方式
OPE	OISA 协议引擎	OISA Protocol Engine，负责处理 OISA 协议的核心引擎模块
ORW	开放机架(宽)	Open Rack Wide，一种机柜尺寸标准
OS	操作系统	Operating System，管理计算机硬件与软件资源的计算机程序
OSFP	八通道小型可插拔封装	Octal Small Form-factor Pluggable，一种高速光模块封装标准
P2P	点对点	Peer-to-Peer，指网络中节点之间直接进行通信，无需通过中间节点转发的模式
PAM4	四电平脉冲幅度调制	4 级脉冲幅度调制（Pulse Amplitude Modulation 4），RoCE 物理层采用的编码方式，可在相同的带宽下传输更多数据，支撑高速以太网传输；从 NVLink 4.0 开始使用的信号编码方式，相比传统的 NRZ 编码，能在相同的波特率下传输双倍的数据，实现带宽翻倍。
PCB	印制电路板	Printed Circuit Board，电子元器件的支撑体，用于连接各类电子元件
PCH	平台控制中枢	Platform Controller Hub，负责连接 CPU 与外设，管理 I/O 接口等功能的芯片
PCIe	外设组件互连标准	Peripheral Component Interconnect Express，通用的总线接口标准，计算机系统中核心的高速互联总线标准，用于连接 CPU 与外设（GPU、固态硬盘、网卡等）及各类加速器，实现数据高速传输。
PCS	物理编码子层	Physical Coding Sublayer，负责数据编码和解码的物理层子层
PDB	电源供电分配板	Power Distribution Board，一种将主电源高效率分配至多个子系统组件的电路板
PDU	电源配电单元	Power Distribution Unit，一种专门为机柜内电子设备（如服务器、交换机）供电的工业级插座，具备电流电压监测、过载保护及模块化设计功能。
PFC	基于优先级的流控	Priority-based Flow Control，IEEE 802.1Qbb 标准定义的技术，为 RoCE 实现无损以太网的核心，当交换机优先级队列缓冲区接近满载时，发送 PFC 帧通知上游暂停发送对应流量，避免丢包

术语	中文名称	解释/定义
PL	物理层	Physical Layer，网络协议栈的最底层，负责处理物理介质上的信号传输
PLL	锁相环	Phase Locked Loop，一种用于同步信号频率和相位的电路
PMA	物理介质连接层	Physical Medium Attachment，负责物理介质连接的物理层子层
PMBus	电源管理总线	Power Management Bus，用于管理电源设备的串行通信总线
PMD	物理介质相关层	Physical Media Dependent，与具体物理介质相关的物理层子层
PN	制造商部件号	Part Number，制造商为每个产品分配的唯一部件标识
POST	加电自检	Power-On Self-Test，计算机开机时自动进行的硬件检测程序
PP	流水线并行	Pipeline Parallelism，一种 AI 模型并行策略，将模型按层拆分到不同设备，并行执行不同层的计算
PRBS	伪随机二进制序列	Pseudorandom Binary Sequence，一种具有随机特性的二进制序列，常用于通信系统测试
PRI，Packet Rate Improvement	报文速率提升	通过压缩以太网报文头部，增加帧传送效率的技术
protocol data unit (PDU)	协议数据单元	SUE定义的协议封装单元名称
PSU	电源模块	Power Supply Unit（原Power Distribution Unit表述有误，修正为PSU标准定义），为电子设备提供稳定电源的模块
PUE	电源使用效率	Power Usage Effectiveness，衡量数据中心能效的指标，计算方式为数据中心总耗电量与IT设备耗电量的比值
PXI，Peer-to-Peer AXI	端到端的AXI总线	一种基于以太网的Scale up互联协议
QoS	服务质量	Quality of Service，事务层的核心功能之一，为不同优先级的事务分配带宽，保障高优先级事务的传输效率
QP	队列对	InfiniBand 传输层的核心，由发送队列（SQ）和接收队列（RQ）组成，是端到端消息隔离和可靠传输的基础，需在通信两端初始化
RAG	检索增强生成	Retrieval-Augmented Generation，大模型的一种应用模式，通过检索外部知识来增强生成内容的准确性
RAM	随机存取存储器	Random Access Memory，可随时读写数据的内存设备，断电后数据丢失

术语	中文名称	解释/定义
RAS	可靠性、可用性、可维护性	Reliability, Availability, Serviceability，衡量系统稳定性和可维护性的三大核心指标
RC	根复合体	Root Complex，PCIe 总线树的根节点，通常由 CPU 担任，负责发起和管理 PCIe 总线上的事务
R-CRC	里德-所罗门循环冗余校验	将RS编码和CRC编码融合的纠错技术，用于提升数据传输的可靠性
RDMA	远程直接内存访问	Remote Direct Memory Access，无需主机 CPU 参与数据路径，应用程序可直接访问远程计算机虚拟内存的技术，支持 Read/Write 单边操作和 Send/Recv 双边操作，实现零拷贝通信
Ring 算法	环形算法	NCCL 中实现多 GPU 通信的基础算法，构建环形网络，每个 GPU 仅与相邻两个邻居交换数据，适用于小规模 GPU 集群，设计简洁、对等性强
RoCE	基于融合以太网的 RDMA	RDMA over Converged Ethernet，基于以太网的 RDMA 技术，在以太网上承载 IB 协议，兼顾 InfiniBand 的性能与以太网的低成本，需无损以太网络支撑，分 RoCEv1 和 RoCEv2
RQ	接收队列	队列对（QP）的组成部分，用于存放待接收的工作请求（WR），是 RDMA 接收数据的核心队列
RR ARB（Round-Robin Arbiter）	轮询仲裁器	按固定顺序轮流服务每个请求源，每个源获得相同的服务机会
RTT	往返时间	Mirror Round Trip Time，数据从发送方发出到接收方确认接收的总时间
SA	交换 Tray 管理代理	Switch Management Agent，负责管理交换 Tray 设备的代理模块
SC	子计算单元	Sub Cores，处理器中的子计算模块，负责执行部分计算任务
SEL	系统事件日志	System Event Log，记录系统运行过程中各类事件的日志文件
SerDes	串行器/解串行器	Serializer/Deserializer，位于物理层的核心技术，将并行数据转换为串行数据进行高速传输，支持多速率自适应与极性/通道反转，用于减少电磁干扰（EMI）；高速数据传输的接口电路技术
SHARP	可扩展分层聚合归约协议	CollNet 算法的基础协议，支持将数据聚合 / 归约操作卸载到网络设备，减少 GPU 间直接通信需求
SHM	共享内存	NCCL 节点内通信的传输方式之一，当直接 P2P 通信不可用 / 性能不佳时，通过系统内存路由流量，利用 CPU 优化的 PCIe – 内存传输避免性能问题

术语	中文名称	解释/定义
Simple 协议	简单协议	NCCL 的基础通信协议，实现简单，适用于无需特殊优化的常规通信场景
Snoop	探听报文	CXL 等协议中实现缓存一致性时的交互报文，用于确认数据在各设备缓存中的状态，保证读取数据为最新
SoC（System on Chip）	片上系统	系统级芯片，将CPU、GPU、内存控制器等多种组件集成在单一芯片上的集成电路
SoIC	系统整合芯片	System on Integrated Chips，台积电的 3D 堆叠封装技术
SPA	系统物理地址	对端设备的本地物理地址，由网络物理地址经对端 MMU 转换得到，用于访问对端本地内存
SQ	发送队列	队列对（QP）的组成部分，用于存放待发送的工作请求（WR），是 RDMA 发送数据的核心队列
SR-Req	Switch 归约读请求	Switch Reduce Read Request，交换机发起的归约读操作请求
SR-Resp	Switch 归约读响应	Switch Reduce Read Response，对 Switch 归约读请求的响应报文
SSH	安全外壳协议	Secure Shell，一种用于远程登录和数据传输的安全协议
SUE-T	Scale-Up 以太网传输协议	Scale-Up Ethernet Transport，博通主导的基于以太网的互联协议
sysfs	系统文件系统	Linux 系统中的虚拟文件系统，NCCL 通过该文件系统获取 PCIe 设备层次结构、设备特性等拓扑信息
TL	事务层	Transaction Layer，PCIe 协议栈中的一层，负责处理事务请求和响应
TLP	事务层数据包	Transaction Layer Packet，PCIe 事务层中，设备核心的请求（存储器读写、I/O 操作等）被封装后的数据包，用于在事务层与数据链路层间传输
TOR	架顶交换机	Top of Rack，位于机柜顶部的交换机，用于连接机柜内的服务器设备
TP	张量并行	Tensor Parallelism，一种 AI 模型并行策略，将模型的张量维度拆分到不同设备，并行执行计算
TPU	张量处理单元	Tensor Processing Unit，Google 自研的 AI 专用芯片，专为加速张量计算设计
Tree 算法	树形算法	NCCL 中的通信优化算法，常用双二叉树算法，利用二叉树特性降低通信延迟（log2N < N），延迟性能优于 Ring 算法

术语	中文名称	解释/定义
UALink	一种AI高速互联协议标准	Ultra Accelerator Link，AMD、Intel 等发起的开放互联标准
UB	统一总线	Unified Bus，华为提出的互联架构概念
UB-Mesh	统一总线互联网络	Unified Bus Mesh，华为发布的分层局部化网络拓扑架构
UEFI	统一可扩展固件接口	Unified Extensible Firmware Interface，用于替代 BIOS 的固件接口标准
UMA	统一内存寻址	Uniform Memory Address，将不同内存区域映射到统一地址空间的寻址方式
UPS	不间断电源	Uninterruptible Power Supply，在电网断电时能为设备提供临时供电的设备
URMA	异步内存访问	超节点上层应用的内存访问方式之一，可实现无阻塞的远端内存访问操作
UUID	通用唯一识别码	Universally Unique Identifier，用于唯一标识信息的128位数字
UVA	统一虚拟地址	英伟达 NVLink 中实现的地址机制，将所有连接的 GPU 的本地内存、显存、CPU 内存映射到同一个虚拟地址空间，支持直接的远端内存访问
UVM	统一虚拟内存	统一虚拟内存技术，允许 GPU 和 CPU 共享统一的内存空间，使多个 GPU 可像访问本地内存一样访问远程 GPU 显存，减少数据拷贝开销。
VC	虚拟通道	Virtual Channel，总线物理层 / 链路层的逻辑隔离机制，在同一物理链路上划分的多个独立逻辑传输通道。
VLAN	虚拟局域网	Virtual Local Area Network，将一个物理的局域网在逻辑上划分成多个不同的广播域（子网）的技术
VRM	电压调节模块	Voltage Regulator Module，为芯片供电的模块，负责将输入电压转换为芯片所需的稳定电压
WR	工作请求	RDMA 中提交到队列对（QP）的操作请求，网卡硬件直接处理该请求，实现硬件卸载的零拷贝通信
WRR ARB（Weight ARB）	加权轮询仲裁器	在轮询基础上，为每个请求源分配权重Weight，权重越高，获得的服务机会越多

新华三集团 www.h3c.com

北京总部
北京市朝阳区广顺南大街8号院
利星行中心1号楼
邮编:100102

杭州总部
杭州市滨江区长河路466号
邮编:310052

客户服务热线
400-810-0504

Copyright ©2026 新华三集团 保留一切权利
免责声明：虽然新华三集团试图在本资料中提供准确的信息，但不保证本资料的内容不含有技术性误差或印刷性错误，
为此新华三集团对本资料中信息的准确性不承担任何责任。新华三集团保留在没有任何通知或提示的情况下对本资料的内容进行修改的权利。
CN-260501-IN-v1.0