arXiv:2603.03731v1 [cs.DC] 4 Mar 2026

HYPERPARALLEL：一种超节点亲和型人工智能框架*

Xin Zhang†、Beilei Sun‡、Teng Su‡、Qinghua Zhang‡、Chong Bao‡、Lei Chen†和Xuefeng Jin‡

†香港科技大学和香港科技大学（广州），中国香港和广州 ‡华为技术有限公司，中国

摘要

随着大规模、稀疏、多模态和智能体人工智能模型的兴起，硬件也逐渐转向超节点架构，将数百至数千个加速器与超低延迟互连和统一内存池集成在一起。然而，现有AI框架并非为高效利用这些架构而设计，导致编程复杂度高、负载不均衡以及内存利用率低。本文提出一种超节点亲和型人工智能框架，将超节点视为单一逻辑计算机，并将硬件感知编排融入框架之中。我们的HyperParallel架构基于MindSpore实现，包括用于自动分层内存管理的HyperOffload、用于异构工作负载间细粒度多程序多数据并行的HyperMPMD，以及用于声明式并行策略规范的HyperShard。这些技术共同显著提升了训练和推理效率，同时降低了并行编程和系统调优的开销，证明了超节点亲和对于下一代AI框架的必要性。

1 引言

人工智能的快速发展，得益于模型架构、训练范式与硬件系统的协同演进 [14, 15, 16]。现代基础模型在参数量 [6,7, 20] 和功能范围上持续扩展，从纯语言系统拓展到多模态 [22], 稀疏 [3, 5], 和智能体架构 [24]。这些模型带来了显著的计算负载异构性、不规则的数据路由，以及对内存和通信带宽的巨大需求。与此同时，硬件架构已演进为超节点集成集群，其中数千个加速器通过超低延迟的点对点互连和统一内存池相连接 [10, 11, 26]。在这一背景下，人工智能框架已成为关键的中间件层，负责连接高层算法抽象与日益复杂的硬件架构。

我们发现，现有 AI 框架面临性能与可编程性差距。传统的单程序多数据（SPMD）范式 [4]难以应对混合专家（MoE） [3, 5] 和多模态架构中固有的负载不均衡。此外，在分层内存层级中手动管理中间状态（权重、激活值和 KV缓存），给研究人员带来了难以承受的工程负担。当前的系统扩展层 [1, 2, 8, 9, 18, 19, 21], 虽然在特定领域有效，但往往导致模型逻辑与硬件拓扑紧密耦合，造成代码可移植性差、集群利用率低。

这些挑战表明，对现有框架进行增量优化是不够的。相反，AI框架必须经历一次架构转变，明确拥抱超节点的特性，并重新思考并行、内存与执行的抽象方式。在本文中，我们认为超节点亲和应成为下一代AI框架的核心设计原则。超节点亲和框架将超节点视为单一的、逻辑上统一的计算实体，并将硬件感知编排直接嵌入框架运行时与编译器之中。由此，它可以让算法开发者免受底层系统复杂性的影响，同时系统地释放现代硬件的性能潜力。

*完整的贡献者名单列于论文末尾的致谢部分。

基于这一洞见，我们提出了一种超节点亲和型人工智能框架，并介绍了其在MindSpore中的设计与实现。该框架引入了一种新颖的HyperParallel架构，整合了三个关键组件：HyperOffload，通过统一内存池与自动卸载将计算与模型状态解耦；HyperMPMD，将并行执行从单程序多数据扩展到细粒度的多程序多数据（MPMD），以应对异构和多任务工作负载；以及HyperShard，提供声明式并行编程接口，将算法逻辑与并行策略设计解耦。这些组件共同使混合专家、全模态和强化学习工作负载能够在超节点集群上实现灵活、高效的执行。

本文做出如下贡献：

- 新兴挑战的系统分析：我们分析了模型工作负载、硬件架构以及训练—推理范式的最新趋势，并指出它们在超节点上部署时对现有AI框架所施加的根本限制。
- 超节点亲和框架设计：我们提出一种新的架构范式，将超节点视为一台巨型计算机，并在框架内抽象复杂网络拓扑、分层内存系统和异构执行环境。
- HyperParallel架构：我们引入并实现了HyperOffload、HyperMPMD和HyperShard，从而实现统一状态管理、灵活的多程序多数据并行以及声明式并行编程。
- 实证验证：我们证明所提出的框架显著提升了训练与推理效率，同时降低了开发与优化开销，并在代表性大规模工作负载上验证了其有效性。

本文其余部分组织如下。第2节刻画模型与硬件的演进，回顾现有AI框架，并分析万亿参数大语言模型时代AI框架面临的挑战。第3节介绍所提出的超节点亲和框架的设计与关键技术。第4节以对结果与未来方向的讨论作结。

2 背景

人工智能框架是模型算法与底层硬件架构之间至关重要的中间件层，作为基础软件基础设施，推动着整个人工智能领域的发展。其演进由算法复杂度与硬件创新的双重需求所驱动。一方面，框架必须提供高级抽象，使研究人员能够高效地开发、调试和验证新型架构，从而加快算法迭代。另一方面，框架必须系统地将这些高级抽象转换为经过优化的可执行程序，以最大限度地利用硬件，并充分发挥专用计算资源的潜力。

2.1 小型深度学习模型时代

模型与硬件。在深度学习发展的早期阶段，大多数模型的规模都较小。例如，小型计算机视觉（CV）模型 [25] 通常可以装入单个加速器的内存中。参数量适中的自然语言处理（NLP）模型 [13]虽然超出单张显卡的容量，但通常可以装入配备多张图形处理器的单机中。

训练范式。训练小型深度学习模型时，计算机视觉模型的主流训练策略是数据并行（DP），单卡算力是主要的性能瓶颈。对于规模适中的自然语言处理模型，通常采用模型并行，将模型组件分片到同一节点内的多张卡上。节点内通信量较大，使内部带宽成为关键瓶颈。

AI 框架。 为了在单机上处理此类工作负载，PyTorch [16] 凭借高度灵活的执行能力，成为主流生态系统。PyTorch 的核心优势在于其动态图计算和强大的生态系统，使其非常适合快速原型开发、模型定制和学术研究。

2.2 十亿规模大语言模型时代

模型与硬件。 随着大语言模型的出现，模型参数迅速扩展至数千亿。庞大的预训练数据集也对硬件提出了更高要求。通常，在训练十亿规模的大语言模型时，会搭建包含多个节点的服务器集群。

训练范式。 预训练和监督微调是十亿规模大语言模型最常见的两种训练任务。此外，多维切分（数据并行、张量并行和流水线并行）策略也得到了广泛采用。不同的切分策略对集群带宽提出了不同要求。尤其是张量并行（TP）、流水线并行（PP）和上下文并行（CP）需要频繁进行跨服务器通信——每次传输的数据量通常超过数 GB——这很难通过计算通信重叠来隐藏。例如，在典型的训练设置中，TP 的数据流量开销占训练时间的 52.9% [11]。因此，跨机器通信成为主要的性能瓶颈，导致集群资源利用率不佳。

人工智能框架。 PyTorch 的主要局限包括缺乏全面的多程序多数据（MPMD）能力，以及原生不支持计算与存储解耦。为了解决这些瓶颈，许多框架应运而生，例如 JAX [2], 、DeepSpeed [18], 、Megatron [21], 和 HLX [9], ，它们作为专门的系统扩展层而开发。这些框架专注于分布式策略、内存优化、并行拓扑、混合注意力兼容性以及高效的多模态处理。

- **JAX：函数式编程与编译器驱动加速** – JAX 采用函数式编程范式，并结合可组合变换（即时编译、并行映射、向量化映射），使编译器能够对算子、并行和求导执行统一优化。它擅长大规模 TPU 训练、科学计算，以及需要精确控制计算图变换的研究工作。然而，其生态系统高度专业化，导致编程复杂度较高。
- **DeepSpeed：** 系统工具箱 **用于大规模训练** – DeepSpeed 提供 ZeRO 系列冗余消除切分、混合专家优化器、高效通信和卸载支持，是在 GPU资源受限的情况下训练超大规模模型的关键基础设施。它作为与 PyTorch 紧密耦合的系统增强层，用于弥补内存效率和分布式调度方面的不足。其一个显著缺点是默认网络拓扑相对均匀；而在实际的超节点中，机架间延迟和带宽可能存在巨大差异，DeepSpeed 对单程序多数据和静态切分的依赖可能导致严重的负载不均衡和硬件利用不足。

- **Megatron：面向稠密与稀疏模型的极致并行**——Megatron强调对流水线并行、张量并行和混合专家并行的优化，以提供峰值分布式性能。其“规模优先”的设计理念非常适合拥有充足GPU资源、追求最大吞吐量的环境。尽管Megatron-Core正逐步演进为大语言模型训练的基础模块，但该框架存在通信逻辑与硬件拓扑之间的紧耦合。这导致代码可移植性差、编程复杂度高，以及参数调优和性能优化成本过高。

- **HLX：面向混合注意力架构的加速**——HLX专为采用线性注意力与全局混合注意力的新型Transformer变体而设计。它引入多线程并行编排以及双维度（行/列）计算流水线，以加速稀疏与线性注意力算子。通过将算子结构与线程调度深度集成，HLX为混合注意力模型提供了系统级加速方案。然而，它缺乏面向超节点的框架级统一内存池管理，迫使研究人员手动管理中间训练和推理状态。

相关研究工作。 除了上述框架，还有一些研究工作致力于解决大语言模型预训练整个生命周期中的其他瓶颈。

- **集群级容错与算法—系统协同设计** – MegaScale [8] 代表了集群级训练基础设施的发展趋势。在跨多个节点、拥有海量分布式计算能力的环境中，它通过算法／内核优化、三维并行通信重叠和自动化故障监控，在万卡集群上实现了较高的模型浮点运算利用率（MFU）。然而，要实现有效的计算通信重叠，需要高度定制化的并行策略和代码实现，导致调试和迁移成本高得令人难以承受。

- **面向中大规模训练的分片与卸载**——以*ZeRO* 生态系统（具体为ZeRO-3和ZeRO-Offload） [17, 19], 为例，该方向针对参数规模从数百亿到数万亿的模型训练。其核心方法是对参数、梯度和

优化器状态在GPU集群间进行完全分片，并常与CPU/NVMe卸载相结合，以突破GPU显存墙。这些方法利用高效的通信划分实现近线性扩展。然而，它们依赖静态内存划分，且缺乏对统一内存池的自动化管理能力，可能导致内存碎片。

• **超大规模加速器集群的调度与通用执行层**– *Pathways* [1]展示了超大规模集群采用统一执行层的必要性。该方向着重于加速器Pod之间的异步数据流、调度和拓扑优化，以实现流水线并行、数据并行和模型并行之间的无缝协作。与传统僵化的 SPMD 模型不同，它支持精细且不规则的路由，这对于混合专家模型至关重要。然而，在异构节点上管理高度动态、大规模的任务图时，依赖集中式资源管理器和协调器可能会带来调度开销和延迟瓶颈。

2.3 万亿级参数大语言模型时代

模型。近期，我们观察到主流大语言模型在规模、架构和目标模态方面发生了重大变化。具体而言，最先进的大语言模型通常拥有万亿级参数，并采用混合专家（MoE）架构，在扩大参数规模时提高稀疏性。除文本之外，对多种模态的支持也日益普遍。

• 从模型规模和架构的角度来看，许多最先进的大语言模型（例如 DeepSeek- V3 [12] 和 Qwen-3 [23], ）采用混合专家（MoE）架构，在扩展参数规模时提高稀疏性。这种方法能够扩大模型容量并增强泛化能力，同时保持每个 token 的计算成本不变。

• 从纯语言模型向全模态系统的转变已成为新发布模型的主流趋势。这种转变要求整合多种子模块，使模型结构日益不规则，令并行编程和优化都更加复杂。全模态模型通常采用多编码器、模态融合层和多解码器架构。对多种模态的整合引入了模块级异构性和动态路由负载。

训练范式。 训练范式正从传统预训练转向智能体强化学习（RL）。这一转变要求协同部署多种任务，包括训练、推理和智能体执行。异构化趋势对多任务、多模型工作流的编排提出了更严格的要求。具体而言，为了增强指令遵循、深度推理和复杂问题解决能力，最先进的大模型在初始预训练阶段后越来越多地引入强化学习。强化学习将工作负载范式从基于静态数据集的纯训练，转变为涉及动态数据的异构“采样—评估—更新”任务。此类工作流通常依赖大规模 rollout 操作和异步演员—学习者架构，其中训练和推理任务会并发执行。因此，底层框架必须支持稳健的异构任务管理、异步任务调度，以及适用于多任务、多模型环境的动态编排能力。

硬件特性。超节点集群专为训练具有混合专家、多模态和智能体特征的大语言模型而设计。以华为矩阵 384 [26] (Atlas 900)超节点为例，我们分析超节点集群的典型特征：

• **统一内存寻址与点对点架构**– 基于UB互联协议（灵衢），该系统将384个昇腾910C NPU和192个鲲鹏CPU整合为一个超节点。计算、内存和网络资源解耦为独立的资源池单元。

• **极致带宽与超低时延**– 传统架构依赖PCIe或以太网进行互联。昇腾384超节点采用高效协议，与传统服务器架构相比，通信带宽提升15倍，同时将单跳时延从 2μs降低至200ns，性能提升十倍。

• **分层多样化网络拓扑**– 单个昇腾超节点的规模预计将从 384 张卡扩展到 8,192 张卡，未来目标则是达到 15,488 张卡。为支持这种大规模、可扩展的互联，超节点采用分层拓扑：在每个机架内构建二维全互连，并将其扩展为跨机架的另一个二维全互连，最终形成四维全互连拓扑。

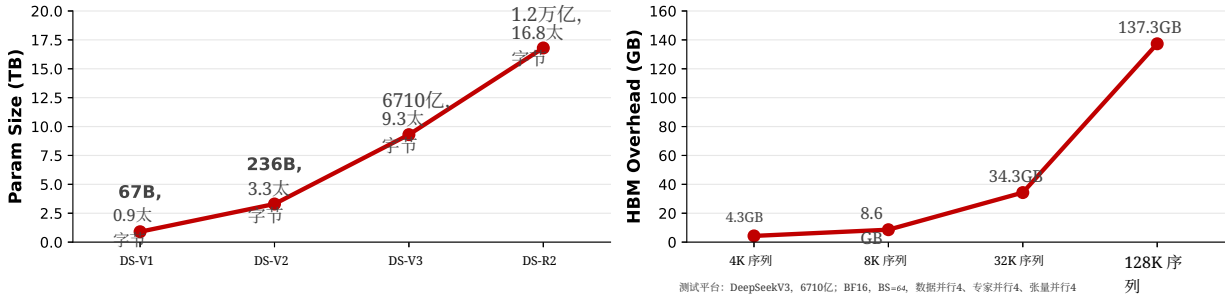


图1：在模型训练和推理过程中，存储和管理参数及中间状态的复杂性持续增加。

AI框架面临的挑战。虽然超节点为快速增长的模型参数和日益复杂的架构提供了卓越的解决方案，但要充分发挥其潜力，还需要应对若干根本性挑战。为降低算法开发和调优的复杂度，同时最大限度地提升大规模训练和推理效率，AI框架面临以下关键挑战：

表1：各模型采用的策略。

模型与算法	策略
稠密 Transformer	数据并行、流水线并行、张量并行、序列并行
稀疏混合专家	数据并行、流水线并行、张量并行、序列并行、专家并行
扩散	数据并行、全分片数据并行
长序列	序列并行，上下文并行
RL	MPMD
...	...

表 2：各集群的策略。

集群	策略
单机（8 个裸片）	TP8，其余部分采用流水线并行
单机（16 个裸片） 单机	高维张量并行（TP16），减少流水线并行
8k 节点超平面	拓扑感知的 TP16，减少流水线并行

• **挑战 1：抽象复杂网络拓扑，简化并行编程** – 随着超节点消除了节点间带宽瓶颈，集群规模可扩展至 100,000 张卡。要驾驭如此复杂的网络拓扑，需要精心设计并优化并行策略。正如表 1 和表 2 所示，模型架构或集群配置的任何变更，都需要彻底重新设计这些策略。根据经验，每轮适配和调优都需要资深系统优化工程师投入 1–2 周，并消耗大量集群资源，导致运营效率低下。

• **挑战 2：实现灵活、细粒度的并行，以最大限度地提高硬件利用率** – 当前的并行分片主要依赖单程序多数据（SPMD）范式。然而，随着模型向混合专家和全模态架构演进，不同模块和层呈现出异构计算负载。标准 SPMD 方法经常导致负载不均衡；例如，在子模块负载各异的全模态模型中，SPMD 与流水线并行（PP）结合往往会产生大量流水线气泡。此外，混合专家模型还面临较高的通信开销；在 DeepSeek-V3 中，专家并行（EP）通信占执行时间的 17%，通信掩蔽率仅为 61%，远低于 90% 的理想目标。

• **挑战 3：通过内存池化管理中间状态，以优化内存利用率**– 随着模型规模从数千亿参数扩展到数十万亿参数，管理包括权重、激活值和 KV缓存在内的中间状态的复杂性呈指数级增长。超节点通过内存语义互连提供统一的 DRAM内存池。为维持性能，必须在使用前主动将中间状态预取（换入）到高带宽内存（HBM）中，以避免内存访问延迟。手动管理这些换入换出操作会给算法研究人员带来巨大的编程和优化负担。

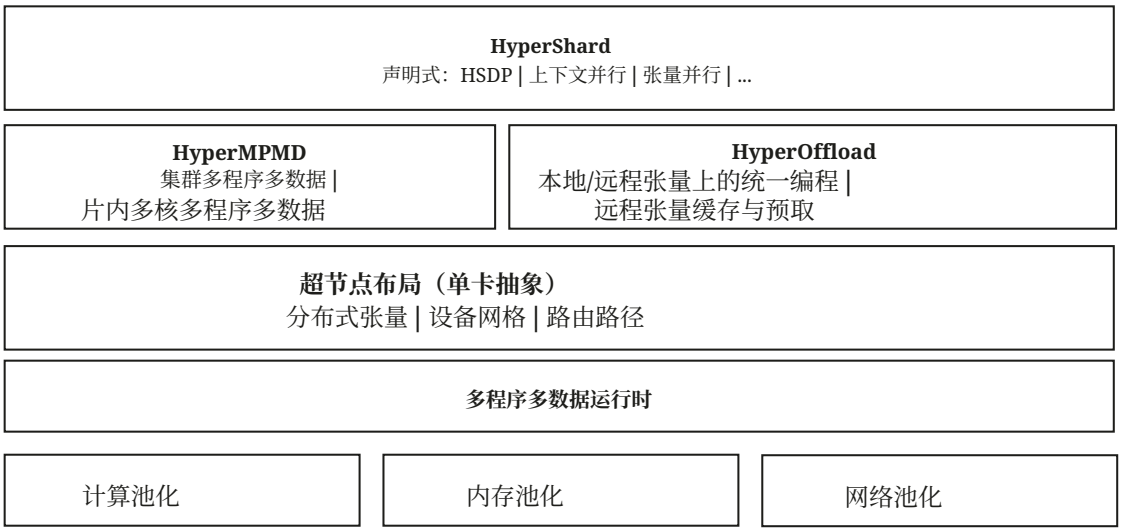


图 2：*Hyperparallel* 整体设计。

3 超节点亲和型人工智能框架的设计

当前工业界训练和推理软件中的优化工作，尚不足以从根本上解决超节点给人工智能框架带来的系统性挑战。为了充分发挥超节点的优势，并解决模型演进带来的瓶颈，人工智能框架必须围绕超节点亲和进行架构重设计。通过融合超节点特有的架构特征，框架能够为研究人员带来变革性的编程体验，同时显著提升训练和推理效率。本章结合 MindSpore 的实际实现，介绍超节点亲和型人工智能框架的架构设计与关键技术。

3.1 总体设计

MindSpore 超节点亲和设计的核心理念是通过人工智能框架抽象底层硬件复杂性，将超节点视为一台巨型计算机。这种方法为算法研究人员提供了类似单节点系统的编程与执行体验。MindSpore 将并行分片、训推状态管理等复杂的软件工程任务直接融入框架，借助超节点对等架构的优势，解决混合专家、全模态和强化学习等复杂场景中的性能瓶颈。为此，我们提出了如图 2 所示的 **HyperParallel** 架构，其中包含三个关键组件，旨在屏蔽硬件复杂性，同时最大限度地发挥架构优势：

- **HyperOffload** ——实现计算与状态的解耦。利用超节点的池化存储能力，解决模型规模不断扩大带来的内存瓶颈。
- **HyperMPMD** ——将并行分片从传统的单程序多数据（SPMD）范式转变为细粒度的多程序多数据（MPMD）方式。它利用对等计算架构，为强化学习和全模态工作负载提供灵活并行。
- **HyperShard** – 推动编程范式从命令式转向声明式。它抽象了超节点集群复杂的拓扑结构，为全模态及其他先进模型提供简洁且可组合的编程体验。

使用 MindSpore 在超节点上进行模型开发、训练和推理的操作流程如下：

- **步骤 1：算法开发** – 研究人员通过该框架开发算法，利用 Hy- perShard 的声明式编程接口来配置模型的并行策略。



图3：*HyperOffload*的架构。

• **步骤2：灵活并行**——HyperMPMD以多程序多数据格式执行灵活的细粒度并行，并专门针对算法及独特的集群拓扑进行优化。
• **步骤3：运行时编排**——在执行过程中，HyperOffload对计算、存储和网络进行全局编排。它管理高带宽内存（HBM）与DRAM之间的状态交换，以充分利用超节点的分层内存池。

3.2 HyperOffload

HyperOffload 的基本理念是将模型状态卸载到外部内存池，并将片上内存（HBM）用作高速缓存。该架构使单个加速器能够容纳规模大得多的模型。核心挑战在于自动完成HBM与DRAM之间的状态交换，同时不牺牲性能。为此，HyperOffload采用多级缓存流水线调度与整体图编排，如图3所示。

• **多级缓存流水线调度** – HyperOffload 利用通信隐藏技术，在计算算子请求之前，异步地将下一执行阶段所需的缓存块预取到高速存储层。通过结合模型结构特征与数据访问模式预测，系统可动态调整预取路径，从而有效地将加载延迟与计算时间重叠。

• **整体图编排与调度** – HyperOffload 将分层缓存操作（如数据预取和块卸载）抽象为原生算子，从而统一描述缓存读写和迁移行为。借助 MindSpore 的 JIT 图编译，该框架在解析用户网络后，会自动插入缓存管理算子并重新组织执行流程。通过对缓存、计算和通信算子进行统一编排，编译器可自动调度并行执行链，免去手动插入同步点的复杂操作。

通过充分利用超节点的统一大内存池，HyperOffload 显著缓解了训练和推理期间的 HBM 内存限制。这样便可以用简单的 1D-SPMD数据并行取代复杂的 ND-SPMD 并行，从而降低状态同步带来的通信开销，进一步提升性能。

实证结果表明取得了显著改进：

• **训练场景** – 在相同硬件配置下，Llama-8B 模型每步的迭代时间从使用传统方法的 5.2 秒缩短至 4.08 秒，性能提升约 20%。

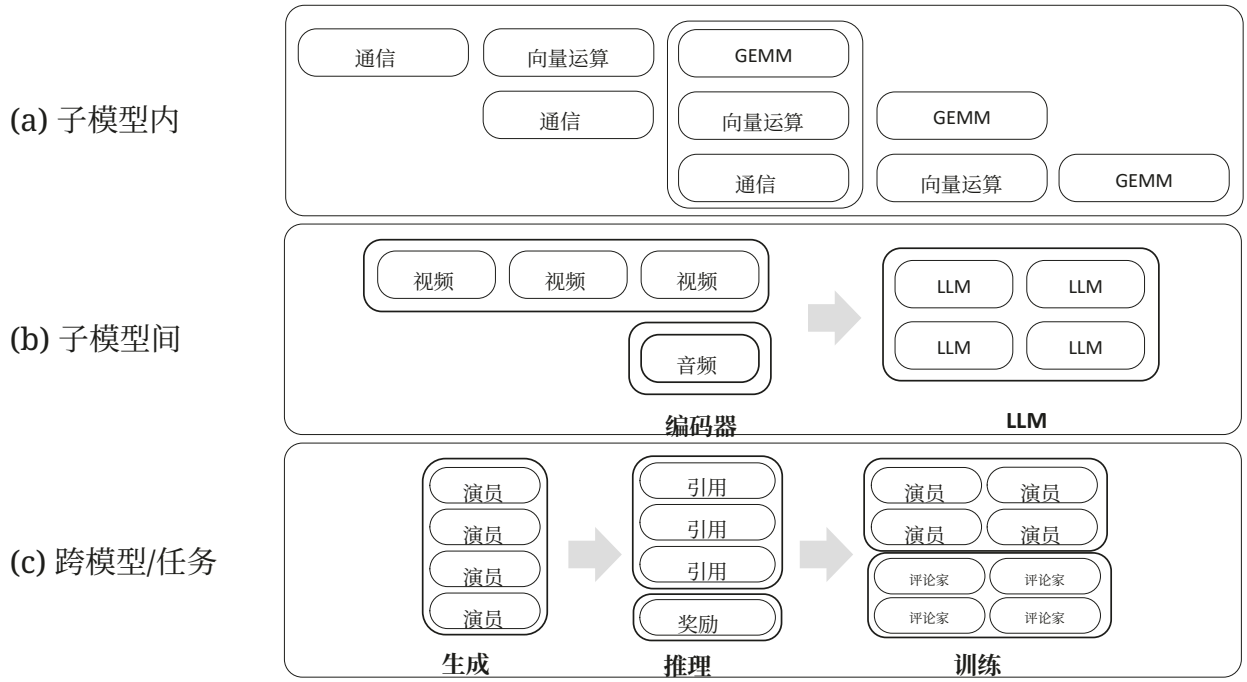


图 4：HyperMPMD 示意图。

• **推理场景** – 在相同延迟约束下，HyperOffload 将支持的序列长度从 71K 提升至 123K，相较于传统方法提升了 70%。

3.3 HyperMPMD

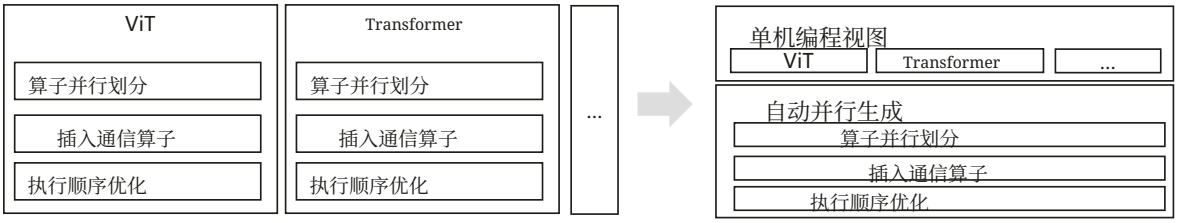
HyperMPMD的核心目标是实现灵活、细粒度的多程序多数据（MPMD）分片方法，以解决全模态学习和强化学习等场景中的计算不均衡问题，从而最大限度地提高硬件利用率。HyperMPMD 在三个不同维度上提供多程序多数据能力：

• **子模型内核心级并发**–通过对模型张量进行分片，并针对 AICube 和 AIVector 任务采用卡内多程序多数据调度，该框架实现了对计算与通信重叠的细粒度编排（如图 4 (a) 所示）。这解决了混合专家架构固有的通信掩蔽难题。通过协同卡内多核并行与卡间并行，HyperMPMD 将通信掩蔽比率从传统的 60% 提高到 90%。

• **子模型间并发均衡**–该框架将子图解耦为相互独立的并发任务（如图 4 (b) 所示），并利用动态调度来缓解负载不均衡。这有效消除了全模态或多模态模型中因异构子模块负载而通常出现的 10%–40% 流水线气泡，使整体训练性能提升约 15%。

• **跨模型并发调度** – 集成多程序多数据运行时后，该框架支持单控制器，可在超节点的池化计算资源内执行细粒度并行分片和动态调度（如图 4 (c) 所示）。这实现了模型级并发，消除了拖尾效应，解决了多任务强化学习中的负载不均衡问题，并将集群级资源利用率提升了 15%。

HyperMPMD 根据模态或任务（例如文本、图像、音频、融合以及任务调度组）划分出彼此独立的 MPMD 进程组。每个组执行专门的程序逻辑，并通过标准化接口进行通信。以下示例展示了全模态 MPMD 进程组的配置。通过将核心逻辑封装到独立模块中，并如清单1所示通过配置文件定义节点到模块的映射，该框架无需进行僵化的硬编码。



（a）命令式并行编程：
耦合的模型定义与并行策略。

（b）声明式并行编程：
解耦的模型定义与并行策略。

图5：声明式并行编程与命令式并行编程的比较。

清单1：映射配置示例。

```
进程_组 = {
    "文本": {"秩": [0,1], "模块": 文本模块, "硬件": "神经网络处理器"},
    "图像": {"ranks": [2,3], "模块": 图像模块, "硬件": "神经网络处理器"},
    "音频": {"ranks": [4], "模块": 音频模块, "硬件": "CPU"},
    "融合": {"秩": [5], "模块": 跨模态融合, "硬件": "神经网络
    "任务_调度器": {"秩": [6], "模块": 任务优先级调度器,
                    "硬件": "CPU"}
}
```

3.4 HyperShard

HyperShard 的根本目标是提供声明式并行编程能力，使开发者能够采用单节点编程惯例构建模型。这种方法有效地将算法逻辑与并行策略解耦，使后者能够根据网络拓扑灵活配置。如图5（a）所示，传统的命令式并行编程将模型定义与并行策略紧密耦合，因此每个模型变体都需要专门的手动干预，例如算子切分、显式插入通信算子以及执行序列优化。相比之下，图5（b）展示了声明式范式：研究人员从单设备视角开发算法，只需声明所需的并行约束。随后，框架会自动生成底层并行策略，实现完全解耦。

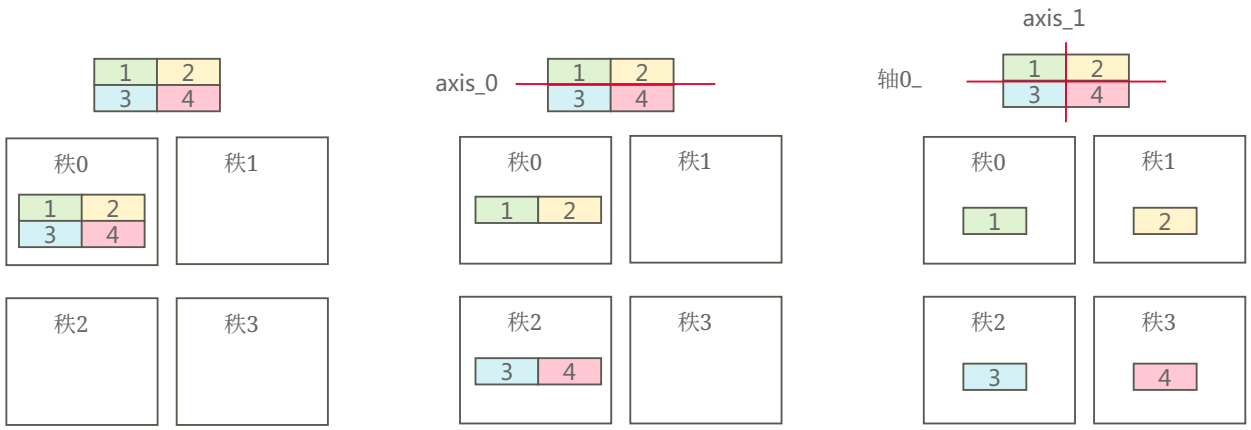
HyperShard 的主要编程抽象是接口 Layout(device_matrix, alias_name, tensor_map)，定义如下：

- 设备_矩阵 – 描述集群中加速器的逻辑排列方式。
- 别名_名称 – 为设备矩阵中的每个维度指定描述性标识符。
- 张量_映射 – 指定给定张量的每个维度如何在设备矩阵中进行分区。

清单 2：超切分策略生成示例。

```
# 4 个加速器组成 2*2 设备矩阵
设备_矩阵 = (2 , 2)
# 设备矩阵每个维度的别名
别名_名称 = ("x", "y")
# 假设张量 . 形状 =(2, 2)
张量_映射 = ("x", "y")
# 声明 Layout
layout =Layout ( 设备_矩阵 , 别名_名称 )
# 生成并行划分策略
并行_策略 =layout ( 张量_映射 )
```

以清单 2 中提供的代码为例，我们演示如何将一个张量分片到八个加速器上。自动分片过程分为三个阶段：



(a) 初始状态 (b) 沿轴 0分区 (c) 沿轴 1分区

图6：Hypershard分区示意图。

- **图6 (a)：初始化** – 框架最初假定张量位于单个逻辑秩上（例如，秩0）。
- **图6 (b)：第一维分区** – 由于 tensor_map[0] = "x"，框架沿设备_矩阵的 "x" 维对张量的第0维进行分片。
- **图6 (c)：第二维分区** – 由于 tensor_map[1] = "y"，框架沿设备_矩阵的 "y" 维对张量的第1维进行分片。

需要特别注意的是，此过程在编译时并不涉及张量物理切片。相反，框架基于用户配置对并行策略（分片策略）进行形式化推导；分片的实际执行发生在运行时。

通过将算法与并行策略解耦，使开发者免受硬件复杂性的影响，**Hy-perShard** 显著降低了工程开销。新算法并行化所需时间缩短至不到一天，而并行策略优化周期则从数天压缩至短短数小时。

4 总结与讨论

本文系统分析了模型工作负载、硬件架构以及训练与推理范式的演变趋势，指出大规模模型和超节点集群给人工智能框架带来的挑战。我们发现，当前框架在并行分片、状态管理和集群利用效率方面存在关键缺陷。为弥补这些不足，我们提出了超节点亲和型人工智能框架的设计，并基于 MindSpore 实现了 **HyperParallel** 架构。
通过整合三项核心技术——**HyperOffload**、**HyperMPMD**和**HyperShard**——该框架实现了计算与状态的解耦，支持细粒度的多程序多数据并行，并引入了声明式编程接口。这些创新带来了以下贡献：

- **复杂拓扑抽象化** – 该框架简化了开发与调优流程，使研究人员能够像进行单节点编程一样轻松地开发大规模模型。
- **灵活且高效的并行** – 通过提供动态、细粒度的并行策略调度，该框架最大限度地利用超节点计算资源，有效解决混合专家、全模态和强化学习场景中的负载不均衡问题。
- **统一状态管理**– 利用统一内存池管理中间状态，可提高内存利用率，从而支持训练和推理规模越来越大的模型，同时不降低性能。

实证结果表明，在硬件条件相同的情况下，HyperOffload 和 HyperMPMD 能显著提升训练和推理性能，同时大幅缩短算法开发所需的时间

以及并行策略优化。展望未来，超节点亲和型人工智能框架可进一步扩展，以支持异构任务调度、多节点动态扩缩容和跨数据中心协同训练，为下一代大规模模型研究与工业部署提供可靠的软件基础设施。

5 致谢

我们谨向 所有贡献者致以诚挚谢意，感谢他们在开发过程中发挥的宝贵作用 t HyperParallel 框架。

研究与工程

马长伟、傅超、李晨、刘崇明、雷达、范大伟、耿东杰、黄福鑫、张广鹏、王浩然、唐慧康、钱家宏、宋佳琪、丁金山、王开胜、李明琪、李诺航、郭琦、舒倩、张仁伟、李珊妮、孟祥宇、徐兴磊、高雄、黄毅、王一德、YifanYao、冯一兴、ZhenbangWang、李铮、杨振章、常哲睿、梁智博

参考文献

[1] Paul Barham 等人。Pathways：面向机器学习的异步分布式数据流。载于操作系统设计与实现研讨会（*OSDI*），2022年。[2] James Bradbury、Roy Frostig、Peter Hawkins、Matthew James Johnson、Chris Leary、Dougal Maclaurin、George Necula、Adam Paszke、Jake VanderPlas、Skye Wanderman-Milne 与 Qiao Zhang。JAX：Python+NumPy 程序的可组合变换，2018年。[3] Zixiang Chen、Yihe Deng、Yue Wu、Quanquan Gu 与 Yuanzhi Li。面向理解深度学习中的混合专家层。神经信息处理系统进展，35:23049–23062，2022年。[4] Frederica Darema。SPMD 模型：过去、现在与未来。载于欧洲并行虚拟机/消息传递接口用户组会议，第1–1页。施普林格，2001年。[5] Tim Dettmers、Ruslan Svirschevski、Vage Egiazarian、Denis Kuznedelev、Elias Frantar、Saleh Ashkboos、Alexander Borzunov、Torsten Hoefler 与 Dan Alistarh。SpQR：一种用于近无损大语言模型权重压缩的稀疏量化表示。*arXiv* 预印本 *arXiv:2306.03078*，2023年。[6] William Fedus、Barret Zoph 与 Noam Shazeer。Switch Transformers：借助简单高效的稀疏性扩展至万亿参数模型。机器学习研究期刊，23(120):1–39，2022年。[7] Jordan Hoffmann、Sebastian Borgeaud、Arthur Mensch、Elena Buchatskaya、Trevor Cai、Eliza Rutherford、Diego de Las Casas、Lisa Anne Hendricks、Johannes Welbl、Aidan Clark 等人。训练计算最优的大语言模型。*arXiv* 预印本 *arXiv:2203.15556*，2022年。[8] Ziheng Jiang、Haibin Lin、Yinmin Zhong、Qi Huang、Yangrui Chen、Zhi Zhang、Yanghua Peng、Xiang Li、Cong Xie、Shibiao Nong 等人。{MegaScale}：将大语言模型训练扩展至超过10,000 {图形处理器}。载于第*21*届 *USENIX* 网络系统设计与实现研讨会（*NSDI 24*），第745–760页，2024年。[9] In-Jun Jung、Gyeongrok Yang、Jaeha Min 与 Joo-Young Kim。HLX：面向混合 Transformer-Mamba 语言模型优化性能的统一流水线架构。载于第 *58* 届 *IEEE/ACM* 国际微体系结构研讨会论文集，第461–475页，2025年。[10] Huaicheng Li、Daniel S Berger、Lisa Hsu、Daniel Ernst、Pantea Zardoshti、Stanko Novakovic、Monish Shah、Samir Rajadnya、Scott Lee、Ishwar Agarwal 等人。Pond：面向云平台的基于 CXL 的内存池化系统。载于第*28*届 *ACM* 编程语言与操作系统体系结构支持国际会议论文集，第*2*卷，第574–587页，2023年。[11] H. Liao。UB-Mesh：面向大型 AI 超节点的新型互连技术。载于 *IEEE Hot Chips*，2025年。[12] Aixin Liu、Bei Feng、Bing Xue、Bingxuan Wang、Bochao Wu、Chengda Lu、Chenggang Zhao、Chengqi Deng、Chenyu Zhang、Chong Ruan 等人。DeepSeek-V3 技术报告。*arXiv* 预印本 *arXiv:2412.19437*，2024年。[13] Yinhan Liu、Myle Ott、Naman Goyal、Jingfei Du、Mandar Joshi、Danqi Chen、Omer Levy、Mike Lewis、Luke Zettlemoyer 与 Veselin Stoyanov。RoBERTa：一种经过稳健优化的 BERT 预训练方法。*arXiv* 预印本 *arXiv:1907.11692*，2019年。

[14] Dheevatsa Mudigere、Yuchen Hao、Jianyu Huang、Zhihao Jia、Andrew Tulloch、Srinivas Sridharan、Xing Liu、Mustafa Ozdal、Jade Nie、Jongsoo Park 等人。面向深度学习推荐模型快速可扩展训练的软硬件协同设计。载于第*49*届年度国际计算机体系结构研讨会论文集，第993–1011页，2022年。[15] Darpan Pandey、Kamal Niwaria 与 Bharti Chourasia。机器学习算法：综述。机器学习，6(2)，2019年。[16] Adam Paszke、Sam Gross、Francisco Massa、Adam Lerer、James Bradbury、Gregory Chanan、Trevor Killeen、Zeming Lin、Natalia Gimelshein、Luca Antiga 等人。PyTorch：一种命令式风格的高性能深度学习库。神经信息处理系统进展，32，2019年。[17] Samyam Rajbhandari、Jeff Rasley、Olatunji Ruwase 与 Yuxiong He。ZeRO：面向万亿参数模型训练的内存优化。载于国际高性能计算、网络、存储与分析大会论文集，SC '20。IEEE出版社，2020年。[18] Jeff Rasley、Samyam Rajbhandari、Olatunji Ruwase 与 Yuxiong He。DeepSpeed：系统优化使超过1000亿参数的深度学习模型训练成为可能。载于第*26*届 *ACM SIGKDD*知识发现与数据挖掘国际会议论文集，第3505–3506页，2020年。[19] Jie Ren、Samyam Rajbhandari、Reza Yazdani Aminabadi、Olatunji Ruwase、Shuangyan Yang、Minjia Zhang、Dong Li 与 Yuxiong He。 {ZeRO-Offload}：普及 {十亿规模}模型训练。载于*2021*年 *USENIX* 年度技术会议（*USENIX ATC 21*），第551–564页，2021年。[20] Xiaozhe Ren、Pingyi Zhou、Xinfan Meng、Xinjing Huang、Yadao Wang、Weichao Wang、Pengfei Li、Xiaoda Zhang、Alexander Podolskiy、Grigory Arshinov 等人。盘古-{\Sigma}：借助稀疏异构计算迈向万亿参数语言模型。*arXiv* 预印本 *arXiv:2303.10845*，2023年。[21] Mohammad Shoeybi、Mostofa Patwary、Raul Puri、Patrick LeGresley、Jared Casper 与 Bryan Catanzaro。Megatron-LM：利用模型并行训练数十亿参数语言模型。*arXiv* 预印本 *arXiv:1909.08053*，2019年。[22] Quan Sun、Yufeng Cui、Xiaosong Zhang、Fan Zhang、Qiying Yu、Yueze Wang、Yongming Rao、Jingjing Liu、Tiejun Huang 与 Xinlong Wang。生成式多模态模型是上下文学习者。载于*IEEE/CVF*计算机视觉与模式识别会议论文集，第14398–14409页，2024年。[23] An Yang、Anfeng Li、Baosong Yang、Beichen Zhang、Binyuan Hui、Bo Zheng、Bowen Yu、Chang Gao、Chengen Huang、Chenxu Lv 等人。Qwen3 技术报告。*arXiv* 预印本 *arXiv:2505.09388*，2025年。[24] Qiying Yu、Zheng Zhang、Ruofei Zhu、Yufeng Yuan、Xiaochen Zuo、Yu Yue、Weinan Dai、Tiantian Fan、Gaohong Liu、Lingjun Liu 等人。DAPO：大规模开源大语言模型强化学习系统。*arXiv* 预印本 *arXiv:2503.14476*，2025年。[25] Xia Zhao、Limin Wang、Yufei Zhang、Xuming Han、Muhammet Deveci 与 Milan Parmar。计算机视觉中卷积神经网络综述。人工智能评论，57(4):99，2024年。[26] P. Zuo 等人。在华为 CloudMatrix384 上服务大语言模型。*arXiv* 预印本 *arXiv:2506.12708*，2025年。