XXXX

面向PD分离式混合专家模型部署优化的分布式推理系统

孙梦宇[1]，谭宇凯[1]，陆钢[2]，黄志兰[2]，王亚森[2]，朱泽亚[1]，李伊青[2]，陈映[1]

（1. 中国电信研究院，中国 北京 102209；
2. 中国电信研究院，中国 广州 510630）

摘　要：通过将混合专家模型推理过程解耦为计算密集型的预填充阶段和**访存**密集型的解码阶段，分别部署在分布式物理计算节点，**从而实现推理系统的高效运行提升模型推理效率。**本文详细阐述了MoE大模型推理过程，基于Attention层和MoE层的推理过程，面向单推理任务和批处理推理任务过程进行在线推理系统建模，通过计算预填充阶段和解码阶段的计算和传输时延获得吞吐，旨在实现两阶段的吞吐量平衡。提出了一种基于二分查找算法的资源配置和策略部署机制，以确定每阶段的计算资源配比、部署实例数量和并行策略。**在两种主流计算节点进行验证，并与非PD分离基线方法和当前主流的分离式推理优化方法进行对比，**实验结果表明，**PD分离式推理相较于非PD分离基线方案，可达到3倍以上的吞吐提升，相较于当前主流方案仍有性能提升，且本文提出的**机制能够简化手动配置调优流程，为不同输入输出长度、并发数、请求频率条件下找到近似最优**的PD分离部署决策，相较于其它可行PD资源配比决策，**单卡平均吞吐量提升30-50%。

关键词：混合专家模型；PD分离式推理；在线推理系统；模型部署优化

中图分类号：

文献标志码：A

doi: 10.11959/j.issn.1000−0801.

Prefill and Decode Disaggregation Deployment Optimization for Mixture-of-Experts Model in Distributed Inference Systems

SUN Mengyu[1], TAN Yukai[1], Author Corresponding[2], LU Gang[2], HUANG Zhilan[2], WANG Yasen[1], ZHU Zeya[2], Li Yiqing[1]

1. China Telecom Research Institute, Beijing 102209, China
2. China Telecom Research Institute, Guangzhou 510630, China

Abstract: By decoupling the inference process of the Mixture of Experts (MoE) model into a computationally intensive prefill phase and a memory-intensive decode phase, and deploying them on distributed physical computing nodes respectively, **the efficient operation of inference systems is achieved, and the model inference efficiency is improved.** This paper elaborates on the inference process of the MoE model in detail. Based on the inference processes of the Attention layer and the MoE layer, it models the online inference system for both single inference tasks and

收稿日期：XXXX−XX−XX；修回日期：XXXX−XX−XX

batch inference tasks. Through calculating the computation and transmission latency of the prefill phase and the decode phase, the throughput is obtained, aiming to achieve throughput balance between the two phases. A resource configuration and strategy deployment mechanism based on the binary search algorithm is proposed to determine the computation resource ratio, the number of deployed instances, and the parallel strategy for each phase. **Experimental verification is conducted on two mainstream computing nodes, with comparisons made against both non-PD-disaggregation baseline and state-of-the-art PD disaggregation inference optimization approaches. Experimental results show that compared with non-PD-disaggregation baseline approach, PD disaggregation inference achieves a throughput improvement of more than 3x. It still outperforms state-of-the-art approach in performance. Additionally, the mechanism proposed in this paper can simplify the manual configuration and tuning process, finding near-optimal PD disaggregation deployment decisions under different conditions of input/output lengths, concurrency numbers, and request frequencies. Compared with other feasible PD resource allocation decisions, the average throughput per card is increased by 30-50%.**

Key words: Mixture-of-Experts Model, Prefill and Decode Disaggregation Inference, Online Inference System, Model Deployment Optimization

1 引言

随着通用人工智能（AGI）技术的高速发展，基于Transformer架构的大语言模型（LLM）得到了广泛应用，**依托于模型参数量和训练数据集规模的不断增加，**大模型性能持续提升，然而，受限于可用训练数据规模和所需计算资源的约束，LLM的Scaling Law法则增长规律逐渐放缓[1]，因此，模型结构正在逐渐向专家混合（MoE）架构过渡[2-4]。在MoE模型中，通过组合处理特定任务或数据类型的子网络（也称为“专家”），基于门控机制选择性地动态激活“少数专家”，其优势在于提高模型性能的同时控制计算成本，“按需调用”机制减少了每次推理时的计算量，成为一种新的服务模式[5]，具有代表性的MoE大模型包括Grok-4[6]和DeepSeek-V3[7],[8]。

MoE大模型推理是大模型应用的“最后一公里”，其推理过程分为预填充和解码两个阶段[9-11]，其中，预填充阶段针对用户输入的特定需求进行解析，其速度影响大模型生成第一个token所需时间，解码阶段依托首token及前序KV Cache以自回归形式逐个生成后续token，其速度影响每个输出token的间隔。根据用户需求场景的不同，对上述指标有差异化需求，**对需求的满足程度极大程度影响用户服务质量。**为了满足各类用户对多样化应用严格的时延要求，**常需要超额分配计算资源以维持服务水平目标（SLO）**[12]，**造成不可避免的资源浪费。因此，构建高效的大模型推理系统，需要在保证用户既定SLO的前提下，最大化资源利用率，从而实现降本增效。**

近年来，大模型推理系统成为智算领域关注的热点话题，其核心在于通过实现多租户推理任务的卸载优化和资源调度，**提高算力使用效能，从而快速、高质量地满足用户需求**。现有工作主要分为多租户推理任务调度和模型解耦部署两类，**前者将多个推理任务分发到不同计算节点，通过负载均衡优化任务卸载，**在不解耦推理任务执行过程的情况下对资源池全局任务进行深度优化[13-14]。后者则通过解耦大模型推理机理，将模型推理过程拆分成计算密集型的预填充（Prefill）阶段和访存密集型的解码（Decode）阶段，也称为PD分离式推理，分别部署在物理隔离的计算单元，从而解决两阶段因算力、显存资源竞争而造成的服务延迟问题，并可依据用户SLO需

求自定义资源和模型并行策略，使每阶段能够独立扩展[15-17]。本文采用第二种方法，对大模型推理过程进行解耦和精细化调度，并针对MoE模型结构精细化部署，以实现模型业务的资源部署优化和模型推理系统的最优吞吐量。本文主要创新点包括：（1）针对MoE大模型推理过程进行多层解耦，拆分为计算密集型的预填充阶段和访存密集型的解码阶段，不同阶段依据其模型结构由注意力层和混合专家层组合形成，将不同推理阶段部署在差异化计算节点上独立运行，从而消除**两阶段**间因相互干扰造成的资源浪费，从而优化系统处理效率；（2）构建了面向高并发场景的在线大模型推理系统，针对单一推理任务的计算开销和通信开销进行分阶段建模，考虑在线系统批处理情况，并计算高并发情况推理预填充阶段与解码阶段系统吞吐，形成目标优化问题，旨在平衡且最大化系统处理能力；（3）实现了一种基于二分搜索的分阶段资源配置和并行策略部署方法，基于两种主流算力设计不同模型参数、不同推理场景（包括用户输入/输出长度、并发量等）、不同资源配置情况下的对比实验，并在差异化场景下分别找到了最优解决方案，**并与非PD分离基线、现有PD分离式推理方法进行对比，**验证了方法的可行性与高效性。

全文组织如下，**第2章是针对MoE模型架构、大模型推理过程和MoE大模型推理影响因素**的先验知识，并针对本文使用的专业术语进行定义。第3章描述大模型推理系统模型，分别对单推理任务和在线系统的批处理推理进行建模。第4章介绍PD分离推理系统优化方法，**第5章通过大量实验验证，呈现了该方法在不同场景下的性能优势**。

2　研究背景与相关概念

2.1　专家混合模型MoE架构

MoE架构已经成为提高神经网络计算效率的关键方法，并在主流大模型中被广泛采用。与每层采用全连接前馈网络（Feedforward Neural Network，FFN）的传统Transformer架构不同，MoE架构旨在通过整合多个子模型，也被称为专家，并引入专家选择机制通过门控网络动态选择部分专家参与计算。MoE层通常以模块化方式集成到Transformer架构中，主要取代原始模型架构中的FFN子层，将传统的“Attention - FFN”堆叠转换为“Attention - MoE”。标准的MoE层由多个结构相同但具有独立参数的专家网络、一个根据输入特征选择激活相关专家的门控网络和一个路由-聚合机制组成，在每次前向传播期间，门控网络根据用户输入文本特征选择部分专家激活，路由-聚合机制将每个输入分配给选中的top-k个专家，并依据权重对专家的输出进行加权求和，得到最终的模型输出结果，从而实现了模型容量和计算成本之间的解耦，使得MoE架构在保持大模型低成本推理开销的同时显著扩展其参数规模。在MoE架构模型推理中，通过专家并行策略通将模型的多个专家分散到不同计算单元，突破单算力和显存限制，加速推理过程，优化推理性能。

2.2　大模型推理流程

大模型推理可分为预填充和解码两个顺序执行阶段，每阶段都与大模型的响应速度、资源利用、系统性能呈现正相关趋势。在预填充阶段，模型一次性接收用户输入的全量推理问题，通常以token形式表示，并行处理token并负责生成模型反馈答案的首token，由于每个输入token均需要通过transformer架构进行完整前向计算，因此通常被认为是计算密集型任务[18-19]。同时，在预填充阶段会产生键值缓存，通常被称为KV-Cache，这些KV-Cache将被存储以供后续阶段推理调用，以存换算从而无需重复向量计算。解码阶段大模型采用自回归策略，依据前序token和存储KV-Cache值每次生成单个token，固有的输

出顺序限制了并行计算的可行性，由于需要高频的KV-Cache读取和增量写入，通常被认为是访存密集型，尤其是当输出序列长度增加时，为显存带宽和访问时延带来巨大压力。大模型经过完整的预填充和解码阶段后，将问题答案按token逐步返回给用户，完成单个用户的大模型推理流程。

在大模型推理中，预填充和解码阶段的处理速度分别影响首token生成时间（Time-To-First-Token，TTFT）和token间隔时间（Time-Per-Output-Token，TPOT）两个评估系统性能的SLO指标[20]。具体来说，TTFT是指从用户提交问题到模型系统后获得第一个输出token的时间，从用户视角看，TTFT通常指问题回复响应时间，尤其在聊天机器人或实时交互工具等应用中，减少TTFT对于保持交互体验至关重要。TPOT表示在生成首token后，后续每个token输出的时间间隔，其反映解码阶段的性能，TPOT越小说明大模型推理系统能够有效维持回复的平滑生成，避免出现系统卡顿现象。

由于预填充和解码阶段的不同特点，在相同计算节点运行通常会导致算力、显存资源的相互竞争，出现两阶段性能降低，从而导致TTFT和TPOT间的性能不均衡。

2.3 MoE大模型推理影响因素分析

MoE模型通过引入动态路由机制实现专家的稀疏激活，其推理过程的特殊性主要体现在路由决策的动态性、通信模式的阶段差异性及专家负载的不均衡性三个方面，且上述特性在预填充与解码两个推理阶段呈现显著差异。

MoE模型的门控网络会根据输入特征动态选择激活的专家子集，该机制在预填充与解码阶段的计算复杂度与影响逻辑存在本质区别，在预填充阶段，需为每个输入token独立执行路由决策，导致计算图呈现“高度动态+稀疏化”特征，不同token的专家选择结果无固定规律，需实时适配输入内容调整计算路径；解码阶段因自回归生成特性，每次仅处理一个新生成token，路由决策的计算量显著降低，但决策结果直接决定后续KV-Cache的存储位置与访问模式，间接影响解码效率。MoE模型通常采用EP并行部署，即不同专家分散在独立计算单元，推理过程中需通过跨单元通信实现专家数据交互，其通信开销随推理阶段变化，在预填充阶段，所有输入token需并行处理，激活的专家分布在不同计算单元，必然产生频繁All-to-All通信（每个计算单元需与其他单元交换专家数据），成为预填充阶段的核心性能瓶颈；而在解码阶段，单次仅处理一个token，专家通信量大幅减少，但通信频率与自回归步数一致（每生成一个token需进行一次通信），对网络延迟的敏感度显著提升，微小的通信延迟会随生成步数累积，直接影响端到端时延。专家负载不均衡是MoE模型的固有问题，其影响程度随推理场景与部署规模显著变化，在单推理任务场景，负载不均衡影响较小，仅需关注单次任务内的专家激活分布，而在大规模在线推理场景，负载不均衡问题被显著放大。预填充阶段的不同输入token可能集中激活少数“热门专家”，导致对应计算单元过载，而其他单元处于空闲状态，严重制约并行效率；解码阶段中多个任务的解码过程并行执行，可能导致部分专家持续处于高负载状态，成为解码吞吐的瓶颈。

综上所述，在预填充和解码分离的部署架构中，MoE模型的推理性能瓶颈呈现显著的阶段分化，预填充阶段主要受制于动态路由带来的计算复杂度和频繁All-to-All通信产生的通信开销两大因素，解码阶段的核心制约因素转为专家负载不均衡（尤其是高并发下的持续过载）与高频通信对网络时延的敏感性。

2.4 相关概念

定义1（计算节点）：一个计算节点 nd^C 可以表示为一个六元组 $(id, n^{ut}, c^{ut}, s^{ut}, bdw^{in}, bdw^{out})$，

其中：

· id 表示 nd^C 的唯一标识符；

· n^{ut} 表示 nd^C 中计算单元数量；

· c^{ut} 表示 nd^C 中单计算单元的计算能力；

· s^{ut} 表示 nd^C 的存储空间；

· bdw^{in} 表示 nd^C 中多计算单元间的传输带宽；

· bdw^{out} 表示多个计算节点 nd^C 间的传输带宽；

每个计算节点包括多个计算单元，在大模型推理场景中，这些计算单元**表示**图形处理单元（Graphics Processing Unit，GPU），通常计算节点内的全部计算单元均为同一架构、同一型号，因此，它们具有相同的计算能力、存储空间和传输带宽。

定义 2（计算资源池）：一个计算资源池 P^{CR} 可以表示为一个三元组 $\left(f^{tp}, n^{cn}, ntw\right)$，其中：

· f^{tp} 表示 P^{CR} 的承载类型；

· n^{cn} 表示 P^{CR} 中包含的计算节点数量；

· ntw 表示 P^{CR} 中的网络拓扑结构；

计算资源池按照大模型推理系统功能分别用于预填充阶段和解码阶段，每个阶段单独在分布式资源池运行，算力规模从单个计算阶段到数百个甚至数千个计算节点不等，通过 spine-leaf 等网络架构高速互联。本文中的资源池也可指具有逻辑分区的物理资源，其核心思想在于预填充阶段和解码阶段不使用相同计算节点的资源，服务于分离式部署架构。

定义 3（推理任务）：一个推理任务 $tsk_{t,i}^{ifn}$ 可以表示为一个五元组 $(Lnt^{\text{in}}, Lnt^{\text{out}}, Ca, SLO_{TTFT}, SLO_{TPOT})$，其中：

· Lnt^{in} 表示 $tsk_{t,i}^{ifn}$ 的输入文本长度；

· Lnt^{out} 表示 $tsk_{t,i}^{ifn}$ 的输出文本长度；

· Ca 表示 $tsk_{t,i}^{ifn}$ 的缓存命中率；

· SLO^{TTFT} 表示 $tsk_{t,i}^{ifn}$ 的 TTFT 约束；

· SLO^{TPOT} 表示 $tsk_{t,i}^{ifn}$ 的 TPOT 约束；

每个推理任务根据其应用场景的差异性具有不同的输入和输出文本长度要求，在预填充阶段，系统通过解析输入文本计算 KV Cahce，典型的推理系统具备 KV Cache 存储能力，当输入文本的部分内容与先前处理的其它任务内容匹配时，这些片段无需重新计算，可以读取以存换算提升推理效率。因此，采用缓存命中率表示每个任务可重用 KV Cache 的比例。每个推理任务都应在其预先约定好的服务水平目标(SLO)下执行反馈，SLO 指标在大模型推理过程中主要包括从用户输入问题到首 token 反馈的 TTFT 时延和用户接收到连续 token 反馈的时间间隔 TPOT 时延，采用 TTFT 和 TPOT 共同评估推理任务的服务质量。

3　大模型推理系统模型

3.1　单推理任务建模

从计算和通信两个维度，对单推理任务处理过程进行建模，从计算维度，在预填充阶段和解码阶段，每个注意力层 C_P^{att} 和 C_D^{att} 所需的计算成本如公式（1）所示:

$$C^{att(P/D)} = 24 \times seq_{len} \times h^2 \qquad \#(1)$$

其中，seq_{len} 表示输入或输出的文本长度，当处于解码阶段时，seq_{len} 被设置为 1，表示单次解码仅输出一个 token，h 表示隐藏维度，其具体数值由模型架构决定。

某推理任务 $tsk_{t,i}^{ifn}$ 在预填充阶段和解码阶段，Attention 层的计算时延分别记作 $L^{att(P)}$ 和 $L^{att(D)}$，计算过程如公式（2）和（3）所示：

$$L^{att(P)} = \frac{C^{att(\mathrm{P})} \times \mathrm{tsk}_{t,i}^{ifn}\left(Lnt^{in}\right) \times (1 - \boldsymbol{C_a})}{N^{(\mathrm{P})} \times nd^C\left(n_j^{ut}\right) \times nd^C\left(c_j^{ut}\right)} \times L_i^{att} \quad (2)$$

$$L^{att(D)} = \frac{C^{att(\mathrm{D})} \times \mathrm{tsk}_{t,i}^{ifn}\left(Lnt^{out}\right)}{N^{(\mathrm{D})} \times nd^C\left(n_j^{ut}\right) \times nd^C\left(c_j^{ut}\right)} \times L_i^{att} \quad (3)$$

其中，L_i^{att} 表示预填充阶段和解码阶段的注意力

层数，$\mathrm{tsk}_{t,i}^{ifn}\left(Lnt^{in}\right)$和$\mathrm{tsk}_{t,i}^{ifn}\left(Lnt^{out}\right)$分别表示某推理任务 $\mathrm{tk}_{t,i}^{ifn}$ 的输入序列长度和输出序列长度，$N^{(P)}$和$N^{(D)}$分别表示预填充阶段和解码阶段部署的计算节点数量，$\mathrm{nd}^{\mathrm{C}}\left(\mathrm{n}_{\mathrm{j}}^{\mathrm{ut}}\right)$表示每个计算节点的计算单元数量，$nd^{C}\left(c_j^{ut}\right)$表示每个计算单元的计算能力。$C_a$表示缓存命中率，主要影响大模型推理的预填充阶段，**C_a越大，计算时延越小。**由于解析输入序列占据了大量的计算开销，其计算复杂度随输入长度呈现平方增长。因此，当输入序列中已有部分token曾被计算且缓存在KV Cache中时，这部分token的表示无需重新计算，只需从缓存直接读取即可。**例如，当某推理任务的缓存命中率C_a为40%，即在预填充阶段，有40%的输入序列不需要进行二次解析，只需解析剩余60%的输入序列，**这样的复用机制显著减少了需要参与计算的有效序列长度，从而大幅降低Prefill阶段的总时延，通过“以存换算”的方式实现了计算加速。

在预填充阶段和解码阶段每个MoE层的计算成本$C^{\mathrm{MoE}(P)}$和$C^{MoE(D)}$如公式（4）所示:

$$C^{MoE(P/D)}=C^{E}\times k \qquad (4)$$

其中，C^{E}表示MoE层中单个专家每token的计算成本，k表示前向传播期间激活的专家数量，MoE层的计算量很大程度取决于每次激活的专家数量。

某推理任务$tsk_{t,i}^{ifn}$在预填充阶段和解码阶段，MoE层的计算时延$L^{MoE(P)}$和$L^{MoE(D)}$计算如公式（5）和（6）所示：

$$L^{MoE(P)}=\frac{C^{\mathrm{MoE(P)}}\times \mathrm{tsk}_{t,i}^{ifn}\left(Lnt^{in}\right)}{N^{(\mathrm{P})}\times nd^{C}\left(n_j^{ut}\right)\times nd^{C}\left(c_j^{ut}\right)}\times\frac{\omega}{|\mathrm{EP}|}\times L_i^{MoE(P)} \qquad (5)$$

$$L^{MoE(D)}=\frac{C^{\mathrm{MoE(D)}}\times \mathrm{tsk}_{t,i}^{ifn}\left(Lnt^{out}\right)}{N^{(\mathrm{D})}\times nd^{C}\left(n_j^{ut}\right)\times nd^{C}\left(c_j^{ut}\right)}\times\frac{\omega}{|\mathrm{EP}|}\times L_i^{MoE(D)} \qquad (6)$$

其中$\omega\in[1,|\mathrm{EP}|]$，$\omega$表示EP并行策略对计算时延的影响系数，当专家负载完全均衡时ω=1，若存在专家分配负载不均，ω增大，最大不超过EP并行数。若ω超过EP并行数，则EP并行策略对推理性能没有优化效果。$L_i^{MoE(P/D)}$表示预填充阶段和解码阶段的MoE层数，$\mathrm{tsk}_{t,i}^{ifn}\left(Lnt^{out}\right)$表示推理任务 $\mathrm{tk}_{t,i}^{ifn}$ 的输出序列长度。

某推理任务$tsk_{t,i}^{ifn}$的预填充阶段所需的计算时延$L_{tsk_{t,i}^{ifn}}^{\mathrm{C}(P)}$可表示为公式（7）：

$$L_{tsk_{t,i}^{ifn}}^{\mathrm{C}(P)}=L^{att(P)}+L^{MoE(P)}\#(7)$$

其中，$L^{att(P)}$表示预填充阶段Attention层的计算时延，$L^{MoE(P)}$表示预填充阶段MoE层的计算时延

相似地，某推理任务$tsk_{t,i}^{ifn}$的解码阶段所需的总计算时延$L_{tsk_{t,i}^{ifn}}^{\mathrm{C}(\mathrm{D})}$可表示为公式（8）：

$$L_{tsk_{t,i}^{ifn}}^{\mathrm{C}(\mathrm{D})}=L^{att(P)}+L^{MoE(D)} \quad \#(8)$$

其中，$L^{att(P)}$表示解码阶段Attention层的计算时延，$L^{MoE(D)}$表示解码阶段MoE层的计算时延

从通信维度，预填充阶段和解码阶段中每注意力层$V^{att(P/D)}$所需的数据传输量计算如公式（9）所示:

$$V^{att(P/D)}=|DP|\times\left(\left(|TP|-1\right)\times C^{cn}+|PP|\times C^{cn}\right)\#(9)$$

其中，C^{cn}表示每token传输的数据量，通常通过物理测量获得具体值，$|DP|,|PP|$和$|TP|$表示LLM分布式部署时数据并行、流水线并行和张量并行策略的设置值。**根据并行部署策略，其间数据传输包括同一计算节点内不同计算单元间传输和不同计算节点间传输两种模式，**每种模式具有差异化带宽，从而导致不同时延。将同一计算节

点内不同计算单元间以及计算节点间的数据传输量分别记为 $V^{att(P/D)_{u2u}}$ 和 $V^{att(P/D)_{n2n}}$，且计算节点内和计算节点间的数据量和为一项推理任务的数据传输量，记作 $V^{att(P/D)_{u2u}}+V^{att(P/D)_{n2n}}=V^{att(P/D)}$。

预填充阶段和解码阶段中 MoE 层 $V^{MoE(P/D)}$ 所需的传输量计算如公式（10）所示:

$$V^{MoE(P/D)}=2\times V^{E}_{\text{token}}\times K \qquad \#(10)$$

其中，V^{E}_{token} 表示单专家每次发送和接收每 token 的数据传输量，K 表示每次激活的专家数量。相同地，同一计算节点内不同计算单元间和计算节点间的数据传输量分别记为 $V^{MoE(P/D)_{u2u}}$ 和 $V^{MoE(P/D)_{n2n}}$，且 $V^{MoE(P/D)_{u2u}}+V^{MoE(P/D)_{n2n}}=V^{MoE(P/D)}$，专家间的具体数据传输模式取决于激活专家的不同部署位置。

特定推理任务 $tsk^{ifn}_{t,i}$ 在预填充阶段或解码阶段的总传输延迟 $L^{T(P/D)}_{tsk^{ifn}_{t,i}}$ 如公式（11）所示:

$$L^{T(P/D)}_{tsk^{ifn}_{t,i}}=\frac{V^{att(P/D)_{u2u}}+V^{MoE(P/D)_{u2u}}}{nd^{C}_{p}\left(bdw_{in}\right)}+\frac{V^{att(P/D)_{\text{n2n}}}+V^{MoE(P/D)_{\text{n2n}}}}{nd^{C}_{(p,q)}\left(bdw_{out}\right)} \qquad \#(11)$$

3.2　在线推理系统建模

我们构建一个在线推理系统，用户定期发布推理任务请求并由该推理系统进行任务分发处理，系统初始化时间记作 t_0，记录每个推理任务的到达时间 $tsk^{arv}_{t,i}$。这些任务被依次放入预填充资源池和解码资源池按序进行两阶段处理，以生成所需的输出。我们使用系统单位时间内处理的 token 数量描述其处理能力，即吞吐量，在预填充阶段系统的吞吐量记作 γ^{P}_{t}，在时隙 t 时 D 解码阶段的吞吐量记作 γ^{D}_{t}，分别计算如公式（12）和公式（13）所示：

$$\gamma^{P}_{t}=\frac{\sum_{i=0}^{I}\text{tsk}^{ifn}_{t,i}\left(Lnt^{out}\right)}{\text{MAX}\left(\text{tsk}^{arv}_{t,i}-t_0+L^{T(P)}_{tsk^{ifn}_{t,i}}\right)} \qquad \#(12)$$

$$\gamma^{D}_{t}=\frac{\sum_{i=0}^{I}\text{tsk}^{ifn}_{t,i}\left(\text{Lnt}^{in}\right)}{\text{MAX}\left(\text{tsk}^{arv}_{t,i}-t_0+L^{T(D)}_{\text{tsk}^{ifn}_{t,i}}\right)-\text{MIN}\left(\text{tsk}^{arv}_{t,i}-t_0+L^{T(D)}_{\text{tsk}^{ifn}_{t,i}}\right)} \qquad \#(13)$$

值得注意的是，大规模在线推理系统中激活专家的负载情况会对系统吞吐产生一定影响，部分论文通过引入动态 Token 重分配、热点专家复制等机制，试图减小因专家负载不均而造成的性能损失。然而，本文中讨论的集群规模较小，负载不均导致的性能损失基本可忽略[23]。

4　PD 分离推理系统优化方法

4.1　问题定义

为确保 PD 分离推理系统达到最优性能，预填充阶段的输出吞吐量应与解码阶段的输入吞吐量相匹配，可防止任一阶段出现 token 处理阻塞。因此，在 PD 分离推理系统中，要求能够两个阶段实现吞吐量的动态平衡。因此，根据第 3 章针对两阶段的吞吐建模，将目标优化问题建模如公式（14）所示：

$$\gamma^{P}_{t}\approx\gamma^{D}_{t},\forall t \qquad \#(14)$$

优化问题求解过程中，需遵循每项推理任务不超过 TTFT 和 TPOT 的 SLO 约束，且张量并行、流水线并行、数据并行、专家并行的总数量与计算节点中的所有计算单元总数相同，形成约束条件如公式（15）至（17）所示:

$$tsk^{wt}_{t,i}+L^{C(P)_{tsk^{ifn}_{t,i}}}+L^{T(P)_{tsk^{ifn}_{t,i}}}\le tsk^{ifn}_{t,i}\left(SLO^{TTFT}\right) \qquad \#(15)$$

$$\frac{L^{C(D)_{tsk^{ijn}_{t,i}}}+L^{T(D)_{tsk^{ijn}_{t,i}}}}{tsk^{ijn}_{t,i}\left(Lnt^{out}\right)}\le tsk^{ifn}_{t,i}\left(SLO^{TPOT}\right) \qquad \#(16)$$

$$|TP|\times|PP|\times|DP|\times|EP|=\left(N^{(P)}+N^{(D)}\right)\times nd^{C}\left(n^{ut}_{j}\right) \qquad \#(17)$$

4.2 资源配置与策略部署机制

本文通过构建一种资源配置与策略部署（Resource Configuration and Strategy Deployment，RCSD）机制以寻找上述目标的最优解，基于MoE模型结构、算力/显存资源可用性、推理任务SLO约束等因素，本文提出的RCSD机制可以寻找：(i) 预填充阶段和解码阶段的算力资源配比，例如预填充阶段部署在6个计算节点，解码阶段部署在6个计算节点，(ii) 预填充阶段和解码阶段的实例部署数量，例如预填充阶段的6个计算阶段部署3个实例（即每2个计算节点部署1个实例）和解码阶段的6个计算节点部署1个实例，以及 (iii) 预填充阶段和解码阶段的并行策略，例如预填充阶段部署|TP|=8，|DP|=2，解码阶段部署|TP|=2，|PP|=8。本文的目标旨在寻找一种资源配置与策略部署的最佳决策，在保持预填充与解码两阶段吞吐量平衡的同时使得吞吐量最大化。

RCSD机制设计了一种两阶段资源供应方案，该方案分别针对预填充和解码阶段独立优化并行策略，以实现每阶段最优吞吐量，然后根据吞吐量比率按比例复制这些策略，以建立全局吞吐量平衡，从而为每个阶段**分配**专用的资源配置。资源供应确定过程高度依赖应用程序，不同输入/输出长度的工作负载和不可预测的并发数等均会对资源供应策略产生影响，在本文研究范围内，假设系统已知推理任务的大多数工作负载、任务到达时间以及输入/输出长度的分布，RCSD机制通过二分搜索和模拟试验逐步枚举并行部署策略配置，以确定SLO下的最大吞吐量。

算法 1 表示上述过程伪代码，RCSD机制划分预填充和解码两阶段资源，并遍历两阶段可行的并行策略配置，从而计算所有可行策略条件下的吞吐（第2-8行），针对预填充阶段全部可行性方案的吞吐量，选择最大的吞吐值，并将此策略作为预填充阶段的最优策略（第9行）。对解码阶段全部可行性方案所对应的吞吐值进行升序排序（第10行），并采用二分查找法找到最接近预填充阶段吞吐值的策略方式（第11行），同步根据确定的两阶段并行策略方式分配计算单元数量，获得最优资源配置与策略部署决策方案（第12-14行）。在本文的设置中，搜索空间有限且计算复杂度降低，因此即使通过建模后的枚举法也能实现快速收敛，并替代传统手动针对并行策略、实例数量、资源配比的调优方式，自动化完成资源配置与策略部署的决策。本算法的时间复杂度为$O(|K_p|+|K_D|\times\log|K_D|)$，其中，$|K_p|$表示预填充阶段的可行配置数量，与资源数量和可行并行策略数量相关，$|K_D|$表示解码阶段的可行配置策略数量。

需要注意的是，RCSD机制主要针对预填充和解码资源池的资源配置与策略部署，旨在模型部署时寻找到近似最优的部署优化方法，在此过程中，会充分考虑模型架构及推理任务情况，然而并不能保证在极高并发场景下满足全部推理任务的TTFT与TPOT需求。本文所提出的方案逻辑是针对无法满足特定SLO需求的推理任务进行柔性降级，并在降级后的可行域内选择执行与约定TTFT或TPOT最接近用时（即执行时延最低）的方案，以期获得较高的服务质量。

5 实验评估

5.1 实验设置

本文在两种不同的环境中测试了资源配置与策略部署RCSD机制的有效性，实验环境1采用某主流算力（记作A），每节点配置8卡80GB的GPU作为计算单元，卡间通过PCIe 5.0互联；实验环境2采用某国产主流算力（记作B），每节点配置8卡64GB的GPU作为计算单元，卡间通过高速互联模组连接。服务器间互联基于RDMA RoCEv2高速网络，且每节点配置8块200G的RDMA网口，参数平面网络交换机由6个400G

算法 1 资源配置与策略部署RCSD方法

Require：

$nd^C\left(n^{ut}\right)$：每个计算节点上的典型计算单元数量

N：总计算单元数量

N^P：预填充阶段分配计算单元数量

N^D：解码阶段分配计算单元数量

LLM^{STG}：大模型所需的存储空间

Ensure:

A_{opt}：最优资源配置与策略部署决策

$cfg^P, cfg^D \leftarrow \varnothing, \varnothing$

for all $(N^P \geq \frac{LLM^{STG}}{nd^C(s^{ut})})$ do

if $(cfg_i^P(EP)\times cfg_i^P(PP)\times cfg_i^P(TP)\times cfg_i^P(DP)=N^P)$ then

$R_i^P \leftarrow \gamma_i^P(cfg_i^P)$

end if

for all $(N^D \geq \frac{LLM^{STG}}{nd^C(s^{ut})} \,\&\, N^D \leq N-N^P)$ do

if $(cfg_i^D(EP)\times cfg_i^D(PP)\times cfg_i^D(TP)\times cfg_i^D(DP)=N^D)$ then

$R_j^D \leftarrow \gamma_j^D(cfg_j^D)$

end if

end for

end for

$\gamma_{max}^P \leftarrow \mathrm{MAX}(R_i^P)$

$R_j^{D\_Seq} \leftarrow \mathrm{Sort}(R_j^D)$

$\gamma_{fit}^D \leftarrow \mathrm{BinarySearch}(R_j^{D\_Seq}, \gamma_{max}^P)$

$n \leftarrow \left\lceil \frac{R}{\gamma^P} \right\rceil$

$m \leftarrow \left\lceil \frac{R_R^{rg}}{\gamma_{offs}^D} \right\rceil$

$A_{\text{opt}} \leftarrow \left(n, \gamma_{max}^P(cfg^P), m, \gamma_{fit}^D(cfg^D)\right)$

RDMA网络交换机组成两层spine-leaf网络。在实验过程中，将推理任务SLO设置为TTFT＜1秒和TPOT＜200毫秒，并记录高并发推理任务处理过程中的推理吞吐。

实验采用Qwen和DeepSeek系列的最新开源模型，由于算力A组成的实验环境规模较小，因此部署DeepSeek-R1-Distill-Qwen-32B模型，算力B组成的实验环境规模相对较大，因此部署DeepSeek-R1-671B模型，且按照显存容量计算，每2个计算节点可部署一个完整的模型实例。在实验中，分别模拟了在线用户推理场景中LLM推理的计算和通信过程。

5.2　实验结果

基于真实数据集[21-22]，将RCSD机制获取到的最优解与部署过程中其他可能的解决方案进行对比。服务拉起阶段会进行显存预分配，用于权重存储、KV Cache预留等核心场景，实际运行过程中显存利用率稳定在80%左右，且该利用率水平主要与输入输出序列长度呈正相关，随序列长度的动态变化呈现规律性波动。实验环境1评估算力A部署DeepSeek-R1-Distill-Qwen-32B进行PD分离推理性能，由于算力A显存为80G，因此使用单计算单元即可部署完整DeepSeek-R1-Distill-Qwen-32B模型，图1分别展示了不同并发数和不同请求频率情况下的吞吐情况，用以模拟推理系统任务压力。图1(a)展示了此场景下可行的PD分离资源配比情况下（包括三种情况，1P1D，即1个计算单元部署P阶段，1个计算单元部署D阶段，且实例数量均为1；1P2D，即1个计算单元部署P阶段，1个计算单元部署D阶段，P阶段与D阶段实例数量为1:2；2P1D，即1个计算单元不部署P阶段，1个计算单元部署D阶段，且P阶段与D阶段的实例数量为2:1）的平均单卡吞吐量，并使其符合既定SLO。

当推理任务并发数逐渐增加时，1P1D、1P2D和2P1D的吞吐量也逐渐增加，当并发数达到100后，吞吐量的上升趋势减缓，且当并发数达到120时，吞吐量达到最大值。当并发数达到120时，RCSD机制寻找到的最优1P2D方案的吞吐量相比于比其他资源配置的吞吐量提升30%至50%。图1(b)也呈现同样的趋势，当请求频率逐渐增加时，1P2D也实现了最佳的吞吐量性能，并当请求频率达到120时，其吞吐量比其他资源配置的吞吐量高升30%至50%。这是由于，通过

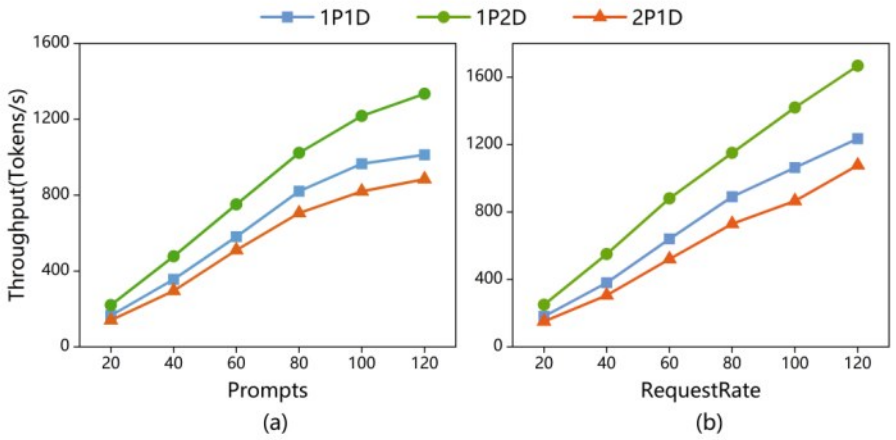

图1　(a)不同并发数情况下,(b)不同请求频率情况下,采用主流算力A(80G显存)进行不同PD资源配比情况下的单卡吞吐

RCSD机制可以找到预填充阶段和解码阶段最优的平衡吞吐，有效的减少了预填充阶段与解码阶段由于任务处理速率不匹配导致的相互等待，获得了最大的算力使用效率。

为了深度探索RCSD机制在推理集群的应用效果，使用算力B的16个计算节点集群部署DeepSeek-R1-671B并评估推理性能。图2(a)-(b)分别展示了在符合既定SLO条件下，输入序列长度为512个token和1.5K个token的非PD分离部署和PD分离部署8P8D的性能对比情况。按照算力B的64G显存计算，模型部署最少需要2个计算节点共16个计算单元。PD分离部署在预填充和解码阶段可行资源配置依托2的整数倍进行取值，本次实验采用8P8D场景与非PD分离基线方法做对比。为了保持总算力相同，非PD分离部署共使用16个计算节点部署8个模型服务。如图2(a)显示，在输入序列长度为512时，输出序列长度分别设置为512、1K、2K、4K、8K和16K，相比于非PD分离部署，PD分离方案均存在显著优势，尤其是在输出序列长度为2K-4K时，吞吐优势明显，平均可达非PD分离基线部署吞吐的3.8倍，最高可达5.4倍。当输出长度达到16K时，非PD分离部署无法提供有效服务，而PD分离方案仍然可支持较高吞吐，也因此说明PD分离方案能够支持更长上下文长度。图2(b)表示输入序列长度为1024时, 同等算力条件下，PD分离与非PD分离的吞吐对比，与图2(a)趋势类似，PD分离方案在输出序列长度为1K-4K时吞吐具有更大优势，此时平均吞吐可达非PD分离部署的3.1倍。最高可达3.9倍。在当前输入长度条件下，非PD分离部署在输出大于等于8K的场景下就无法正常提供服务，进一步验证了PD分离方案的优势。

图3(a)-(f)分别展示了在符合既定SLO的条件下，预填充和解码阶段的四种可能资源配置方案，分别记为4P13D、6P10D、8P8D和10P6D，按照算力B的64G显存计算，部署模型最少需要2个计算节点共16个计算单元，因此预填充和解码阶段可能的资源配置依托2的整数倍进行变换。随着请求频率逐渐增加，4P13D、6P10D、8P8D和10P6D的吞吐量均呈现上升趋势。图3(a)中显示，当输入输出长度分别为512token时，请求频率在20时达到峰值，即每秒到来20个推理任务需要处理时，达到系统的最大可接受程度，当推理请求数量增加后，系统达到过饱和状态，过多的等待任务堆积拖慢系统的处理速度。图3(b)-(f)中显示，峰值吞吐量大多数情况下出现在请求频率为30时，且根据输入、输出长度差异，RCSD

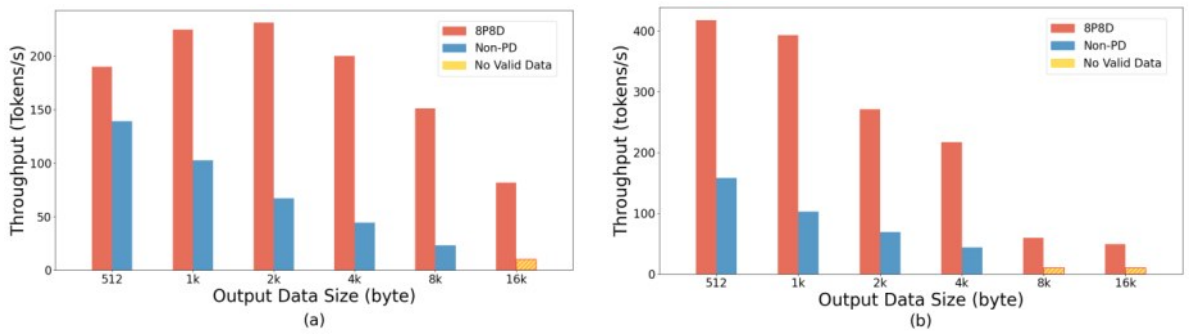

图2　(a) 输入长度512，输出长度512-16k时，(b) 输入长度1.5k，输出长度512-16k时，在16个计算节点B (64G显存)集群上吞吐量随输出序列长度的变化情况

机制寻找到的最优部署方案不同，在较短输出场景下，10P6D方案效果最好，因为推理任务计算瓶颈在预填充阶段的文本解析过程，在较长输出场景下，6P10D与8P8D方案效果最好，因为推理任务瓶颈在解码阶段，因此对解码阶段相对分配更多的资源。从图中可以看出，RCSD机制选择的最优资源配置，相比其他手动调整的可行资源配置方案，单卡吞吐量提升20%-50%。这是由于RCSD机制在为预填充和解码阶段寻找最优资源配比的同时，选取最合适的并行策略优化目标。具体而言，通过分析DeepSeek-R1-671B在算力B上部署策略与实验验证，RCSD机制可以有效地平衡两阶段多实例间的负载，以最小的成本满足SLO要求，通常两阶段资源配比、实例数量和各自的并行策略需要通过手动调整尝试多次，或根据经验进行部署，复杂度极高，也证明了该方案的有效性。

将本文提出的RCSD与Washington大学和Microsoft联合提出的推理优化方法Splitwise[15]进行对比，其中，Splitwise方法利用类似预填充阶段与解码阶段分离的方式，构建预填充资源池、解码资源池和混合资源池，分别处理计算密集型的预填充任务、访存密集型的解码任务和处理两阶段混合型任务，以应对突变负载。图4呈现了Splitwise与本文提出的RCSD方案，在不同输入输出序列长度下的吞吐量对比，如图中显示，当推理任务的输入输出长度序列较短时，在我们的实验场景主要表现为512以内，两种方法的吞吐相近。然而，随着输入输出序列长度的逐渐增加，Splitwise方法与本文提出的RCSD方法间的吞吐量差距呈逐渐扩大趋势。究其原因，**Splitwise方法虽然通过混合池增加了资源灵活度性，但推理服务在预填充池、解码池及混合池间进行动态切换，造成一定的推理服务状态扰动**[15]，**且仅支持并行度受限的部署模式，难以实现性能最大化**[24]，**而RCSD方法快速选择最优的资源策略和并行部署策略，资源高效利用的同时兼顾了多种并行策略部署混用。因此，**在性能上具有一定优势，证明了该方案的有效性。

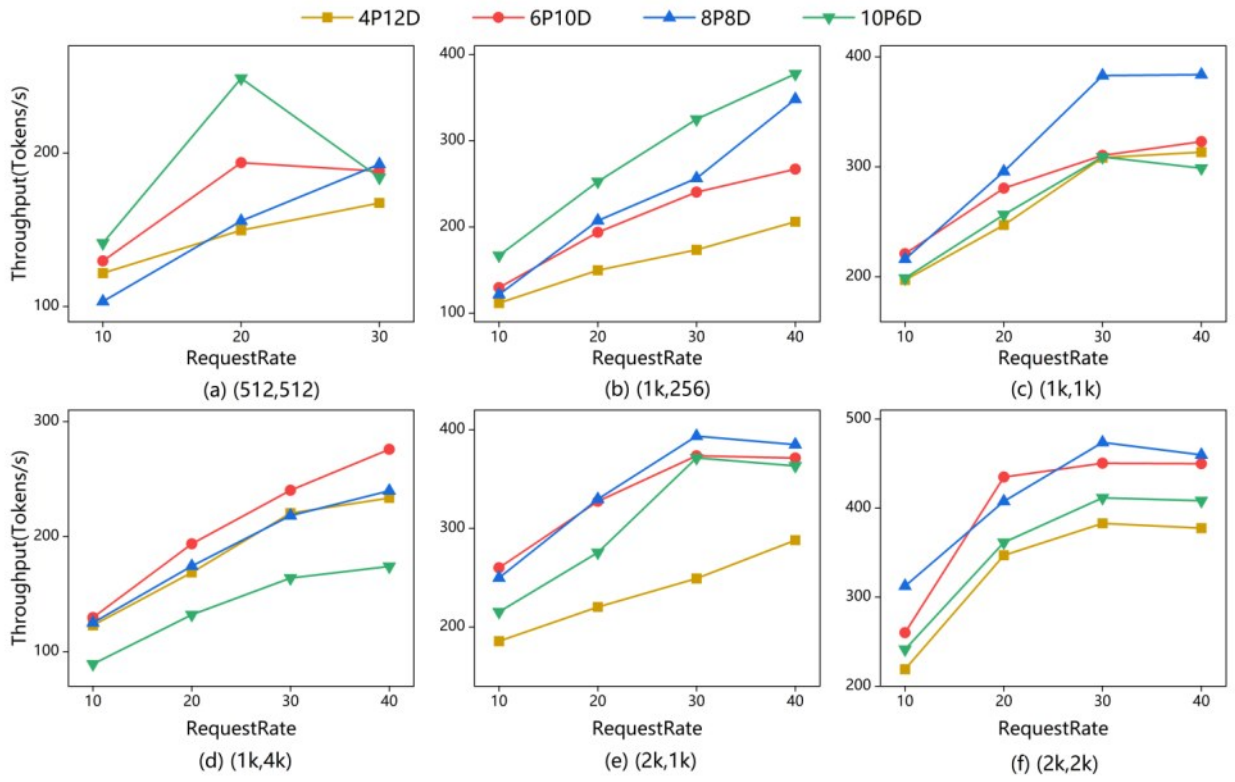

图3　(a) 输入长度512，输出长度512时，(b) 输入长度1024，输出长度256时，(c) 输入长度1024，输出长度1024时，(d) 输入长度1024，输出长度4096时，(e) 输入长度2048，输出长度1024时，(f) 输入长度2048，输出长度2048时，在16计算节点算力B(64G显存)集群上吞吐量随请求速率的变化情况

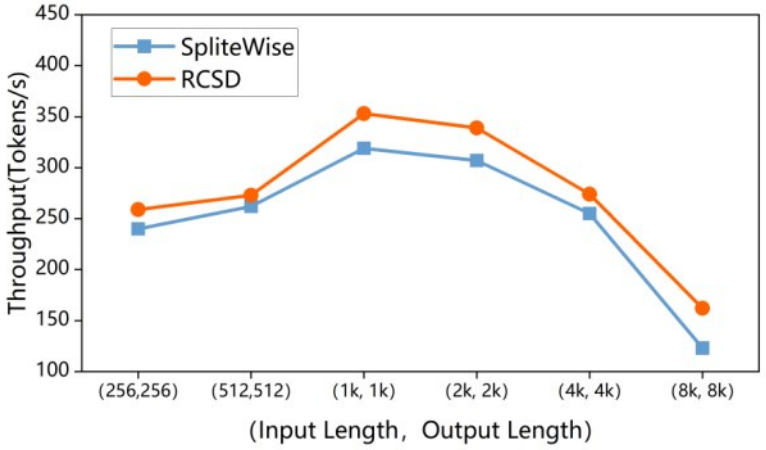

图4　RCSD和Splitewise两种方法在16计算节点B集群上吞吐量随输入序列长度和输出序列长度的变化情况

参考文献：

[1] HOFFMANN J, BORGEAUD S, MENSCH A, et al. Training compute-optimal large language models[C]. Advances in Neural Information Processing Systems 35 (NeurIPS 2022), New Orleans: Curran Associates Inc., 2022: 30016-30030.

[2] CAI W, JIANG J, WANG F, et al. A survey on mixture of ex-

perts in large language models[J]. IEEE Transactions on Knowledge and Data Engineering, 2025, 37(7): 3896–3915.

[3] 问佳琳, 李晓军, 姚俊萍, 等. 算力约束下混合专家模型计算优化方法:现状及研究进展[J]. 计算机工程与应用, 2025, 61(22): 20-35.
WEN J L, LI X J, YAO J P, et al. Computational optimization methods for mixture-of-experts models under computing power constraints: current status and research progress[J]. Computer Engineering and Applications, 2025, 61(22): 20-35.

[4] 史宏志, 赵健, 赵雅倩, 等. 大模型时代的混合专家系统优化综述[J]. 计算机研究与发展, 2025, 62(5): 1164-1189.
SHI H Z, ZHAO J, ZHAO Y Q, et al. Survey on optimization of mixture-of-experts systems in the era of large models[J]. Journal of Computer Research and Development, 2025, 62(5): 1164-1189.

[5] ZHANG Y, LIANG W, XU Z, et al. AoI-aware inference services in edge computing via digital twin network slicing[J]. IEEE Transactions on Services Computing, 2024, 17(6): 3154–3170.

[6] https://grok.com/. XAI. Grok-3 [EB/OL]. 2025.

[7] https://github. com/deepseek-ai/DeepSeek-V3. DEEPSEEK-AI. DeepSeek-V3 [EB/OL]. [2025-09-25].

[8] DAI D, DENG C, ZHAO C, et al. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models[J]. arXiv preprint arXiv:2401.06066, 2024

[9] 徐子恒, 雷波, 孙一豪, 等. 异构智算中心分布式 PD 分离推理技术的探索与实践[J]. 通信世界, 2025(18): 41-42.
XU Z H, LEI B, SUN Y H, et al. Exploration and practice of distributed prefill and decode disaggregation inference technology in heterogeneous intelligent computing centers[J]. Communications World, 2025(18): 41-42.

[10] PATEL P, CHOUKSE E, ZHANG C, et al. Splitwise: Efficient generative LLM inference using phase splitting[C]. 2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA), Piscataway: IEEE Press, 2024: 118-132.

[11] KWON W, LI Z, ZHUANG S, et al. vLLM: Easy, fast, and cheap LLM serving with PagedAttention[EB/OL]. 2023. https://vllm.ai/ (accessed 9 August 2023).

[12] ZHU R, JIANG Z, ZHANG Z, et al. Cannikin: No lagger of SLO in concurrent multiple LoRA LLM serving[J]. IEEE Transactions on Parallel and Distributed Systems, 2025, 36(9): 1972-1984.

[13] KAKOLYRIS A K, MASOUROS D, XYDIS S, et al. SLO-aware GPU DVFS for energy-efficient LLM inference serving [J]. IEEE Computer Architecture Letters, 2024, 23(2): 150–153.

[14] FANG J, HE Y, YU F R, et al. Large language models (LLMs) inference offloading and resource allocation in cloud-edge networks: An active inference approach[C]. 2023 IEEE 98th Vehicular Technology Conference (VTC2023-Fall), Piscataway: IEEE Press, 2023: 1–5.

[15] CHOUKSE E, PATEL P, ZHANG C, et al. Splitwise: efficient generative LLM inference using phase splitting[J]. IEEE Micro, 2025, 45(3): 78-92.

[16] ZHONG Y, LIU S, CHEN J, et al. DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving[C]//17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24), Santa Clara: USENIX Association, 2024: 1-15.

[17] RAJBHANDARI S, LI C, YAO Z, et al. DeepSpeed-MoE: Advancing mixture-of-experts inference and training to power next-generation AI scale[C]. International Conference on Machine Learning (ICML 2022), Baltimore: PMLR, 2022: 18332-18346.

[18] JIANG H, LI Y, ZHANG C, et al. MInference 1.0: Accelerating pre-filling for long-context LLMs via dynamic sparse attention [C]. Advances in Neural Information Processing Systems 37 (NeurIPS 2024), 2024: 52481-52515.

[19] FU Q, CHO M, MERTH T, et al. LazyLLM: Dynamic token pruning for efficient long context LLM inference[J]. arXiv preprint arXiv:2407.14057, 2024.

[20] HE Z, ZHANG Y, ZHANG C, et al. TriangleMix: A lossless and efficient attention pattern for long context pre-filling[J]. arXiv preprint arXiv:2507.21526, 2025.

[21] COBBE K, KOSARAJU V, BAVARIAN M, et al. Training verifiers to solve math word problems[J]. arXiv preprint arXiv: 2110.14168, 2021.

[22] HENDRYCKS D, BURNS T, BASART S, et al. Measuring massive multitask language understanding[C]. 9th International Conference on Learning Representations (ICLR 2021), Virtual Event, 2021: 4762-4772.

[23] MIRIYALA V P K, SVIRIDOV G, CHEN B X, et al. Latency-optimal load balancing for distributed MoE inference[C]. Proceedings of the 1st Workshop on Inter-networking challenges for AI (INET4AI '25), New York: Association for Computing Machinery, 2025: 7–13.

[24] STRATI F, MCALLISTER S, PHANISHAYEE A, et al. DéjàVu: KV-cache streaming for fast, fault-tolerant generative LLM serving[C]. Proceedings of the 41st International Conference on Machine Learning (ICML 2024), Vienna: Proceedings of Machine Learning Research (PMLR), 2024: 49605-49626.

[作者简介]

孙梦宇（1994-），女，中国电信集团公司研究院云网融合技术研究所科研人员，副研究员，主要研究方向包括云计算、智算基础设施、大模型推理、服务计算等。

谭宇剑（1995-），男，中国电信集团公司研究院云网融合技术研究所科研人员，主要研究领域包括智算基础设施、云原生、大模型推理等。（通信作者）

陆钢（1978-），男，中国电信集团公司研究院云网融合技术研究所副所长，教授级高级工程师，主要研究领域包括云及算力基础设施、先进计算等。

黄志兰（1981-），女，中国电信集团公司研究院云网融合技术研究所云及算力基础设施技术研究中心总监，主要研究领域包括智算基础设施、大模型适配优化、网络云化等。

王亚森（1995-），男，中国电信集团公司研究院云网融合技术研究所科研人员，主要研究领域包括智算基础设施、大模型推理等。

朱泽亚（1995-），男，中国电信集团公司研究院科研人员，主要研究领域包括智算基础设施、云原生、标准研究等。

李伊青（1994-），女，中国电信集团公司研究院云网融合技术研究所科研人员，主要研究领域包

括网络云化、大模型推理等。

陈映（1994-），男，中国电信集团公司云网融合技术研究所云及算力基础设施技术研究中心副总

监，主要研究领域包括智算基础设施、智算网络等。