计 算 机 研 究 与 发 展
Journal of Computer Research and Development
https://crad.ict.ac.cn
63(8): 1988−2004, 2026

面向边缘算力网络的混合专家模型协同推理系统

叶盛源　梁　涵　陈　旭
（中山大学计算机学院　广州　510006）
（yeshy8@mail2.sysu.edu.cn）

Collaborative MoE Inference System over Edge Computility Networks

Ye Shengyuan, Liang Han, and Chen Xu
（*School of Computer Science and Engineering*, *Sun Yat-sen University*, *Guangzhou* 510006）

Abstract　MoE (mixture of expert) architectures have become a widely recognized paradigm for scaling large language models (LLMs), and have been adopted by many frontier models deployed at the network edge. However, deploying MoE-based LLMs on resource-constrained edge devices remains challenging due to their massive parameter sizes, which often exceed the memory capacity of individual platforms. Existing in-situ MoE serving systems adopt expert swapping strategies that offload inactive experts to secondary storage and load them on demand. As LLMs support increasingly longer context windows with correspondingly longer user inputs, the prefilling phase faces a critical bottleneck: long input sequences activate nearly all experts, and the resulting full-scale expert swapping generates substantial I/O traffic, leading to prolonged TTFT (time to first token) latency. Typical edge environments, such as smart homes and smart factories, comprise multiple trusted interconnected devices, including tablets, laptops, and smart home appliances connected via local area networks. These devices collectively form a pervasive edge computility network (ECN), where each node carries underutilize computing power, memory capacity, and I/O bandwidth. This motivates us to harness the aggregated resources of the ECN through coordinated cross-node scheduling. This paper presents Sirius, a collaborative MoE inference system for edge computility networks that pools the collective resources across ECN nodes to accelerate the prefilling phase and reduce TTFT. Specifically, Sirius proposes three key techniques: 1) an interleaved sequence partitioning strategy that resolves workload imbalance in causal attention computation and maximizes communication-computation overlap; 2) an adaptive token scheduling mechanism that jointly considers computation overhead, swap latency, and expert buffer cache states to fully utilize collective I/O bandwidth while achieving load balance; 3) a speculative expert prefetching scheme that exploits the high similarity between adjacent-layer hidden states to overlap expert loading with computation, effectively hiding I/O latency. Experiments on real-world ECN platforming with Mixtral-8x7B and Qwen3-30B-A3B demonstrate that Sirius achieves 1.2–4.0 times inference throughput improvement over state-of-the-art baselines.

Key words　MoE; edge computility network; collaborative inference; sequence parallelism; expert swapping

摘　要　混合专家模型（mixture of expert, MoE）架构已成为扩展大语言模型（large language model, LLM）的广泛认可范式，并被众多前沿模型所采用。然而，MoE 模型参数量巨大，其内存占用往往超出单一边缘

收稿日期：2026-03-12；**修回日期**：2026-04-20
基金项目：国家自然科学基金项目（U25B2002）；广东省重点领域研发计划（2025B0101080001）；广东省珠江人才计划引进创新团队（2017ZT07X355）
This work was supported by the National Natural Science Foundation of China (U25B2002), the Key Research and Development Program of Guangdong Province (2025B0101080001), and the Pearl River Talent Plan for Introducing Innovative Teams of Guangdong Province (2017ZT07X355).
通信作者：陈旭（chenxu35@mail.sysu.edu.cn）

平台的容量，在资源受限边缘设备上部署面临严峻挑战。现有本地 MoE 服务系统采用专家交换策略，将非活跃专家卸载至二级存储并按需加载。随着 LLM 支持的上下文窗口不断扩展，用户输入也相应变长，预填充阶段面临关键瓶颈：长输入序列几乎激活所有专家，全量专家交换产生大量 I/O 流量，导致首词时延（time to first token, TTFT）显著延长。在智能家居、智能工厂等典型边缘场景中，平板电脑、笔记本电脑及各类家庭智能设备等多类可信终端通过局域网彼此互联，共同构成一张泛在的边缘算力网络（edge computility network, ECN）。边缘算力网络中各节点均携带大量闲置的算力、内存容量与 I/O 带宽资源，为汇聚分布式资源、突破单设备瓶颈提供了重要机遇。Sirius 是一种面向边缘算力网络的混合专家模型协同推理系统，通过跨节点的协同调度汇聚资源，加速预填充阶段并降低 TTFT。具体而言，设计了 3 项关键技术：1）基于交错划分的序列并行策略，解决因果注意力计算中的工作负载不均衡问题，最大化通信-计算重叠；2）自适应 Token 调度机制，联合考虑计算开销、交换时延和专家缓冲区缓存状态，充分利用集体 I/O 带宽并实现负载均衡；3）推测性专家预取方案，利用相邻层隐藏状态的高相似性提前预测专家激活，将专家加载与计算重叠，有效隐藏 I/O 时延。在真实边缘算力网络平台上，针对 Mixtral-8x7B 与 Qwen3-30B-A3B 的实验表明，Sirius 相比现有基线方法实现了 1.2~4.0 倍的推理吞吐量提升。

关键词 混合专家模型；边缘算力网络；协作推理；序列并行；专家交换

中图法分类号 TP393

DOI: 10.7544/issn1000-1239.202660131 **CSTR:** 32373.14.issn1000-1239.202660131

凭借动态稀疏激活机制，混合专家模型（mixture of expert, MoE）架构已成为扩展大语言模型（large language model, LLM）的广泛认可范式，并被众多前沿模型采用（如 DeepSeek-V3[1]、Qwen3[2]、Kimi-K2[3]）。这些基于 MoE 的模型在复杂语言处理、对话应用以及新兴的智能体场景中展现出卓越性能，正在重塑机器智能。

尽管 MoE 架构最初诞生于数据中心，但基于 MoE 的 LLM 已快速下沉至边缘设备，并在网络边缘催生了一系列智能应用，如聊天机器人和智能家居 AI 智能体[4-5]。智能个人助手领域的一项近期调研[6]显示，超过 80% 的专家认为这些应用不应完全依赖云端服务。受访者主要关注 3 个方面：1）将敏感用户数据传输至商业云基础设施所带来的潜在隐私风险；2）远程 MoE 服务的显著时延，特别是在网络连接受限的边缘环境中；3）使用云端托管 MoE API 服务所产生的可观运营成本。因此，在本地边缘环境中自托管这些基于 MoE 的 LLM 受到越来越多关注，包括个人电脑、智能手机以及边缘计算平台（如 Raspberry Pi 和 NVIDIA Jetson 系列）[7-10]。

然而，由于 MoE 模型的参数量巨大，在单一边缘设备上部署 MoE 模型面临严峻挑战。即使在云数据中心内，前沿的开源 MoE 模型通常也需要多块高端 GPU（如 NVIDIA H100）以支持高效推理。例如，Mixtral-8×7B[11] 包含 46.7B 参数。即使采用 INT4 量化，仍需要近 24 GB 内存以容纳模型权重，这一需求超出了大多数边缘平台的内存容量。该约束进一步加剧，因为边缘设备还需运行操作系统和其他移动应用，从而压缩了可用内存预算。

为突破内存墙，近期的端侧 MoE 服务系统采用了一种类似操作系统的内存交换策略[7-10,12]，称为专家交换（expert swapping）。这些系统将注意力模块和少量频繁访问的专家保留在主内存中，同时将剩余的冷专家从速度快但容量有限的主内存卸载至二级存储（如磁盘）。在 MoE 推理过程中，由于专家被稀疏激活，系统仅按需将激活的专家从二级存储加载至主内存。该策略在逐 Token 的自回归解码阶段有效，因为每个解码步骤仅激活少量专家。然而，该策略在预填充（prefilling）阶段效果不佳。随着 LLM 支持更长的上下文窗口（如 DeepSeek-V3[1] 最大长度为 128 000 Tokens），用户提供的输入也相应变长（如 Mooncake-Trace[13] 平均输入长度为 12 000 Tokens）。扩展的输入序列导致所有专家被激活，需要将所有专家交换至主内存进行推理，由此产生的 I/O 时延构成显著的性能瓶颈，导致首词时延（time to first token, TTFT）明显延长。

本文观察到，典型的边缘场景（如智能家居、智能工厂）中广泛存在多类可信终端设备，如平板电脑、笔记本电脑及各类家庭智能设备。这些设备通过局域网或近场无线链路彼此互联，在物理上与主推理设备紧密毗邻，共同构成一张广泛分布的边缘算力网络（edge computility network, ECN）。从算力网络视

角审视,这张泛在算力网络中的每个节点均携带着未被充分利用的算力、内存容量与 I/O 带宽资源。这使得将多个边缘节点的集体计算能力、内存容量和 I/O 带宽加以汇聚与协同调度成为可能,从而以分布式方式支撑高效的端侧 MoE 推理,有效加速预填充阶段并降低 TTFT。然而,在多边缘设备协作环境中实现高效的 MoE 推理仍面临若干关键挑战:

1)注意力模块的序列并行通信开销。生成式 LLM 采用因果解码器架构,其单向注意力掩码要求每个输入 Token 关注所有前置 Token。在序列并行推理中,设备间需要交换键值矩阵以确保计算正确性,这引入了额外的通信开销。传统的连续序列划分方式导致严重的负载不均衡——在因果注意力计算中,部分设备的工作负载被掩码完全覆盖而无需计算,而其他设备则需处理完整负载,造成边缘资源利用率低下。

2)MoE 模块的 Token 分发与 I/O 带宽利用。在多设备协作场景下,Token 分发策略直接影响系统性能。一种直观的数据中心(data-centric, DC)范式让每个设备独立处理其本地子序列,但可能导致同一专家被重复换入多个设备,造成冗余的 I/O 带宽消耗。另一种专家中心(expert-centric, EC)范式为每个设备分配固定专家子集,但专家流行度的不均匀分布可能导致严重的负载不均衡,产生拖尾设备。如何在充分利用集体 I/O 带宽的同时实现计算负载均衡,成为降低 MoE 层推理时延的关键挑战。

3)专家交换与计算的串行依赖。在采用专家交换的 MoE 推理中,数据依赖迫使专家计算必须等待专家权重换入完成。与密集模型不同,MoE 模型在每一层根据当前层输出动态选择激活的专家,系统必须等待当前层计算完成以及确定专家选择结果后,才能开始加载下一层所需的专家。这种层间的动态依赖使得传统的预取策略难以直接应用,I/O 时延成为推理系统的潜在瓶颈。

针对上述挑战,本文提出 Sirius,一种面向边缘算力网络的资源高效协作 MoE 推理系统。Sirius 采用序列并行推理架构,将输入序列划分为多个子序列,在多个边缘设备上并行处理,从而加速预填充阶段并降低 TTFT。具体而言,本文设计了 3 项关键技术:1)基于交错划分的序列并行通信优化,通过创新的序列划分机制解决因果注意力计算中的工作负载不均衡问题,实现通信-计算的高效重叠;2)自适应 Token 调度策略,通过动态调度 Token 的设备分配,在充分利用集体 I/O 带宽的同时实现计算负载的均衡分配;3)推测性专家预取机制,利用相邻层隐藏状态的高相似性提前预测专家激活结果,实现计算、调度与 I/O 的并行执行,有效缓解专家交换带来的 I/O 开销。本文的主要贡献总结如下:

1)提出了 Sirius,一种面向边缘算力网络的资源高效协作 MoE 推理系统。Sirius 采用序列并行推理架构,协调边缘算力网络中多个节点的集体计算能力、内存容量和 I/O 带宽,实现高效的分布式 MoE 推理,有效降低预填充阶段的 TTFT。

2)采用了基于交错划分的序列并行策略,将输入序列重新组织为交错子序列,确保每个设备在每个计算迭代中处理均匀分布的 Token 子集,从而实现近似均衡的工作负载分配,最大化通信-计算重叠效率。

3)提出了自适应 Token 调度策略,在决定 Token 放置时联合考虑计算开销、交换时延和专家缓冲区的当前缓存状态,通过贪心启发式算法高效求解最小化最大设备时延的优化目标,充分发挥多设备的集体 I/O 带宽优势。

4)设计了推测性专家预取机制,利用 Transformer 架构中相邻层隐藏状态的高相似性,在当前层计算执行期间提前预测下一层的专家激活结果并进行预取,实现计算-调度- I/O 的流水线并行执行,显著缩短关键路径上的等待时间。

5)在真实边缘算力网络平台上,针对 Mixtral-8x7B 与 Qwen3-30B 两种模型开展系统评估,实验结果表明 Sirius 相比现有基线方法实现了 1.2~4.0 倍的推理吞吐量提升。

1 相关工作

凭借稀疏激活特性,MoE 模型在端侧应用中展现出独特优势。与稠密 Transformer 模型不同,MoE 模型在推理过程中并不激活全部参数,而是依据输入 Token 的特征动态激活少量最相关的专家,显著降低了计算量与内存消耗,使其天然契合资源受限的边缘设备部署需求。然而,边缘部署 MoE 模型面临参数规模与设备资源受限之间的深层矛盾:前沿 MoE 模型的参数量往往达到数十至数百亿规模,即使经过量化压缩也难以完整驻留于单一边缘设备的内存中。此外,如何在多设备协作场景下充分协调集体资源并维持高效推理,亦是一个具有挑战性的研究问题。本节将介绍与上述问题相关的若干代表性工作,主要涵盖端侧 MoE 推理系统(1.1 节)与云数据中心 MoE 推理并行架构(1.2 节)两个方向。

1.1 端侧 MoE 推理系统

随着边缘智能应用的快速兴起以及 MoE 架构在效率上的内在优势，如何在单一端侧设备上高效部署 MoE 模型已成为一个备受工业界和学术界关注的研究方向。MoE 模型的运行时内存开销显著，一类工作从模型压缩角度出发，通过量化技术降低内存需求并加速推理。D^2MoE[10] 提出了 Matryoshka 权重量化（Matryoshka weight quantization, MWQ）方法，以嵌套位宽的方式对专家权重进行渐进式压缩，并引入“热专家优先”（hottest expert bit first）启发式调度策略，在受限内存预算下通过优化 I/O 与计算重叠流水线最大化专家并行度，有效消除 I/O 等待引起的时间气泡。Lu 等人[14] 观察到 MoE 模型各专家对推理精度的贡献并不均等，提出了专家级稀疏性（expert level sparsity）机制，在推理运行时动态识别并跳过对精度影响较小的专家，在减少计算量的同时维持推理质量。PowerInfer[15] 则从神经网络计算稀疏性出发，识别并保留高频激活的神经元，将低频神经元的计算卸载至 CPU 侧执行，通过设计 GPU-CPU 混合推理引擎，在消费级硬件上实现了加速推理。

然而，对于参数规模更大的 MoE 模型，即便经过量化压缩，单一边缘设备的内存仍难以容纳完整权重。为此，另一类工作采用了专家交换机制，类似操作系统的内存分页策略，将非活跃专家卸载至外部存储，按需换入。EdgeMoE[8] 利用存储层次结构，将非专家权重驻留于设备内存，仅在专家被激活时从外部存储动态加载；同时依据各专家对推理精度的重要性评估差异化地选择量化位宽，在降低 I/O 开销的同时减少精度损失。Kong 等人[7] 同样发现专家重要性存在显著差异，提出维护工作参数集，将工作参数集之外的专家参数卸载至磁盘，并通过周期性更新工作参数集成员以适应不同输入带来的专家访问模式变化。Fang 等人[16] 进一步观察到，慢速存储介质上的 I/O 延迟是基于卸载方案的核心瓶颈，设计了 I/O 与计算的重叠执行机制，通过推测性预加载后续可能激活的专家来有效隐藏 I/O 时延。但是，上述工作均受限于单台设备的算力、内存容量与 I/O 带宽，在处理长输入序列时面临较大的时延压力。若能充分挖掘局域网中邻近边缘设备的集体资源，则可突破单台设备的资源瓶颈。这促使本文利用邻近边缘设备的集体计算能力、内存容量与 I/O 带宽，以分布式方式协调多设备，实现高效的端侧 MoE 推理。

1.2 云数据中心分布式 MoE 模型并行推理架构

MoE 架构模型在云数据中心的大规模部署催生了多种分布式推理并行方案。为提升吞吐量、降低推理延迟并充分利用 GPU 集群资源，学术界与工业界提出了多类并行架构。Google 团队在 GPipe[17] 中提出流水线并行方案，在 Transformer 模型层间进行划分，将整个模型的计算拆解为多个流水线阶段，各节点仅负责其中 1 个阶段的计算，节点间仅需传递中间激活值，通信开销可以被各流水线阶段的计算过程重叠掩盖起来。微软团队的 Megatron-LM[18] 提出张量并行方案，将单个神经网络层的参数切分至多台机器并行计算，各机器的计算结果不完整，需在每次计算完成后进行一次结果同步；张量并行能显著降低各节点的内存占用，但引入了较高的通信开销，通常应用于具备高速节点内互联的场景。新加坡国立大学团队提出序列并行方案，每台设备维护完整的模型副本，将输入序列切分为多个子序列并行推理，但每个解码层均须执行 2 次集合通信操作以保证推理结果的正确性。

针对 MoE 架构，Google 团队在 GShard[19] 中率先提出专家并行方案，将专家分配至不同设备，通过 2 次全对全（All-to-All）通信完成 Token 的分派与聚合。Tutel[20] 采用混合并行架构，针对不同规模与结构的 MoE 模型设计了自适应切换机制，可动态选择最优的并行策略组合，在多样化的模型架构与推理负载下均保持较高效率。DeepSeek 团队推出了大规模专家并行架构[1]，令每张 GPU 仅负责极少量专家的计算，通过极细粒度的专家并行追求低延迟、高吞吐的推理性能。

然而，专家并行方案在实践中面临 2 个突出挑战。其一是 All-to-All 通信开销过大的问题：传统专家并行采用以专家为中心的范式，将专家固定于各设备，通过 All-to-All 通信在设备间传输中间激活数据以驱动专家计算；由于激活数据体量往往大于专家参数本身，通信开销随并行规模的扩大急剧攀升。对此，Janus[21] 提出以数据为中心的新范式，改为将激活数据固定于各设备，转而在 GPU 间迁移专家参数，从而将通信量压缩至专家参数规模，有效降低 All-to-All 通信负载。其二是负载不均衡问题：门控网络的动态路由决策使不同专家在同一批次内接收到的 Token 数量差异悬殊，导致部分设备长时间空闲等待，严重削弱集群的整体利用率。SmartMoE[22] 针对这一问题，构建了基于负载感知的性能建模框架，并结合在线自适应并行化机制，在离线阶段预先构建由若干可相互转换并行策略组成的策略池，在运行时依据当前实际负载分布从策略池中快速选取最优策略，

实现并行方案的动态适配。

上述数据中心并行方案依托高速互联网络与充裕的 GPU 内存资源设计，难以直接迁移至带宽受限、设备异构的边缘环境。本文针对边缘场景的特点，在序列并行策略、Token 调度机制以及专家预取方案 3 个层面进行了系统性重新设计，以适应多边缘设备协作推理的约束条件。

2 研究背景与动机

2.1 基于 MoE 架构的 LLM 基础

1)基于 Transformer 的稠密 LLM。当前自然语言处理(natural language processing, NLP)的机器学习应用日益依赖基于 Transformer 的架构[23]，该架构由多个堆叠的 Transformer 层组成，因其卓越的准确性而广泛应用。如图 1(a)所示，标准密集 Transformer 层的架构。每层主要由多头注意力(multi-head attention, MHA)模块和前馈网络(feed-forward network, FFN)模块组成，通过逐元素操作(包括 Dropout、残差连接和层归一化)相互连接。在 MHA 模块内，初始线性投影为每个注意力头生成查询(Query, Q)、键(Key, K)和值(Value, V)表示。每个头独立执行自注意力，所得输出被拼接并通过最终线性投影形成层输出。FFN 模块应用 2 次连续的线性变换，首先将隐藏维度扩展至更高的中间尺寸，然后投影回原始维度。

2)基于 MoE 的稀疏 LLM。基于 Transformer 的密集 LLM 的成功主要源于通过更深的 Transformer 层堆叠来扩展模型规模。然而，进一步增加模型容量的关键挑战是计算成本的快速增长。为解决该问题，MoE 架构利用结构稀疏性降低 LLM 的计算需求。通过仅为每个 Token 激活少量参数子集，MoE 在扩展模型规模的同时避免浮点运算(floating point operation, FLOP)的成比例增长。如图 1(b)所示，稀疏 MoE 层的架构，通过将图 1(a) 中密集层的原始 FFN 模块替换为 MoE 模块而得到。将 FFN 视为一个专家，该模块包含多个稀疏激活的专家，如图 1(c)所示。添加可训练的门控网络，动态地将每个 Token 与所有专家中的 Top-K 个专家匹配。记$G(x)$为输入 Token x的门控网络输出，$G(x)_i$为分配给第i个专家网络的门控权重，$E_i(x)$为第i个专家网络的输出，$\mathcal{T}$为选定的 Top-K 索引集合。输入 Token x的 MoE 层输出y可表示为：

$$y=\sum_{i\in\mathcal{T}}G(x)_iE_i(x)\ 。\tag{1}$$

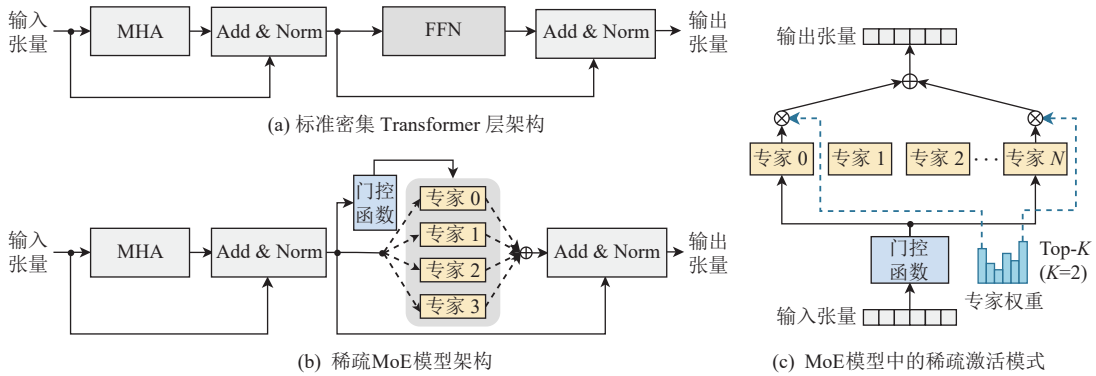


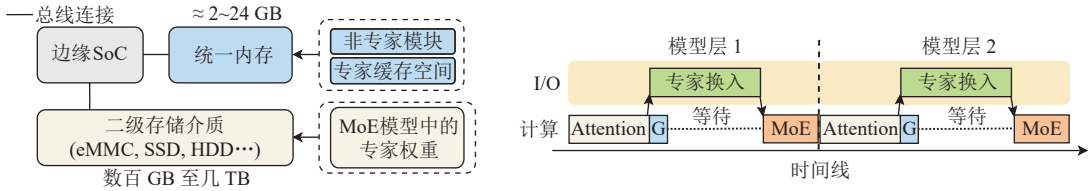
Fig. 1 Architectural comparative diagram between dense Transformer and sparse MoE model

图 1 稠密 Transformer 与稀疏 MoE 模型架构对比图

2.2 资源受限边缘设备上的 MoE 推理

1)采用专家交换的 MoE 推理。设备端 MoE 推理并非易事。MoE 模型包含极大量参数，其模型权重内存占用往往超出单一边缘平台的容量，而边缘平台的总内存通常在 2~24 GB，且还需容纳操作系统和其他移动应用。该挑战因需要在解码阶段为键值(key-value, KV)缓存预留额外内存预算而进一步加剧。例如，Mixtral-8×7B 模型包含 46.7B 参数，即使采用 INT4 量化，仍需要近 24 GB 内存以存储权重。在这些参数中，MoE 模块单独贡献约 46.1B，占总量的 98% 以上，而其余组件(包括多头注意力和门控函数)合计仅占约 2%。

为突破内存限制，近期端侧 MoE 推理系统借鉴操作系统虚拟内存机制，将非活跃专家参数卸载至二级存储(如固态硬盘)，并根据激活情况按需换入，该方法称为专家交换。图 2(a) 展示了采用专家交换的存储层次结构：非 MoE 模块常驻主内存，参数密集的 MoE 模块存储于二级存储，系统在主内存中维护专家缓冲区以保留少量热门专家，从而减少换入开销。图 2(b) 为采用专家交换的 MoE 推理的执行



(a) 基于专家交换的MoE推理系统存储架构图 (b) 基于专家交换的MoE推理执行时间线

Fig. 2 Storage architecture diagram and execution timeline of swapping-based MoE inference system

图 2 基于交换的 MoE 推理系统存储架构图及执行时间线

时间线。在执行 MoE 模块之前，必须将激活的专家交换至主内存。为降低专家交换入的 I/O 开销，部分近期研究[7]跳过对精度贡献可忽略的专家，从而减少必须加载至主内存的专家数量。然而，识别可跳过专家仍是一个开放挑战。错误选择会显著降低推理精度。文献 [9] 设计了预取机制，如专家激活路径利用跨层专家选择频率的相关性。通过预取可能激活的专家，这些方法将计算与通信重叠以降低开销。

2）采用专家交换的 MoE 推理的局限性。上述专家交换优化在逐 Token 自回归解码阶段有效，该阶段每个步骤仅激活少量专家。例如，在包含 128 个专家且采用 Top-2 激活的 MoE 模型中（排除推测采样），每次推理每层最多需要交换 2 个专家。结合上述 I/O 优化技术，仍能维持低时延推理。然而，专家交换策略在预填充阶段效果显著降低。现代 LLM 支持极大的上下文窗口（例如，DeepSeek-V3 支持最多 128 000 Tokens 的序列，Kimi-K2 扩展至 256 000 Tokens），用户查询也相应变长（如 Mooncake-Trace 平均输入长度为 12 000 Tokens）。如此长的输入序列几乎激活每层的所有专家，迫使系统将每个专家从二级存储交换至主内存进行推理。这使得无法通过选择性交换激活的专家来利用 MoE 的稀疏激活特性，因为所有专家均被触发。由此产生的全量加载生成大量的 I/O 流量，形成显著的性能瓶颈，并大幅增加 TTFT。

3）机会与设计目标。面对预填充阶段的 I/O 瓶颈，单一边缘设备受限于有限的内存容量与 I/O 带宽，难以进一步优化。边缘场景的特点为突破该瓶颈提供了新思路：实际部署中往往存在多台物理邻近的可信设备[24-27]，这些设备通过局域网互联构成分布式算力网络。本节利用这些邻近设备的闲置资源，将其计算能力、内存容量与 I/O 带宽汇聚协同，通过跨设备并行推理缓解单点资源约束，从而降低预填充阶段的 TTFT。基于上述思路，系统设计遵循以下目标：

①在多个边缘设备间并行化单序列推理，以降低预填充阶段的 TTFT①。预填充阶段需处理完整输入序列，计算量随序列长度线性增长，构成 TTFT 的主要来源。通过序列并行将输入切分至多台设备协同处理，可有效分摊计算负载。与云端批处理服务不同，边缘场景通常仅服务单个或少量并发请求，系统需针对单请求时延而非批处理吞吐量进行优化。

②支持高效的专家交换，充分利用协作边缘设备的集体内存容量和 I/O 带宽。前沿 MoE 模型参数规模达几十至数百 GB，即使汇聚多台边缘设备的内存仍难以完整容纳，专家交换机制不可或缺。分布式环境下需解决专家在多设备间的分布与调度问题：既要避免同一专家在多个设备重复换入造成 I/O 浪费，又要防止专家访问倾斜导致的负载不均。系统需兼顾 I/O 带宽利用率与计算负载均衡，最大化集群资源效率。

③实现计算与 I/O 重叠的细粒度优化，以最小化开销。分布式推理产生通信开销，专家交换增加交换入成本，这些可能成为性能瓶颈。系统必须在细粒度上协调计算与 I/O，重叠操作并维持低时延推理。

3 面向 MoE 模型的边缘协同推理系统

3.1 并行架构概览

本文提出 Sirius，一种面向边缘算力网络的混合专家模型协同推理系统。如图 3 所示，Sirius 采用序列并行推理架构，将输入序列划分为多个子序列，在多个边缘设备上并行处理。

1）MHA 模块的并行设计。由于本文关注降低长序列预填充阶段的首次 Token 时延，因此采用序列并行推理架构处理注意力模块。如图 3 所示，输入序列被划分为 n 个等长的子序列：$(s_0, s_1, \cdots s_{n-1})$。以双设备为例，设备 0 处理 $s_0 = \{t_0, \cdots t_{i-1}\}$，设备 1 处理

① 本文所提出的系统架构和优化设计，同样兼容解码阶段，无需对核心机制进行实质性改动，即可对 Token 间时延（time between Tokens, TBT）产生显著的优化效果。

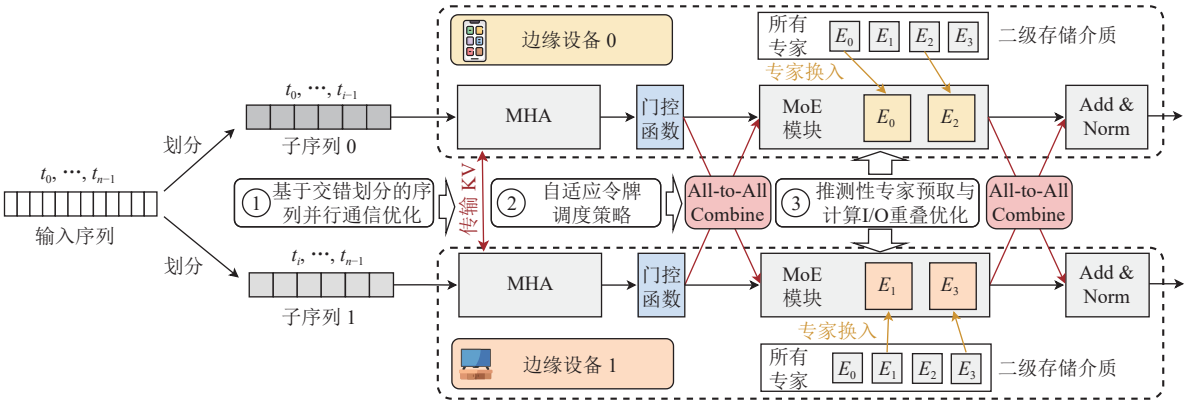


Fig. 3 System architecture of Sirius

图 3 Sirius 系统架构图

$s_1 = \{t_i, \cdots t_{n-1}\}$, 每个设备持有完整的 MHA 模块权重。生成式 LLM 通常采用因果解码器和单向注意力掩码, 这要求每个输入 Token 在计算注意力时关注所有前置 Token, 这在序列并行中引入跨设备数据依赖。以设备 1 为例, 计算 s_1 的注意力输出时需访问 s_0 的键值矩阵, 这种数据依赖关系引入了额外的通信开销, 迫使设备 1 必须等待设备 0 传输完整键值矩阵后才能开始计算, 产生空闲等待气泡。更严重的是, 因果掩码导致不同设备的计算量差异悬殊, 后序设备的 MHA 机制计算规模远大于前序设备, 形成负载倾斜。针对该问题, 本节设计基于矩阵分块的交错划分策略, 通过重组序列分配模式与细粒度的通信-计算流水线化, 实现负载均衡与延迟隐藏(图 3 中①), 详见 3.2 节。

2)MoE 模块的并行设计。每个子序列完成 MHA 计算后, 每个 Token 通过门控函数, 动态选择 Top-K 个专家进行激活。随后, 通过 All-to-All 集合通信将 Token 分发到负责其计算的边缘设备。每个设备在本地二级存储中维护完整的专家集合, 并根据其分配 Token 所需的专家, 将选中的专家交换到主内存执行, 当专家已在专家缓冲区中缓存时跳过传输。Token 由分配的专家处理后, 结果需要发送回其原始设备, 这需要再次进行 All-to-All 通信。然而, 上述工作流中仍存在 2 个关键问题:

1)Token 分发面临关键挑战, 因为次优分配可能阻碍计算和 I/O 带宽资源的充分利用。一种直观的数据中心范式将每个边缘设备分配执行其本地子序列的 MoE 模块, 从二级存储中将由该子序列激活的专家换入到内存中。该方法在设备间平衡计算负载, 但可能导致同一专家被换入到多个设备, 造成冗余的 I/O 带宽消耗。另一种专家中心范式为每个设备分配固定、非重叠的专家子集进行推理, 通过 All-to-All 通信将 Token 分发到负责其指定专家的设备。该范式避免了同一专家在多个设备内被重复换入, 但专家流行度的不均匀可能导致负载不均衡: 如果边缘设备被分配过多频繁激活的专家, 其完成推理的时间可能显著更长, 成为落后设备。为解决该问题, 本文提出一种自适应 Token 分发策略, 在决定 Token 放置时联合考虑计算开销、交换时延和专家缓冲区的当前缓存状态, (图 3 中②)详见 3.3 节。

2)专家交换可能引入大量 I/O 开销, 使 I/O 成为推理系统的潜在瓶颈。如图 2(b)所示, 数据依赖迫使 MoE 推理等待专家交换。即使多个边缘设备的集合 I/O 带宽得到充分利用, 延长的交换时间仍可能显著增加推理时延。本文观察到, 在模型推理过程中, 从二级存储到主内存的 I/O 路径仍然利用不足, 因此引入专家预取(expert prefetching)策略, 将计算与专家交换重叠以缓解 I/O 开销(图 3 中③), 详见节 3.4。

3.2 基于交错划分的序列并行通信优化

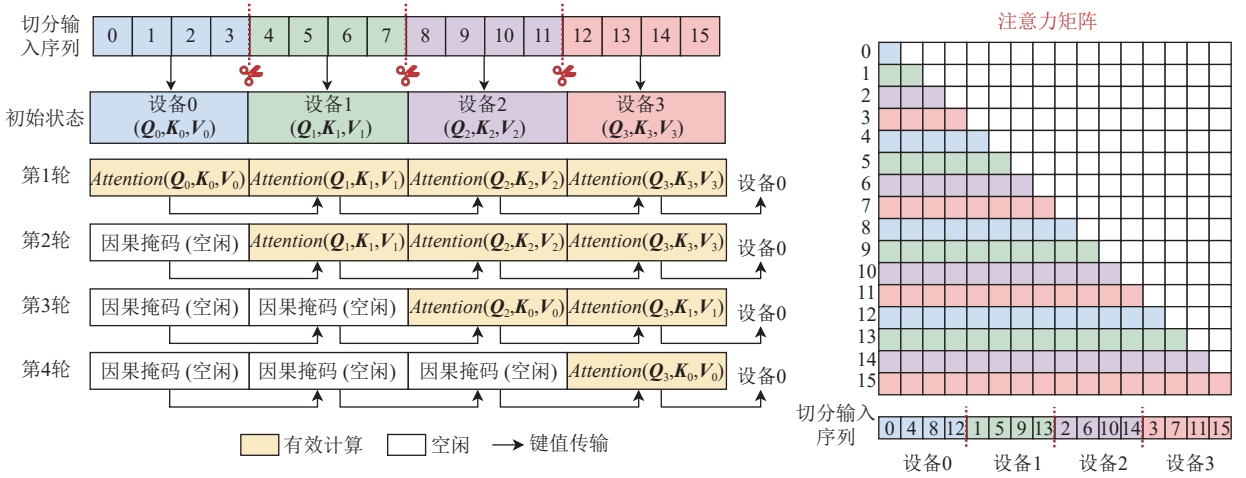
本文采用了一种基于交错划分(interleaved partitioning)的序列并行策略[28-29], 旨在实现多边缘设备协同的注意力模块高效推理。该策略通过创新的序列划分机制, 有效解决因果注意力计算中的工作负载不均衡问题, 从而最大化通信-计算重叠效率, 显著降低预填充阶段的首词时延。

在多设备序列并行推理中, 常规方案是将输入序列划分为 n 个连续的子序列 $s_0, s_1, \cdots, s_{n-1}$(其中 n 表示参与协作推理的边缘设备数量), 每个设备负责处理一个连续子序列。由于注意力机制要求 Token 间的依赖计算, 子序列独立计算无法保证结果正确性, 需要在设备间交换键和值矩阵以确保分布式计算与本地计算的一致性, 这会产生额外的通信开销。一些前沿的研究工作利用矩阵分块技术[30]、解耦矩阵乘法和键值张量的传输, 并利用环形通信技术实

现高效的计算-通信重叠优化，具体工作流程如图 4(a)所示。然而，生成式 LLM 采用因果解码器架构，其单向注意力掩码机制导致连续划分在分布式执行中产生严重的负载不均衡。具体而言，在环形通信拓扑的后续迭代中，部分设备的工作负载被因果掩码完全覆盖而无需计算，而其他设备则需处理完整的未掩码负载。在极端情况下(如图 4(a)中第 4 轮)，仅设备 3 处于工作状态，其余 3 台设备均处于空闲状态。由于系统性能受限于最慢设备的完成时间，这种负载不均衡导致边缘资源利用率低下，严重制约了整体性能。为解决该问题，本文采用了一种交错划分策略，将长度为 L 的输入序列 $\mathcal{T}=\{t_0,t_1,\cdots,t_{L-1}\}$ 重新组织为 N 个交错子序列 $\mathcal{T}_0,\mathcal{T}_1,\cdots,\mathcal{T}_{N-1}$，其中每个子序列 $\mathcal{T}_i$ 包含所有满足 $j \bmod N = i$ 的 Token t_j。形式化地，该划分可表示为：

$$\mathcal{T}_i=\{t_j \mid j \bmod N = i, 0 \leqslant j < L\}。 \tag{2}$$

相应地，注意力机制中的查询矩阵 $\boldsymbol{Q}\in\mathbb{R}^{L\times d}$、键矩阵 $\boldsymbol{K}\in\mathbb{R}^{L\times d}$ 和值矩阵 $\boldsymbol{V}\in\mathbb{R}^{L\times d}$(其中 d 表示隐藏层维度)被划分为 N 个块，每个块的大小为 $c=\lceil L/N\rceil$。这种划分方式确保每个设备在每个计算迭代中都处理到在整个原始序列中均匀分布的 Token 子集，从而实现近似均衡的工作负载分配①。



(a) 基于环状通信的注意力机制计算流程　　(b) 基于交错划分的序列并行注意力分布情况

Fig. 4　Sequence parallelism communication optimization based on interleaved partitioning

图 4　基于交错划分的序列并行通信优化

1)基于交错划分的环形序列并行机制。在完成序列的交错划分后，本文采用环形的序列并行通信拓扑进行分布式注意力计算(环形工作流程与图 4(a)一致)。具体而言，系统将 N 个边缘设备组织为环形结构，每个设备 j 初始持有其分配的查询块 $\boldsymbol{Q}_j$、键块 $\boldsymbol{K}_j$ 和值块 $\boldsymbol{V}_j$。在推理过程中，查询块 $\boldsymbol{Q}_j$ 在各设备上保持静止，而键值块 $(\boldsymbol{K},\boldsymbol{V})$ 在环形拓扑中按顺序传递。在第 i 轮通信迭代中，设备 j 计算其本地查询块 $\boldsymbol{Q}_j$ 与当前接收到的键值块 $\boldsymbol{K}_{(j-i)\bmod N}$ 和 $\boldsymbol{V}_{(j-i)\bmod N}$ 之间的注意力交互，同时将键值块异步传递给下一个设备以隐藏通信时延。经过 N 轮迭代后，每个设备累积得到其最终的注意力输出。在交错划分的序列并行框架下，最终的注意力输出可以通过式(3)计算得到。假设设备 $j(0\leqslant j<N)$在第 i 个通信迭代中处理来自设备 $k=(j-i)\bmod N$ 的键值块，则设备 j 的最终输出为：

$$Out_j=\sum_{i=0}^{N-1} Attention(\boldsymbol{Q}_j,\boldsymbol{K}_{(j-i)\bmod N},\boldsymbol{V}_{(j-i)\bmod N},\boldsymbol{M}_{j,(j-i)\bmod N}), \tag{3}$$

其中 $\boldsymbol{Q}_j,\boldsymbol{K}_j,\boldsymbol{V}_j$ 分别表示设备 j 持有的查询、键、值矩阵块，$\boldsymbol{M}_{j,k}$ 表示设备 j 处理来自设备 k 的键值块时的因果掩码矩阵，其定义为：

$$\boldsymbol{M}_{jk}[p,q]=\begin{cases}0, & \text{if } original_pos(j,p)\geqslant original_pos(k,q),\\ -\infty, & \text{其他}。\end{cases} \tag{4}$$

这里 $original_pos(d,idx)$ 表示设备 d 上局部索引为 idx 的 Token 在原始序列中的全局位置，即 $original_pos(d,idx)=d+idx\times N$ 这种掩码构造确保了

① 本工作将输入序列均匀地划分到 N 个边缘设备上，适用于同构边缘集群环境，但在异构环境中可能并非最优。尽管如此，本文算法可按照计算量动态调整输入序列切分点，轻松拓展至异构场景。

因果注意力机制的正确性,同时允许在环形通信拓扑中高效地进行分布式计算。在实际部署中,交错划分通过在模型前向传播开始时对输入序列进行一次性置换来实现。对于使用旋转位置编码(rotary position embedding, RoPE)的模型,相应的位置标识符也需进行同步置换。该置换操作仅在预填充阶段执行1次,不会引入显著的额外开销,却能带来显著的性能收益。

2)交错划分下的负载均衡特性。交错划分策略的核心优势在于实现设备间的工作负载均衡。由于每个设备持有的Token均匀分布在原始序列的不同位置,在因果注意力计算中,每个设备需要处理的有效(未被掩码)注意力交互数量相对接近,如图4(b)所示。具体而言,设备 i 在处理其查询块与来自设备 j 的键值块时,无论2设备的索引关系如何,其需要计算的注意力交互数量均大致为完整块交互量的一半,仅在是否包含对角线元素上存在微小差异。

这种均衡的负载分配与传统的连续划分方式形成鲜明对比。在连续划分中,每轮迭代至少存在1个设备需要处理完全未被掩码的注意力块,而其他设备的计算则可能被完全掩码跳过,导致严重的负载不均衡。由于分布式系统的推理时延由最慢设备决定,连续划分方式无法充分利用因果掩码带来的计算节省。而交错划分确保每个设备在每轮迭代中都承担大致相等的计算负载,从而有效缩短了整体推理时延,显著提升了边缘计算资源的利用效率。

3.3 自适应Token调度策略

在完成注意力模块的序列并行推理后,系统进入MoE层的执行阶段。由于边缘设备的内存资源受限,难以将所有专家权重常驻内存,本文采用专家交换机制,将专家参数存储于本地二级存储介质(如固态硬盘或闪存),并按需换入主内存执行。然而,如何在多设备协作推理环境中高效调度Token与专家的映射关系,充分利用集体I/O带宽,成为降低MoE层推理时延的关键挑战。

3.3.1 专家交换推理的I/O瓶颈

在序列并行框架下,注意力模块计算完成后,每个边缘设备持有一个子序列的中间表示。随后,每个设备独立执行门控函数,计算其子序列中各Token的专家分配结果。一种直观的执行方式是令每个设备在本地完成其子序列的MoE模块推理:设备从本地二级存储中换入该子序列所激活的全部专家,执行专家计算后输出结果。然而,该方案存在显著的I/O效率问题。由于不同子序列可能激活相同的热门专家,多个设备将独立地从各自的二级存储中换入同一专家的权重副本,导致集体I/O带宽的冗余消耗。在MoE模型的边缘推理中,专家权重从二级存储换入主内存的I/O开销占据主导地位,因此充分发挥多设备的集体I/O带宽对于降低推理时延至关重要。当热门专家被重复换入多个设备时,系统无法有效利用分布式I/O资源,造成严重的带宽浪费。另一种替代方案是采用专家中心的静态分配策略:为每个设备预先分配固定且互不重叠的专家子集,通过All-to-All通信将Token分发至负责其所需专家的设备。该方案避免了同一专家被重复换入多个设备的问题,但专家流行度的不均匀分布可能导致严重的负载不均衡,即持有热门专家的设备将处理远多于其他设备的Token,成为系统的拖尾设备,制约整体推理性能。

3.3.2 自适应Token分发策略

为解决上述问题,本文提出自适应Token调度(adaptive Token scheduling)策略,通过动态调度Token的设备分配,在充分利用集体I/O带宽的同时实现计算负载的均衡分配。

1)专家缓存区与缓存机制。每个边缘设备在主内存中维护一个专家缓存区,用于缓存近期使用的专家权重。该缓存区的容量在MoE模型的各层间均匀分配,采用最近最少使用(least recently used, LRU)策略进行淘汰更新。当某专家已驻留在目标设备的缓存区中时,系统可直接复用缓存的权重而无需执行I/O操作,从而显著降低换入时延。

2)门控函数与专家选择。在MoE层中,每个Token通过门控网络动态选择其激活的专家子集。记MoE层包含 M 个专家 $\mathcal{E}=\{e_0,e_1,\cdots,e_{M-1}\}$,对于输入Token t_i,门控网络计算其对所有专家的亲和度分数 $\boldsymbol{G}(t_i)\in\mathbb{R}^M$,并选取分数最高的Top-$K$个专家构成激活专家集合 $\mathcal{E}_i=TopK(\boldsymbol{G}(t_i))$。记 $E_m(t_i)$ 为第 m 个专家对Token t_i 的输出,则MoE层对该Token的输出 y_i 为各激活专家输出的加权和:

$$y_i=\sum_{m\in\mathcal{E}_i}G(t_i)_m\cdot E_m(t_i)\,, \tag{5}$$

其中 $G(t_i)_m$ 为门控网络分配给专家 e_m 的归一化权重。该稀疏激活机制使得每个Token仅需 K 个专家参与计算,而非全部 M 个专家,从而在扩展模型容量的同时控制计算开销。

3)问题建模。基于上述门控机制,记 $\mathcal{D}=\{d_0,d_1,\cdots,d_{N-1}\}$ 为参与协作推理的 N 个边缘设备集合。设长度为 L 的输入序列经各设备独立执行门控函数

计算后，产生 Token-专家分配集合 $\mathcal{A}=\{(t_i,\mathcal{E}_i)\mid i=0,1,\cdots,L-1\}$，其中 $\mathcal{E}_i\subseteq\mathcal{E}$ 且 $|\mathcal{E}_i|=K$ 表示 Token t_i 激活的 Top-K 个专家。记 $C_j\subseteq\mathcal{E}$ 为设备 d_j 的专家缓冲区中当前缓存的专家集合。

Token 分发的目标是确定映射函数 ϕ: $\{t_0,t_1,\cdots,t_{L-1}\}\to\mathcal{D}$，将每个 Token 分配至某一边缘设备执行。对于设备 d_j，记其被分配的 Token 集合为 $\mathcal{T}_j=\{t_i\mid\phi(t_i)=d_j\}$，则该设备需要执行的专家集合为 $\mathcal{E}_j^{\text{req}}=\bigcup_{t_i\in\mathcal{T}_j}\mathcal{E}_i$ 需要从二级存储换入的专家集合为 $\mathcal{E}_j^{\text{swap}}=\mathcal{E}_j^{\text{req}}\setminus C_j$。设备 d_j 完成其分配任务的总时延由专家换入时延和专家计算时延 2 部分组成：

$$T_j=T_j^{\text{swap}}+T_j^{\text{comp}}=\frac{|\mathcal{E}_j^{\text{swap}}|\cdot W_{\text{e}}}{B_j}+|\mathcal{T}_j|\cdot K\cdot\tau_{\text{e}},\tag{6}$$

其中 W_{e} 表示单个专家的权重大小，B_j 表示设备 d_j 的 I/O 带宽，τ_{e} 表示单个 Token 经过单个专家的计算时延。由于分布式系统的总体时延由最慢设备决定，优化目标为最小化所有设备中的最大完成时延：

$$\min_{\phi}\max_{j\in\{0,1,\cdots,N-1\}}T_j。\tag{7}$$

4）贪心调度算法。上述优化问题涉及组合优化，精确求解的计算复杂度较高。本文提出一种贪心启发式算法进行高效求解。算法的核心思想是逐个遍历待分配的 Token，对于每个 Token 评估将其分配至各候选设备后产生的累积时延增量，选择使当前最大设备时延最小的分配方案。具体而言，算法维护每个设备的当前状态，包括已分配的 Token 集合 $\mathcal{T}_j$、已确定需要换入的专家集合 $\mathcal{E}_j^{\text{swap}}$ 以及当前累积时延 T_j。对于待分配的 Token t_i，算法遍历所有设备 d_j，计算将 t_i 分配至 d_j 后的时延增量：

$$\Delta T_j(t_i)=\Delta T_j^{\text{swap}}(t_i)+\Delta T_j^{\text{comp}}(t_i),\tag{8}$$

其中计算时延增量 $\Delta T_j^{\text{comp}}(t_i)=K\cdot\tau_{\text{e}}$ 为常数，而换入时延增量取决于 Token t_i 所需的专家是否已在设备 d_j 的缓存或已确定换入集合中：

$$\Delta T_j^{\text{swap}}(t_i)=\frac{|\mathcal{E}_i\setminus(C_j\cup\mathcal{E}_j^{\text{swap}})|\cdot W_{\text{e}}}{B_j}。\tag{9}$$

算法选择使更新后最大设备时延最小的目标设备 $d^*=\arg\min_j\max_k\left(T_k+\mathbb{1}[k=j]\cdot\Delta T_j(t_i)\right)$，并更新相应状态。通过该贪心策略，算法在 $O(L\cdot N)$ 的时间复杂度内完成全部 Token 的分配决策。算法 1 给出了该贪心调度算法的伪代码。

算法 1. 自适应 Token 调度的贪心调度算法。

输入：Token-专家分配集合 $\mathcal{A}=\{(t_i,\mathcal{E}_i)\mid i=0,\cdots,L-1\}$，设备集合 $\mathcal{D}=\{d_0,\cdots,d_{N-1}\}$，各设备缓存状态 $\{C_0,\cdots,C_{N-1}\}$；

输出：Token 到设备的映射 ϕ。

/*初始化各设备状态*/

① for $j=0$ to $N-1$ do

② $\quad\mathcal{T}_j\leftarrow\varnothing,\mathcal{E}_j^{\text{swap}}\leftarrow\varnothing,T_j\leftarrow0$;

③ end for

/*逐 Token 贪心分配*/

④ for $i=0$ to $L-1$ do

⑤ $\quad$计算分配至各设备的时延增量；

⑥ $\quad$for $j=0$ to $N-1$ do

⑦ $\quad\quad\mathcal{E}_{\text{new}}\leftarrow\mathcal{E}_i\setminus\left(C_j\cup\mathcal{E}_j^{\text{swap}}\right)$;

⑧ $\quad\quad\Delta T_j^{\text{swap}}\leftarrow|\mathcal{E}_{\text{new}}|\cdot W_{\text{e}}/B_j$;

⑨ $\quad\quad\Delta T_j^{\text{comp}}\leftarrow K\cdot\tau_{\text{e}}$;

⑩ $\quad\quad\Delta T_j\leftarrow\Delta T_j^{\text{swap}}+\Delta T_j^{\text{comp}}$;

⑪ $\quad$end for

⑫ $\quad d^*\leftarrow\arg\min_j\max_k\left(T_k+\mathbb{1}[k=j]\cdot\Delta T_j\right)$;

⑬ $\quad\mathcal{T}_{d^*}\leftarrow\mathcal{T}_{d^*}\cup\{t_i\}$;

⑭ $\quad\mathcal{E}_{d^*}^{\text{swap}}\leftarrow\mathcal{E}_{d^*}^{\text{swap}}\cup\left(\mathcal{E}_i\setminus(C_{d^*}\cup\mathcal{E}_{d^*}^{\text{swap}})\right)$;

⑮ $\quad T_{d^*}\leftarrow T_{d^*}+\Delta T_{d^*}$;

⑯ $\quad\phi(t_i)\leftarrow d^*$;

⑰ end for

⑱ return ϕ。

5）实际部署时的执行流程。在实际部署中，各设备首先独立执行门控函数计算，获取本地子序列的 Token-专家分配结果。随后，各设备将分配结果及其当前专家缓冲区状态汇报至中心调度节点。中心节点执行上述贪心调度算法，生成全局 Token 分发方案，并将调度结果下发至各设备。最后，各设备通过 All-to-All 通信交换 Token，执行本地专家换入与计算，并将结果返回至 Token 的原始设备。该调度机制确保了多设备间 I/O 负载的均衡分配，避免了热门专家的重复换入，从而充分发挥了集体 I/O 带宽的优势。

3.4 推测性专家预取策略

通过自适应 Token 调度策略，系统可充分利用多台边缘设备的集体 I/O 带宽，实现高效的专家换入。然而，该方案在执行流程上存在严格的数据依赖：专家计算必须等待专家权重换入完成，而专家换入又必须等待自适应 Token 调度算法生成分配方案。这种串行依赖导致关键路径上的计算、I/O 与调度操作无法重叠，使得 GPU 在等待期间处于空闲状态，造成计算资源的浪费并增加推理时延。

计算与 I/O 重叠执行是一个非常通用且效果显著的优化方案。然而，MoE 模型的专家加载与计算难以直接重叠。这是因为 MoE 模型在每一层根据当

前层的输出动态选择激活的专家，系统必须等待当前层计算完成及确定专家选择结果后，才能开始加载下一层所需的专家。这种层间的动态依赖使得传统的预取策略难以直接应用。

1)相邻层激活的高相似性观察。人工智能领域的研究人员观察到，在基于 Transformer 架构的 LLM 推理过程中，相邻两层的隐藏状态具有极高的相似性[31]。设第 l 层的输入隐藏状态为 $h^{(l)}$，第 $l+1$ 层的输入隐藏状态为 $h^{(l+1)}$，则两者之间的余弦相似度通常接近于 1。这一现象的根本原因在于 Transformer 架构中广泛采用的残差连接机制：每一层的输出为输入与该层变换结果的加和，即 $h^{(l+1)}=h^{(l)}+f^{(l)}(h^{(l)})$，其中 $f^{(l)}$ 表示第 l 层的变换函数。由于变换结果 $f^{(l)}(h^{(l)})$ 的幅度相对于输入 $h^{(l)}$ 通常较小，相邻层的隐藏状态呈现高度相关性。如图 5 所示，针对 2 种典型 MoE 架构模型的评估结果揭示了相邻 Transformer 层间门控网络的输入张量具有显著的余弦相似性。这种强烈的层间表征相关性为跨层专家预取机制提供了坚实的启发式依据。

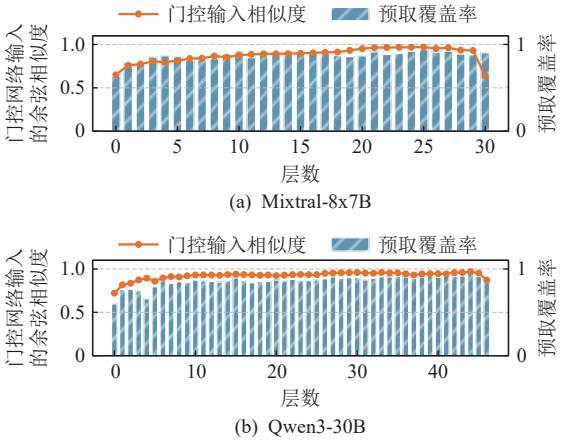


Fig. 5 Feasibility analysis of speculative expert prefetching strategy

图 5 推测性专家预取策略的可行性分析

2)推测性预取机制。基于上述观察，本文采用了推测性专家预取(speculative expert prefetching)机制。核心思想是：利用第 l 层的隐藏状态 $\boldsymbol{h}^{(l)}$ 作为第 $l+1$ 层门控函数的输入，提前预测第 $l+1$ 层的专家激活结果，从而在第 l 层计算执行期间并行完成第 $l+1$ 层的 Token 调度与专家预取。具体而言，在第 l 层的 MoE 模块开始执行时，系统同时将当前层输入 $\boldsymbol{h}^{(l)}$ 送入第 $l+1$ 层的门控函数 $G^{(l+1)}$，计算预测的专家激活集合 $\hat{\mathcal{E}}_i^{(l+1)}=TopK(G^{(l+1)}(\boldsymbol{h}_i^{(l)}))$。基于该预测结果，中心调度节点提前执行自适应 Token 调度算法，生成第 $l+1$ 层的 Token 分配方案 $\hat{\phi}^{(l+1)}$，各设备随即开始预取预测所需的专家权重。当第 l 层计算完成后，系统使用实际的隐藏状态 $\boldsymbol{h}^{(l+1)}$ 执行第 $l+1$ 层的门控函数，获取真实的专家激活集合 $\mathcal{E}_i^{(l+1)}$。若预测结果与真实结果一致，则预取的专家可直接用于计算；若存在偏差，系统对缺失的专家执行补充换入，并相应调整 Token 分配。由于相邻层激活的高相似性，预测准确率通常较高，补充换入的开销可控。

3)计算-调度- I/O 并行执行。对于同时搭载 CPU 与 GPU 的边缘设备，推测性预取机制可充分发挥异构计算资源的并行优势。具体而言，自适应 Token 调度算法在 CPU 上执行，模型层的前向计算在 GPU 上执行，专家权重的预取通过 I/O 子系统完成。三者可在时间上重叠并行：当 GPU 执行第 l 层的专家计算时，CPU 基于预测结果执行第 $l+1$ 层的调度算法，I/O 子系统同时预取第 $l+1$ 层的专家权重。这种流水线式的并行执行显著缩短了关键路径上的等待时间，有效提升了整体推理速度。图 6 展示了推测性预取在 2 种典型场景下的执行流程。其一，在模型层 1 执行过程中，系统完整地预取了第 2 层所需的专家，使得第 2 层完成 Token 分派进入 MoE 层后可以

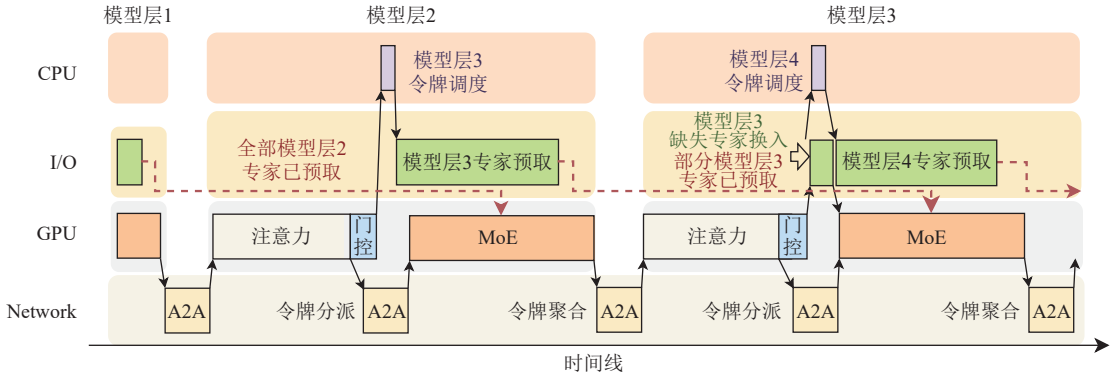


Fig. 6 Computation-scheduling-I/O parallel execution in speculative expert prefetching strategies

图 6 推测性专家预取策略中的计算-调度-I/O 并行执行

直接启动专家计算。其二，在模型层 2 执行过程中，系统预取了大部分第 3 层所需的专家，因此在第 3 层执行过程中，系统仅需对这部分缺失专家补充换入即可继续执行。

为了定量衡量预取机制的有效性，我们定义了预取覆盖率$C = N/N_{\mathrm{A}}$，其中 N 为预取且被实际激活的专家数，N_{A}为实际被激活的专家总数。

如图 5 的预取实验结果所示，对于 MoE 架构的 Transformer 模型的中后层，该机制的预取覆盖率普遍超过 80%。正是得益于大量所需专家被准确地提前加载，系统有效掩盖了专家换入的 I/O 开销。因此，与图 2（b）中展示的全量等待方案相比，上述 2 种情形的整体同步阻塞时延均实现了大幅缩短。

3.5 解码阶段的适用性分析

3.2、3.3、3.4 等 3 个章节的 3 项设计以预填充阶段的首词时延优化为核心案例和优化目标，这源于基于专家交换的边缘 MoE 系统在预填充阶段面临的特殊挑战——长输入序列几乎激活所有专家，产生大量 I/O 流量，严重影响 TTFT。值得指出的是，Sirius 的上述系统架构和优化设计原生兼容解码阶段，无需对核心机制进行实质性改动，即可对 Token 间时延产生显著的优化效果。

在注意力模块方面，预填充阶段完成后，各设备已分别持有不同子序列的键值缓存。在解码阶段，若由各设备将完整历史键值缓存同步至其余设备，通信开销将随序列长度线性增长，在长上下文场景下尤为显著。为此，Sirius 对解码执行路径作出轻量调整：各设备仍然在本地保留其键值缓存，转而将当前解码步骤产生的查询向量广播至所有设备[32]。由于查询向量仅携带单个 Token 的表示，广播开销可忽略不计。各设备在本地独立计算局部注意力输出，再经聚合通信汇总得到最终结果。该方案充分复用了预填充阶段构建的分布式键值缓存布局，将随序列增长的通信量转化为固定的单 Token 广播，通信效率在长上下文场景下优势尤为突出。

在 MoE 模块方面，解码阶段每步仍需将激活的 Top-K 个专家从二级存储换入主内存，单设备的串行 I/O 构成单步解码时延的重要来源。Sirius 的自适应 Token 调度策略可直接应用于此：将单个解码 Token 的各激活专家分配至不同设备，各设备并行完成各自负责的专家换入与计算，随后合并结果，将串行 I/O 转化为多设备并行，充分发挥边缘集群的集体 I/O 带宽。随着解码推进，各设备专家缓冲区的缓存命中率逐步提升，可进一步减少实际换入次数。推测性专家预取机制同样无缝适用：利用当前解码步的隐藏状态提前预测下一步所需专家并触发预取，将相邻步间的专家换入与当前步计算并行执行，消除解码关键路径上的 I/O 等待。这 2 项机制均无需针对解码场景专项修改，在统一的系统框架下同时服务于预填充与解码 2 个阶段。

4 实验与结果

4.1 实验设置

4.1.1 模型和数据集

本文选用 2 种主流混合专家架构 LLM Mixtral-8x7B 与 Qwen3-30B 对 Sirius 进行评估。为适应边缘环境下内存与 I/O 带宽资源受限的部署条件，模型采用 INT8 精度量化。在数据集方面，本文采用 GLUE 基准数据集，并据此构建 3 组不同长度的子集，各子集平均序列长度分别为 100、1 000 和 3 000 个 Token，用于系统地评估推理系统在不同输入序列长度下的性能表现。

4.1.2 边缘算力网络实验平台

本文采用 2 种现成的边缘计算设备 Jetson Nano[33] 与 Jetson TX2[34] 构建仿真边缘环境，以模拟智能家居、智能工厂等典型场景中资源受限的边缘节点。2 类设备的硬件参数配置如表 1 所示，一个由 4 台 Jetson Nano 组成，在存储层面主要依托其板载的 eMMC 闪存；另一个集群由 4 台 Jetson TX2 组成，配备了基于 NVMe 协议的高速固态硬盘，以此构建具备异构 I/O 能力的测试环境。同一边缘算力网络内的各设备通过一台支持网速动态调节的交换机互联，并通过调整设备间（device-to-device, D2D）通信带宽，以模拟真实边缘环境中多样化的网络条件。

Table 1 Hardware Resources and System Environment Configuration of Different Devices

表 1 不同设备的硬件资源和系统环境配置

硬件参数	Jetson Nano	Jetson TX2
CPU	Cortex-A57	Denver 2 + Cortex-A57
GPU	128-core NVIDIA Maxwell	256-core NVIDIA Pascal
单精度浮点算力/GFLOPS	472	1 330
显存/GB	8	8
功耗/W	10	20

4.1.3 基准方法

目前学界尚缺乏面向边缘算力网络的端到端分布式 MoE 推理系统，本文通过将云数据中心的并行

策略与现有端侧专家交换思想相结合，构建以下 3 种基准方法与 Sirius 进行对比。

1)无专家交换的并行推理。直接在边缘算力网络上部署基于专家并行的分布式推理框架，不引入专家交换机制。各设备仅加载并处理分配给自身的专家，内存压力完全由硬件容量决定。

2)专家中心式并行推理架构。每台设备分配一组固定且互不重叠的专家子集，并通过专家交换机制将暂不使用的专家卸载至二级存储以缓解内存压力。推理过程中，采用 All-to-All 通信将 Token 分发至负责其对应专家的设备进行计算，该方案与云数据中心中经典的专家并行策略类似。然而，该方案中专家的设备归属在离线阶段静态划分，未考虑运行时专家访问频率的冷热差异，易导致各设备间负载分配不均衡。

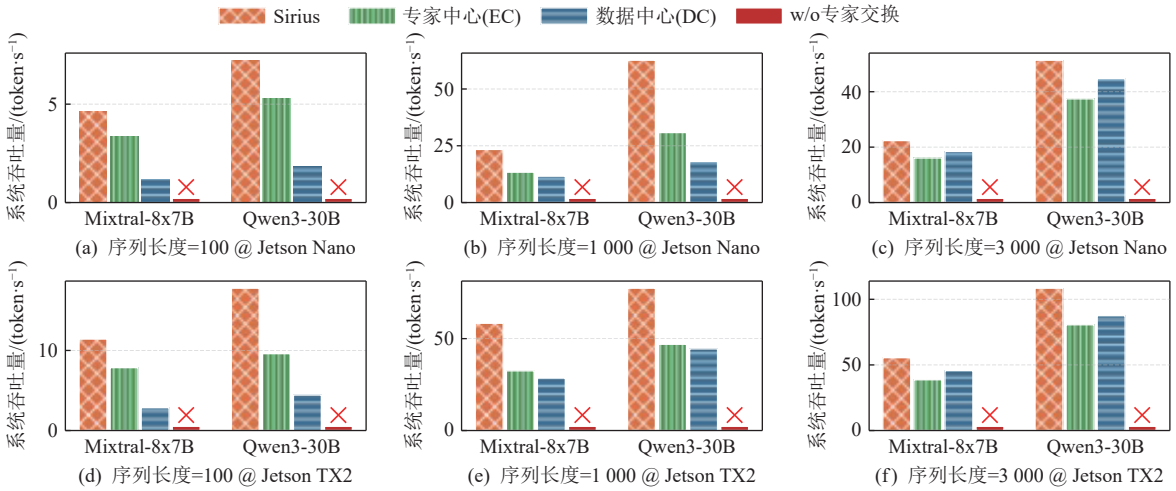
3)数据中心式并行推理架构。将输入序列切分为多个子序列并分配至不同设备，每台设备独立处理所分配的子序列，专家层计算无需设备间通信。然而，每台设备须将其子序列所激活的全部专家从二级存储换入内存后方可完成推理，对于较长子序列，可能会导致不同设备重复换入同一批专家，造成 I/O 带宽浪费。

对于注意力模块，上述 3 种基准方法均统一采用云数据中心中经典的序列并行方案进行并行计算，以确保对比的公平性。

4.1.4 端到端性能对比

本节评估系统的端到端推理吞吐量。实验分别在 Jetson Nano 集群与 Jetson TX2 集群上，针对 Mixtral-8x7B 和 Qwen3-30B 两种模型展开，输入序列长度覆盖 100、1 000 和 3 000 三种典型工作负载场景，D2D 网络带宽设置为 1 Gbps。本文采用推理吞吐量作为性能评估指标，定义为推理系统吞吐量 $T = N/t$，其中 N 为输入 Token 总数，t 为推理总消耗时间。

无专家交换的并行推理方案因内存溢出(out of memory, OOM)无法完成推理，表明直接将云数据中心并行策略应用于边缘环境在内存层面不具备可行性。对于专家中心式与数据中心式 2 种基准方法，Sirius 在全部模型与硬件配置下均取得最优系统性能，推理吞吐量提升幅度为 1.2~4.0 倍，实验结果如图 7 所示。



注: "×"代表由于内存不足无法运行。

Fig. 7 End-to-end performance comparison

图 7 端到端性能对比

进一步分析可知，系统性能瓶颈随输入序列长度呈现出规律性转变。在序列较短(长度为 100)时，推理瓶颈主要在于专家换入的 I/O 开销。数据中心式方案中，短子序列几乎激活模型的全部专家，导致各设备须独立将所有专家权重换入主内存，集体 I/O 带宽被大量冗余换入所消耗。Sirius 通过自适应 Token 调度，将 Token 优先分配至缓存命中率更高的设备，从而有效规避重复换入，相比数据中心式方案实现了最高 4.0 倍的推理吞吐量提升。在序列较长(长度为 3 000)时，推理瓶颈转移至专家计算阶段。此场景下，数据中心式方案因各设备独立处理子序列，计算负载分布相对均匀，性能表现尚可；而专家中心式方案由于专家归属在离线阶段静态划分，无法响应运行时专家访问频率的冷热差异，持有热门专家的设备负载显著偏高，成为制约整体吞吐量的拖尾设备。Sirius 的自适应 Token 调度策略在运行时动态感知各

设备负载与缓存状态，有效化解了负载不均衡问题，相比专家中心式方案实现了最高 1.43 倍的系统吞吐量提升。同时 Sirius 所采用的基于交错划分的序列并行通信优化以及推测性专家预取技术，也同样带来了显著的系统吞吐量提升。

4.1.5 注意力模块序列并行优化效果分析

本节单独评估基于交错划分的序列并行通信优化对注意力模块推理性能的提升效果。实验在 2 种模型与 2 个边缘集群上，针对序列长度 1 000 和 3 000 两种场景展开。通过调节交换机网速，将 D2D 网络带宽在 50 Mbps~1 Gbps 范围内逐步变化，以考察 Sirius 在不同网络条件下的系统鲁棒性。评估指标为单 MoE 层注意力模块的平均分布式推理时延，对比方案为云数据中心中张量并行[18]与序列并行[30] 2 种经典注意力并行策略。

实验结果如图 8 所示，Sirius 在全部实验配置下均取得最优性能。与张量并行相比，Sirius 实现了 1.42~7.03 倍的时延降低。这是因为张量并行在每次前向传播中须执行一次 All-Reduce 集合通信，其通信量随模型隐藏维度线性增长，在边缘环境受限的 D2D 带宽下开销尤为显著。与序列并行相比，Sirius 实现了 1.25~2.24 倍的时延降低，且性能优势随网络带宽的变化呈现出不同的主导因素。在低带宽场景（50 Mbps）下，通信为推理的主要瓶颈，Sirius 通过基于分块的通信-计算重叠机制，将键值矩阵传输与注意力计算并行执行，相比序列并行平均可取得约 1.2 倍的加速。在高带宽场景（1 Gbps）下，瓶颈由通信转移至计算，Sirius 的交错划分策略有效消除了传统连续划分因果掩码所引发的设备间负载不均衡，提升了各设备 GPU 资源的利用效率，平均加速比进一步提升至约 1.6 倍。

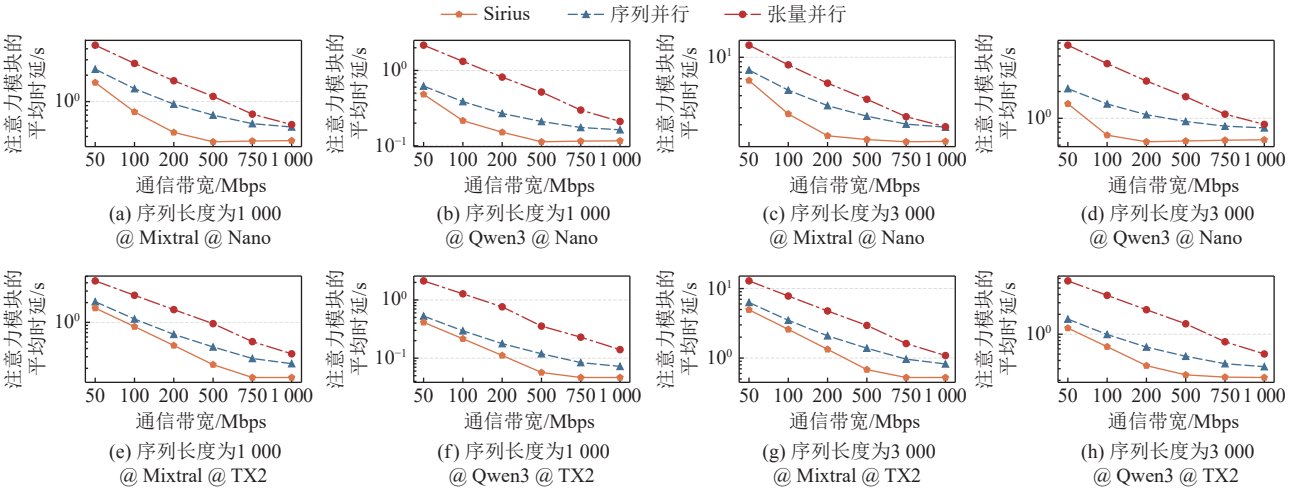
4.1.6 MoE 层 Token 调度策略效果分析

本节单独评估自适应 Token 调度策略对 MoE 层推理性能的提升效果。实验在 Jetson TX2 集群上，针对 2 种模型与序列长度 1 000 和 3 000 两种场景展开，对比方案为专家中心式与数据中心式调度。评估指标为单 MoE 层中专家换入时延与专家计算时延的分项开销，以分析不同调度策略对 I/O 效率与计算负载均衡的影响。

实验结果如图 9 所示，与数据中心式方案相比，Sirius 通过自适应 Token 调度，将激活相同热门专家的 Token 集中分配至同一设备，从根本上消除了多设备重复换入同一专家的冗余 I/O，充分发挥了边缘算力网络的集体 I/O 带宽优势，专家换入时延最高降低 4.1 倍。与专家中心式方案相比，Sirius 的贪心调度算法在确定 Token 到设备的映射时，联合考量各设备当前的换入开销与计算负载，动态响应运行时专家访问的冷热差异，有效规避了静态专家分配所导致的设备间负载不均衡，提升了 GPU 资源利用效率，在专家计算时延上最高取得 1.4 倍的降低。

4.1.7 推测性专家预取策略的有效性验证

本节单独验证推测性专家预取策略对系统性能的提升效果。如第 3.4 节所述，相邻两层隐藏状态之间具有极高的相似性，这为基于当前层隐藏状态预测下一层专家激活结果提供了理论依据。预实验结果如图 5 所示，上述预测对于 Mixtral-8x7B 与 Qwen3-30B 两种模型的预取覆盖率均超过 80%，验证了推测性预取在典型 MoE 架构上的可行性。



注：Mixtral 是 Mixtral-8x7B 简写，Qwen3 是 Qwen3-30B 简写，Nano 是 Jetson Nano 简写，TX2 是 Jetson TX2 简写。

Fig. 8 Effectiveness analysis of sequence parallelism optimization for attention modules

图 8 注意力模块序列并行优化效果分析

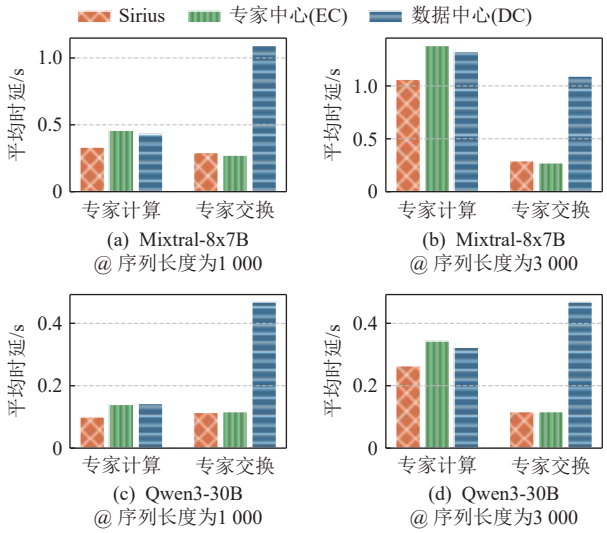


Fig. 9 Effectiveness analysis of Token scheduling strategy for MoE layers

图 9 MoE 层 Token 调度策略效果分析

本节针对 2 种 MoE 模型, 在序列长度为 1 000 和 3 000 两种场景下, 对比以下 3 种方案:

1)按需换入。仅在 Token 需要执行时方才换入其激活的专家, 不引入任何缓存或预取机制。

2)LRU 缓冲区缓存。在按需换入的基础上引入总容量为 2 GB 的专家缓冲区, 容量在各 MoE 层间均匀分配, 并采用最近最少使用(least recently used, LRU)策略进行替换, 当所需专家命中缓冲区时可直接复用, 无需重复换入。

3)Sirius 推测性预取。在 LRU 缓冲区的基础上, 进一步引入推测性专家预取机制, 利用第 l 层的隐藏状态提前预测第 $l+1$ 层的专家激活, 并与第 l 层的计算并行完成调度与换入, 将计算、调度与 I/O 三者流水线化执行。

本文以平均单 MoE 层执行时延作为评估指标, 实验结果如表 2 所示。相比按需换入, 引入 LRU 缓冲区策略可降低约 10% 的执行时延; 在此基础上进一步启用推测性预取后, 时延得到显著压缩, Sirius 相比按需换入基线实现了 1.13~1.86 倍的时延降低。

Table 2 Effectiveness of Experts Prefetching

表 2 专家预取有效性

平均时延/s	Mixtral-8x7B		Qwen3-30B	
	序列长度为 1 000	序列长度为 3 000	序列长度为 1 000	序列长度为 3 000
无预取	0.63	1.34	0.27	0.38
LRU 算法	0.51	1.18	0.23	0.34
本文方案	0.35	1.06	0.15	0.26

5 结 论

本文针对资源受限的本地边缘环境下高效部署 MoE 模型这一核心问题, 提出了 Sirius——一种面向边缘算力网络的资源高效本地 MoE 推理系统。Sirius 以序列并行推理架构为基础, 协调多台边缘设备的集体计算能力、内存容量与 I/O 带宽, 显著降低预填充阶段的 TTFT。针对多边缘协作推理中的 3 项关键挑战, 本文分别设计了对应的解决机制: 基于交错划分的序列并行策略, 通过将输入序列重新组织为交错子序列, 解决因果注意力机制下设备间工作负载不均衡问题, 最大化通信-计算重叠效率; 自适应 Token 调度策略, 通过贪心算法联合考量计算开销、专家换入时延与缓冲区缓存状态, 充分发挥多设备集体 I/O 带宽优势, 同时消除拖尾设备效应; 推测性专家预取机制, 利用 Transformer 相邻层隐藏状态的高度相似性, 在当前层计算期间并行完成下一层的 Token 调度与专家预取, 实现计算-调度-I/O 的流水线并行执行。在真实边缘集群平台上部署多种 MoE 模型的实验结果表明, Sirius 在预填充时延方面取得了显著提升, 验证了上述架构设计的有效性。

作者贡献声明: 叶盛源负责文献调研、方案整体设计和论文撰写; 梁涵负责部分算法思路和实验方法; 陈旭提出指导意见并修改论文。

参 考 文 献

[1] Liu Aixin, Feng Bei, Xue Bing, et al. DeepSeek-V3 technical report[R/OL]. arXiv: 2412.19437[2026-02-27]. https://arxiv.org/abs/2412.19437

[2] Yang An, Li Anfeng, Yang Baosong, et al. Qwen3 technical report[R/OL]. arXiv: 2505.09388[2026-02-27]. https://arxiv.org/abs/2505.09388

[3] Team Kimi, Bai Yifan, Bao Yiping, et al. Kimi K2: Open agentic intelligence[R/OL]. arXiv: 2507.20534[2026-02-27]. https://arxiv.org/abs/2507.20534

[4] King Evan, Yu Haoxiang, Lee Sangsu, et al. Sasha: Creative goal-oriented reasoning in smart homes with large language models[J]. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 2024, 8(1): 1−38

[5] Ye Shengyuan, Ouyang Bei, Du Jiangsu, et al. Resource-efficient collaborative edge Transformer inference with hybrid model parallelism[J]. IEEE Transactions on Mobile Computing, 2025: 10945−10962

[6] Li Yuanchun, Wen Hao, Wang Weijun, et al. Personal LLM agents:

Insights and survey about the capability, efficiency and security[EB/OL]. arXiv: 2401.05459[2026-02-27]. https://arxiv.org/abs/2401.05459

[7] Kong Rui, Li Yuanchun, Wang Weijun, et al. Serving moe models on resource-constrained edge devices via dynamic expert swapping[J]. IEEE Transactions on Computers, 2025, 2799−2811

[8] Yi Rongjie, Guo Liwei, Wei Shiyun, et al. EdgeMoE: Empowering sparse large language models on mobile devices[J]. IEEE Transactions on Mobile Computing, 2025, 7059−7073

[9] Hwang Ranggi, Wei Jianyu, Cao Shijie, et al. Pre-gated MOE: An algorithm-system co-design for fast and scalable mixture-of-expert inference[C]//Proc of 2024 ACM/IEEE 51st Annual Int Symp on Computer Architecture (ISCA). Piscataway, NJ: IEEE, 2024: 1018−1031

[10] Wang Haodong, Zhou Qihua, Hong Zicong, et al. D2MoE: Dual routing and dynamic scheduling for efficient on-device MOE-based LLM serving[C]//Proc of the 31st Annual Int Conf on Mobile Computing and Networking. New York: ACM, 2025: 574−588

[11] Jiang A Q, Sablayrolles A, Roux A, et al. Mixtral of experts[EB/OL]. arXiv: 2401.04088[2026-02-27]. https://arxiv.org/abs/2401.04088

[12] Eliseev A, Mazur D. Fast inference of mixture-of-experts language models with offloading[EB/OL]. arXiv: 2312.17238[2026-02-27]. https://arxiv.org/abs/2312.17238

[13] Qin Ruoyu, Li Zheming, He Weiran, et al. Mooncake: Trading more storage for less computation-a kvcache-centric architecture for serving LLM Chatbot[C]//Proc of the 23rd USENIX Conf on File and Storage Technologies (FAST 25). Berkeley, CA: USENIX Association, 2025: 155−170

[14] Lu Xudong, Liu Qi, Xu Yuhui, et al. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models[C]//Proc of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Stroudsburg, PA: ACL, 2024: 6159−6172

[15] Song Yixin, Mi Zeyu, Xie Haotong, et al. PowerInfer: Fast large language model serving with a consumer-grade GPU[C]//Proc of the ACM SIGOPS 30th Symp on Operating Systems Principles. New York: ACM, 2024: 590−606

[16] Fang Zhiyuan, Hong Zicong, Huang Yuegui, et al. Fate: Fast edge inference of mixture-of-experts models via cross-layer gate[EB/OL]. arXiv: 2502.12224[2026-02-27]. https://arxiv.org/abs/2502.12224

[17] Huang Yanping, Cheng Youlong, Bapna Ankur, et al. GPipe: Efficient training of giant neural networks using pipeline parallelism[J]. Advances in Neural Information Processing Systems, 2019, 32: 103−112

[18] Narayanan D, Shoeybi M, Casper J, et al. Efficient large-scale language model training on GPU clusters using megatron-LM[C]//Proc of the Int Conf for High Performance Computing, Networking, Storage and Analysis. Piscataway, NJ: IEEE, 2021: 1−15

[19] Lepikhin D, Lee H, Xu Yuanzhong, et al. GShard: Scaling giant models with conditional computation and automatic sharding[EB/OL]. arXiv: 2006.16668[2026-02-27]. https://arxiv.org/abs/2006.16668

[20] Hwang Changho, Cui Wei, Xiong Yifan, et al. Tutel: Adaptive mixture-of-experts at scale[J]. Proceedings of Machine Learning and Systems, 2023, 5: 269−287

[21] Liu Juncai, Wang Jessie Hui, Jiang Yimin. Janus: A unified distributed training framework for sparse mixture-of-experts models[C]//Proc of the ACM SIGCOMM 2023 Conf. New York: ACM, 2023: 486−498

[22] Zhai Mingshu, He Jiaao, Ma Zixuan, et al. SmartMoE: Efficiently training sparsely-activated models through combining offline and online parallelization[C]//Proc of 2023 USENIX Annual Technical Conf (USENIX ATC 23). Berkeley, CA: USENIX Association, 2023: 961−975

[23] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. Advances in Neural Information Processing Systems, 2017, 30: 5998−6008

[24] Ye Shengyuan, Du Jiangsu, Zeng Liekang, et al. Galaxy: A resource-efficient collaborative edge AI system for in-situ transformer inference[C]//Proc of IEEE INFOCOM 2024-IEEE Conf on Computer Communications. Piscataway, NJ: IEEE, 2024: 1001−1010

[25] Ye Shengyuan, Ouyang Bei, Zeng Liekang, et al. Jupiter: Fast and resource-efficient collaborative inference of generative LLMs on edge devices[C]//Proc of IEEE Conf on Computer Communications (IEEE INFOCOM 2025). Piscataway, NJ: IEEE, 2025: 1−10

[26] Wei Yuanxin, Ye Shengyuan, Jiang Jiazhi, et al. Communication-efficient model parallelism for distributed in-situ Transformer inference[C]//Proc of 2024 Design, Automation & Test in Europe Conference & Exhibition (DATE). Piscataway, NJ: IEEE, 2024: 1−6

[27] Ye Shengyuan, Zeng Liekang, Chu Xiaowen, et al. Asteroid: Resource-efficient hybrid pipeline parallelism for collaborative DNN training on heterogeneous edge devices[C]//Proc of the 30th Annual Int Conf on Mobile Computing and Networking. New York: ACM, 2024: 312−326

[28] Zhu Zilin. Ring flash attention[CP/OL]. GitHub repository[2026-02-28]. https://github.com/zhuzilin/ring-flash-attention

[29] Brandon W, Nrusimha A, Qian K, et al. Striped attention: Faster ring attention for causal transformers[EB/OL]. arXiv: 2311.09431[2026-02-28]. https://arxiv.org/abs/2311.09431

[30] Li Shenggui, Xue Fuzhao, Baranwal C, et al. Sequence parallelism: Long sequence training from system perspective[C]//Proc of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Stroudsburg, PA: ACL, 2023: 2391−2404

[31] Liu Zichang, Wang Jue, Dao T, et al. Deja vu: Contextual sparsity for efficient LLMs at inference time[C]//Proc of Int Conf on Machine Learning. New York: PMLR, 2023: 22137−22176

[32] Jacobs S, Tanaka M, Zhang Chengming, et al. Deepspeed ulysses: System optimizations for enabling training of extreme long sequence transformer models[EB/OL]. arXiv: 2309.14509[2026-02-28]. https://arxiv.org/abs/2309.14509

[33] NVIDIA. Jetson-Nano[EB/OL]. [2026-02-28]. https://developer.nvidia.com/embedded/jetson-nanodeveloper-kit

[34] NVIDIA. Jetson-TX2[EB/OL]. [2026-02-28]. https://developer.nvidia.com/embedded/jetson-tx2



Ye Shengyuan, born in 1999. PhD candidate. His main research interests include mobile edge computing, distributed machine learning systems, and large language models.

叶盛源, 1999 年生。博士研究生。主要研究方向为移动边缘计算、分布式机器学习系统、大语言模型。



Liang Han, born in 2001. PhD candidate. His main research interests include edge computing, distributed training systems, and multimodal large language model applications.

梁　涵, 2001 年生。博士研究生。主要研究方向为移动边缘计算、分布式训练系统、多模态大语言模型应用。



Chen Xu, born in 1986. Professor. His main research interests include edge computing and large language model systems.

陈　旭, 1986 年生。教授。主要研究方向为边缘计算、大语言模型系统。