HybriMoE: Hybrid CPU-GPU Scheduling and Cache Management for Efficient MoE Inference

DAC 2025

汇报人：姚昌硕

2025年9月11日

背景知识：MoE (Mixture of Experts)

- 背景知识: MoE
  - Decoder-only结构
  - MoE
- 相关工作
- 研究动机和发现
- 系统设计
- 实验
- Idea

- 主流的LLM结构: Decoder-only
  - Masked Self-Attention
  - FFN (Feed Forward Network)

- 这种结构存在的问题:
  - FFN占整个模型计算量比例较大

- MoE的目的:
  - 让FFN变得”稀疏”，减少被激活参数的数量，从而减少计算量

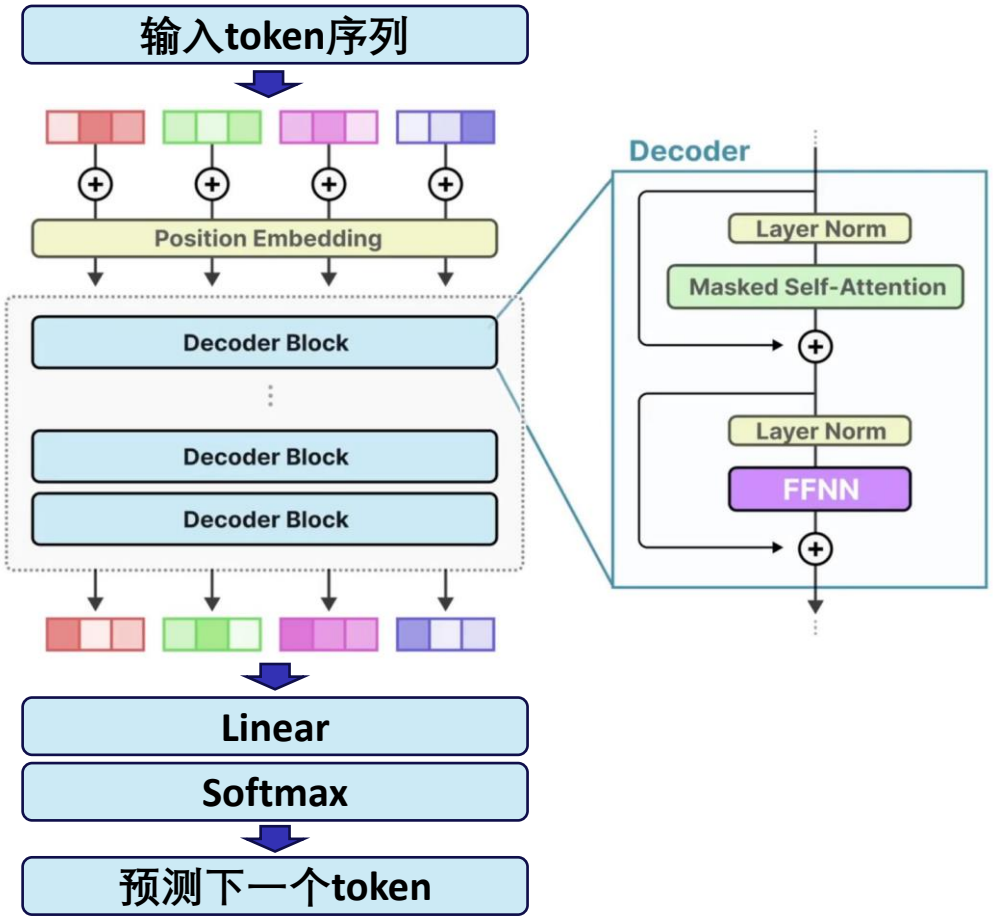


图1: Decoder-only模型结构

背景知识：MoE (Mixture of Experts)

- 背景知识: MoE
  - Decoder-only结构
  - MoE
- 相关工作
- 研究动机和发现
- 系统设计
- 实验
- Idea

- 专家 (Experts)
  - 多个小型FFN
  - 仅被选中的专家激活计算
- 路由网络 (Router)
  - 一个小型FFN
  - 为每个输入Token动态选择专家
- MoE层输出:
  - 被激活专家输出的加权求和

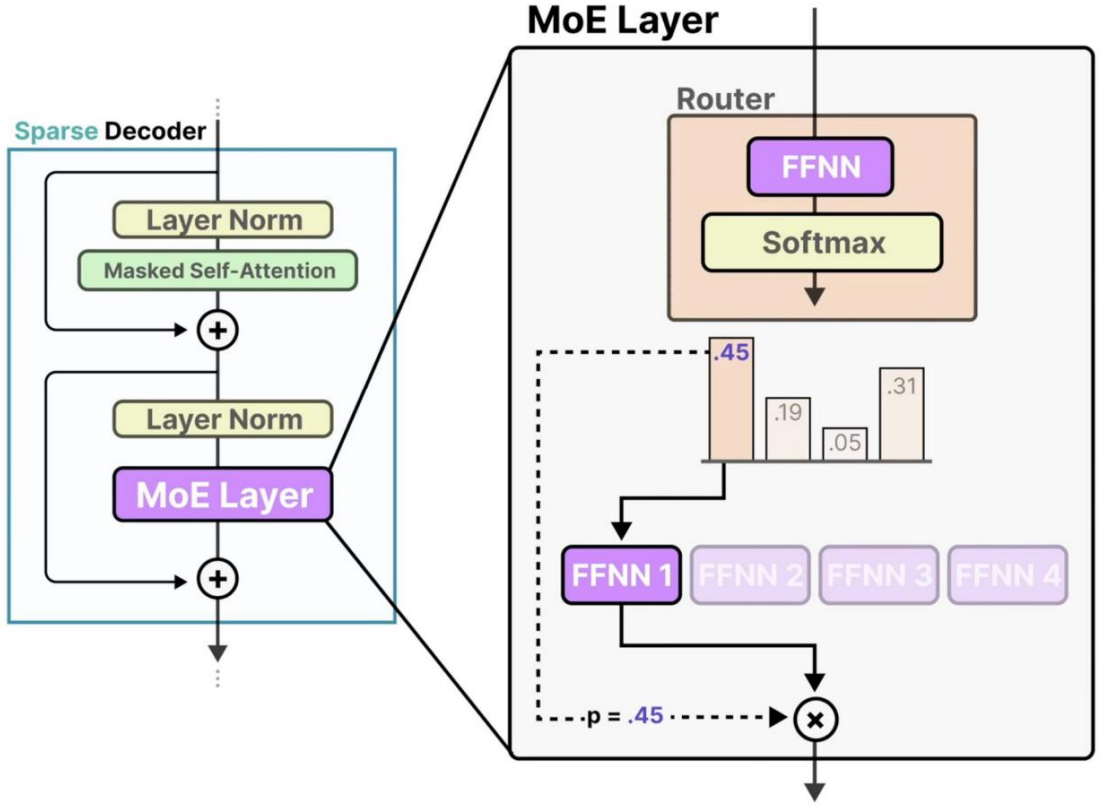


图2: MoE Layer结构

相关工作：CPU-GPU混合推理

- 背景知识
- 相关工作
  - GPU按需加载
  - CPU-GPU混合计算
  - 存在的问题
- 研究动机和发现
- 系统设计
- 实验
- Idea

- 研究背景：显存(VRAM)稀缺，无法将所有参数同时驻留在GPU上
- 方法1：卸载到系统内存(RAM)，按需加载到VRAM
  - 缺点①：GPU需要等待PCIe传输
  - 缺点②：CPU闲置

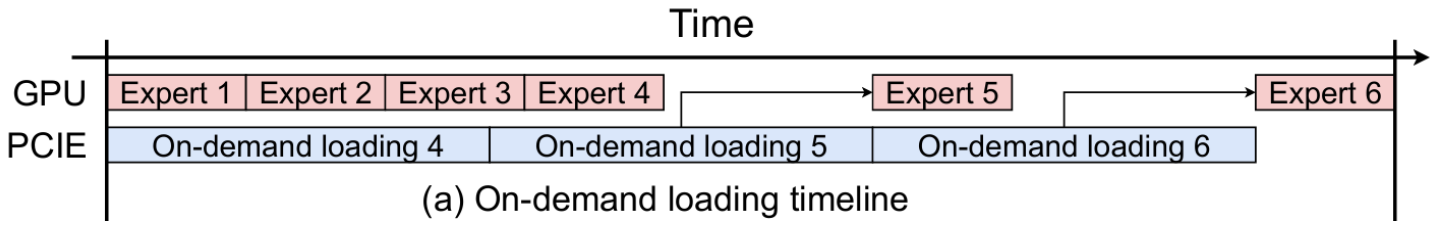


图3: Execution timeline (按需加载)

相关工作：CPU-GPU混合推理

- 背景知识
- 相关工作
  - GPU按需加载
  - CPU-GPU混合计算
  - 存在的问题
- 研究动机和发现
- 系统设计
- 实验
- Idea

- 方法2：引入CPU计算
  - 思路：将常用权重存放在VRAM，不常用权重存放在RAM
  - 相关工作：PowerInfer / Fiddler / kTransformers
  - 缓存卸载方法：基于以往统计信息
    - LFU (Least Frequently Used) / LRU (Least Recently Used)

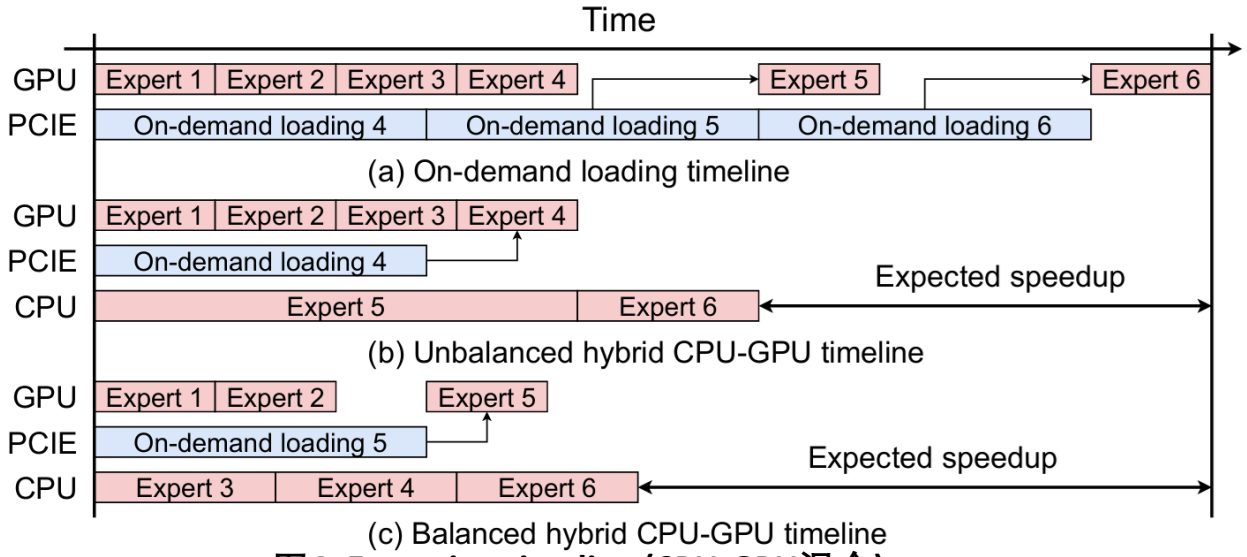


图4: Execution timeline (CPU-GPU混合)

相关工作：CPU-GPU混合推理

- 背景知识
- 相关工作
  - GPU按需加载
  - CPU-GPU混合计算
  - 存在的问题
- 研究动机和发现
- 系统设计
- 实验
- Idea

- 存在的问题：
  - MoE中LFU / LRU并不能有效预测每个专家的激活
  - CPU-GPU负载不均衡，导致推理效率低

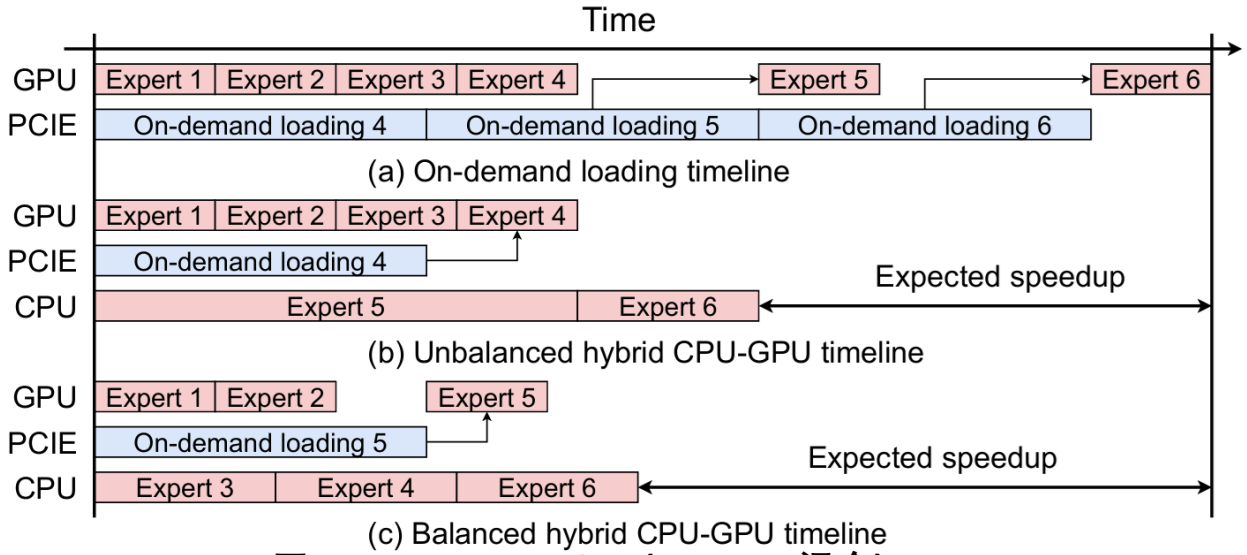


图4: Execution timeline (CPU-GPU混合)

研究动机：MoE模型的特点

- 背景知识
- 相关工作
- 研究动机和发现
  - 专家激活规律
  - CPU/GPU计算特性
- 系统设计
- 实验
- Idea

- 特性1：更高的不稳定性（激活更加随机）

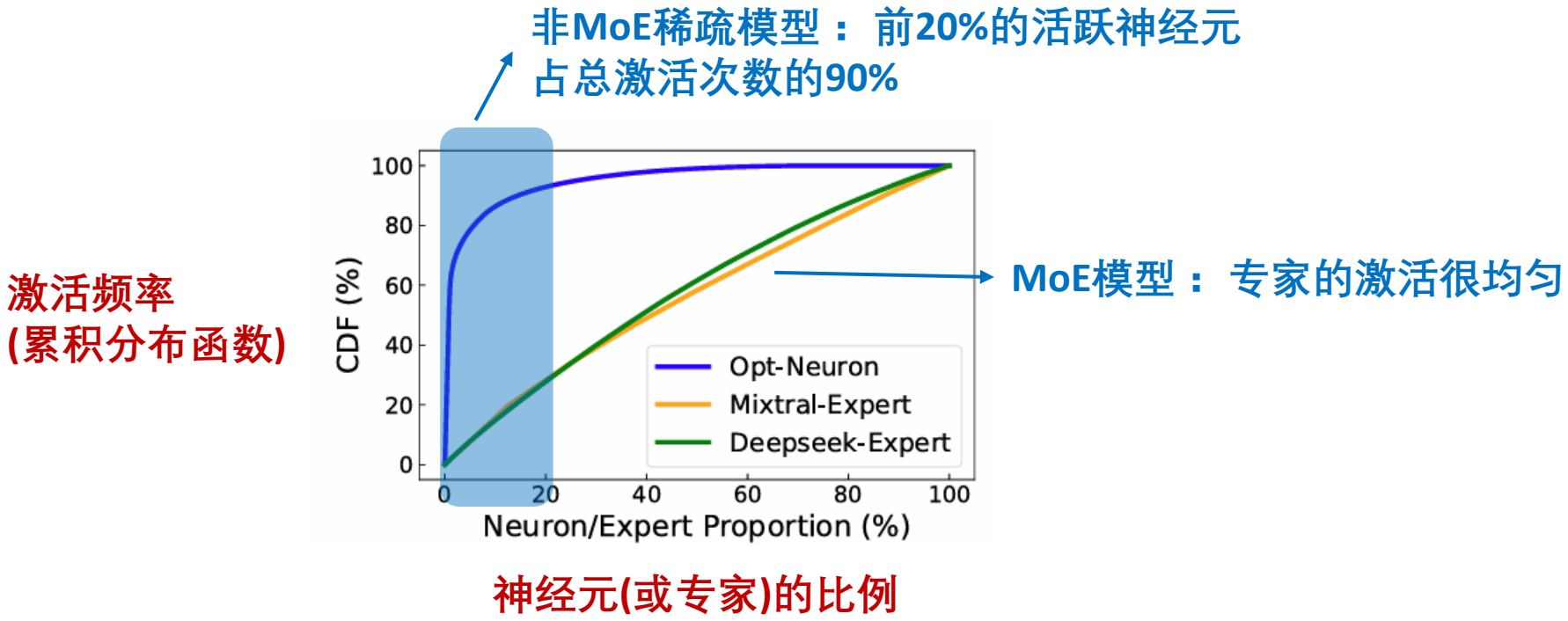


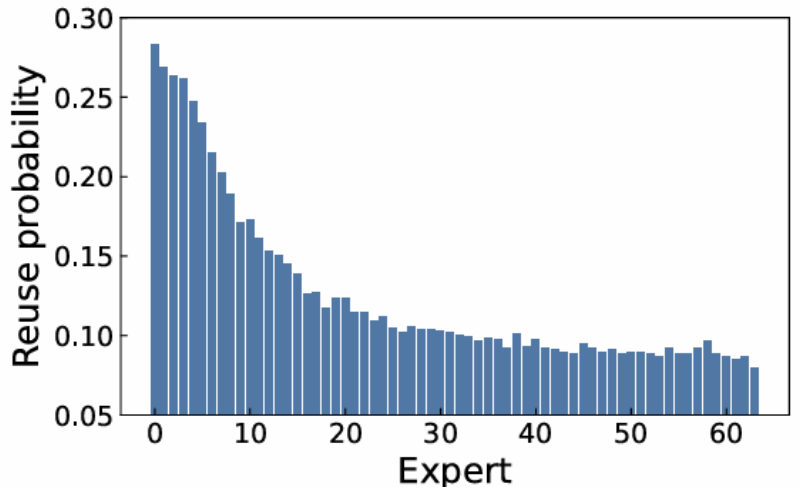
图5: 不同模型中的神经元(或专家)累计激活频率

研究动机：MoE模型的特点

- 背景知识
- 相关工作
- 研究动机和发现
  - 专家激活规律
  - CPU/GPU计算特性
- 系统设计
- 实验
- Idea

- 发现：
  - ① 具有更高激活分数的专家更有可能在下一次迭代中被重用
  - ② 模型在相邻层之间表现出激活值的高度相似性

下次迭代中的重用概率



专家（→分数降序排列）

图6: 不同专家在下次迭代中被重用的概率

研究动机：MoE模型的特点

- 背景知识
- 相关工作
- 研究动机和发现
  - 专家激活规律
  - CPU/GPU计算特性
- 系统设计
- 实验
- Idea

- 发现：
  - ① 具有更高**激活分数**的专家更有可能在下一次**迭代**中被重用

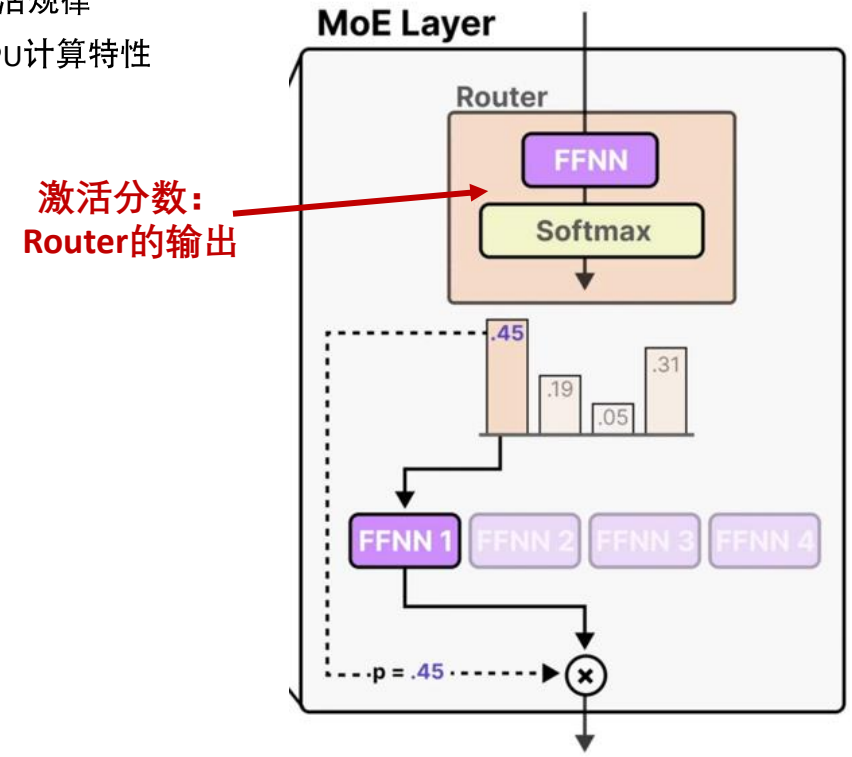


图7: 激活分数

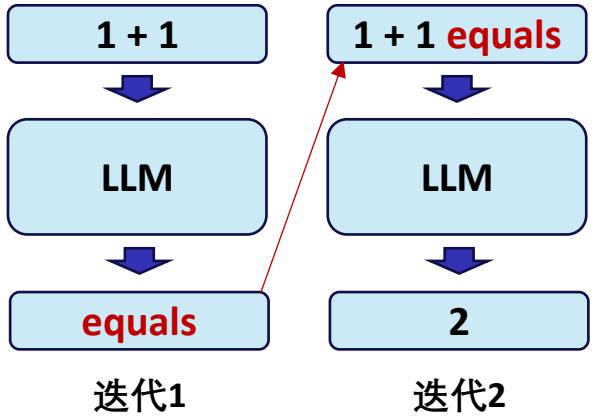


图8: 自回归生成

研究动机：MoE模型的特点

- 背景知识
- 相关工作
- 研究动机和发现
  - 专家激活规律
  - CPU/GPU计算特性
- 系统设计
- 实验
- Idea

- 发现：
  - ① 具有更高**激活分数**的专家更有可能在下一次**迭代**中被重用

- 个人理解：一段文本语义/上下文的连续性
  - 语义很少会因为尾部新增一个token而发生突变，
  - 而是随着token逐步生成而发生连续、平稳的变化。

研究动机：MoE模型的特点

- 背景知识
- 相关工作
- 研究动机和发现
  - 专家激活规律
  - CPU/GPU计算特性
- 系统设计
- 实验
- Idea

- 发现：
  - ② 模型在相邻层之间表现出激活值的高度相似性

- 原因分析：残差连接的存在
  - 残差连接通常会使得层间的激活值分布更加相似

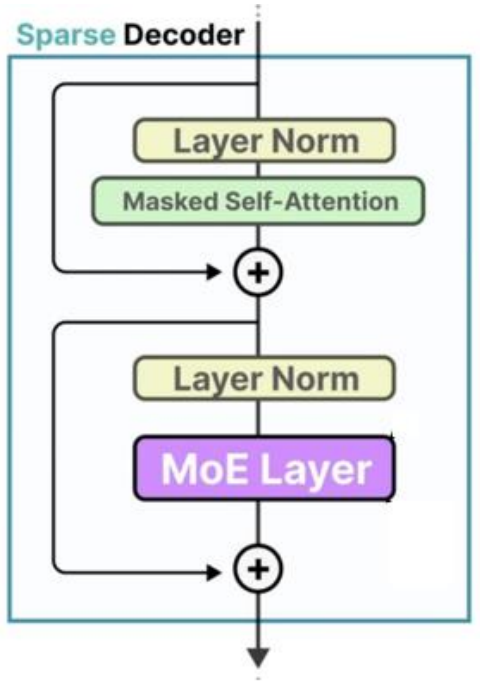


图9: Decoder Block结构

研究动机：CPU/GPU计算特性

- 背景知识
- 相关工作
- 研究动机和发现
  - 专家激活规律
  - CPU/GPU计算特性
- 系统设计
- 实验
- Idea

- 特性2：CPU/GPU不同的计算特性
- CPU：
  - 第一个专家计算较慢，后续效率变高。
  - 随着负载增加，计算时间线性增长。
- GPU：
  - 随着专家数量的增加，计算时间线性增长。
  - 随着负载增加，计算时间相对稳定。

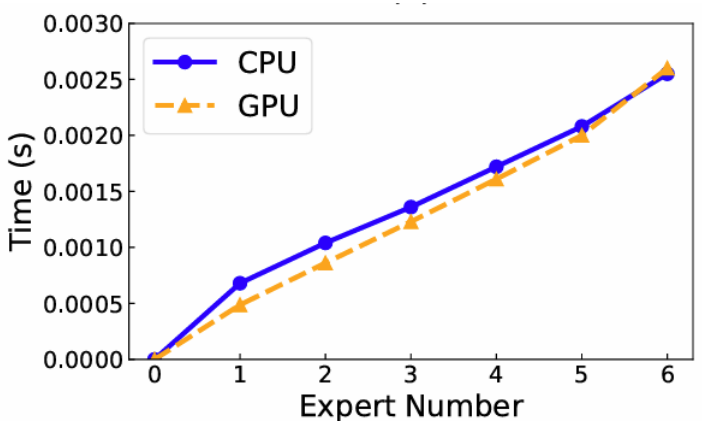


图10: CPU vs. GPU 运行时间随专家数量的变化 (固定负载)

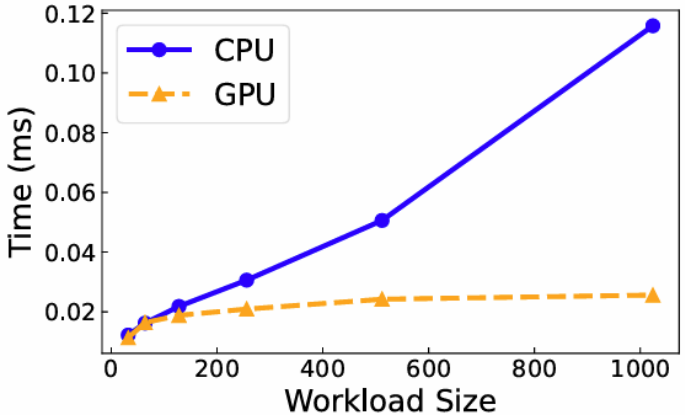


图11: CPU vs. GPU 运行时间随工作负载的变化

系统设计

- 背景知识
- 相关工作
- 研究动机和发现
- 系统设计
  - 混合调度策略
  - 预取机制
  - VRAM缓存管理
- 实验
- Idea

- ① 混合调度策略
- ② 预取机制
- ③ VRAM缓存管理

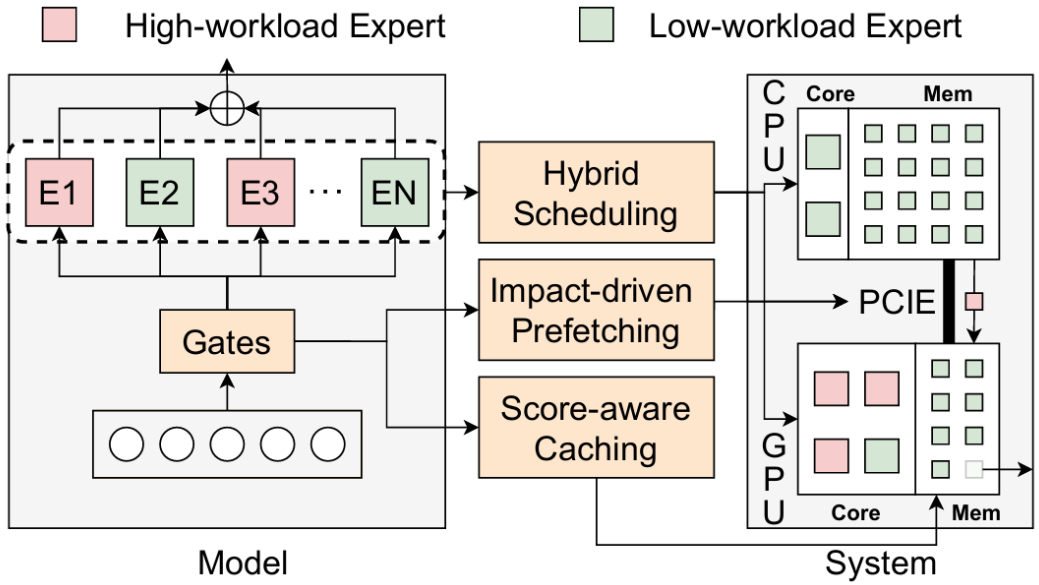


图12: HybriMoE系统设计

系统设计：① 混合调度策略

- 背景知识
- 相关工作
- 研究动机和发现
- 系统设计
  - 混合调度策略
  - 预取机制
  - VRAM缓存管理
- 实验
- Idea

- ① 混合调度策略：基于优先级规则的动态调度
  - 创建两个队列：
    - **GPU**队列：存放当前已缓存至VRAM的专家，负载降序 (高负载优先)
    - **CPU**队列：存放未缓存的专家，负载升序 (低负载优先)
  - 优先级规则：
    - **GPU**计算：从**GPU**队列头部取任务 (最**高**负载)。
    - **CPU**计算：从**CPU**队列头部取任务 (最**低**负载)。
    - **PCIe**传输：从**CPU**队列尾部取任务 (最**高**负载) → 传输到**GPU**缓存

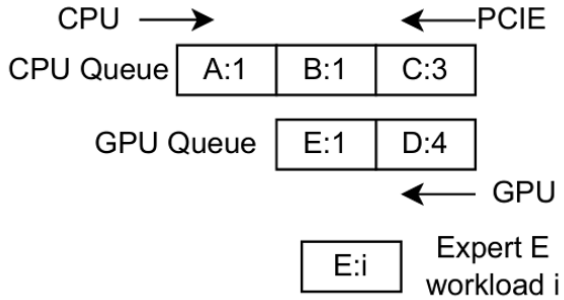


图13: 调度算法中的CPU/GPU队列

系统设计：② 预取机制

- 背景知识
- 相关工作
- 研究动机和发现
- 系统设计
  - 混合调度策略
  - 预取机制
  - VRAM缓存管理
- 实验
- Idea

- ② 预取机制
  - 原理：利用相邻层激活值的相似性
  - 预测未来的激活：
    - 取层i的Router的输入，作为层i+1的Router的近似估计
    - 将其输入层i+1的Router，得到层i+1的专家激活情况（即可以被预取的专家）
  - 使用模拟计算收益：
    - 将候选专家加入GPU队列，用混合调度算法模拟计算时间

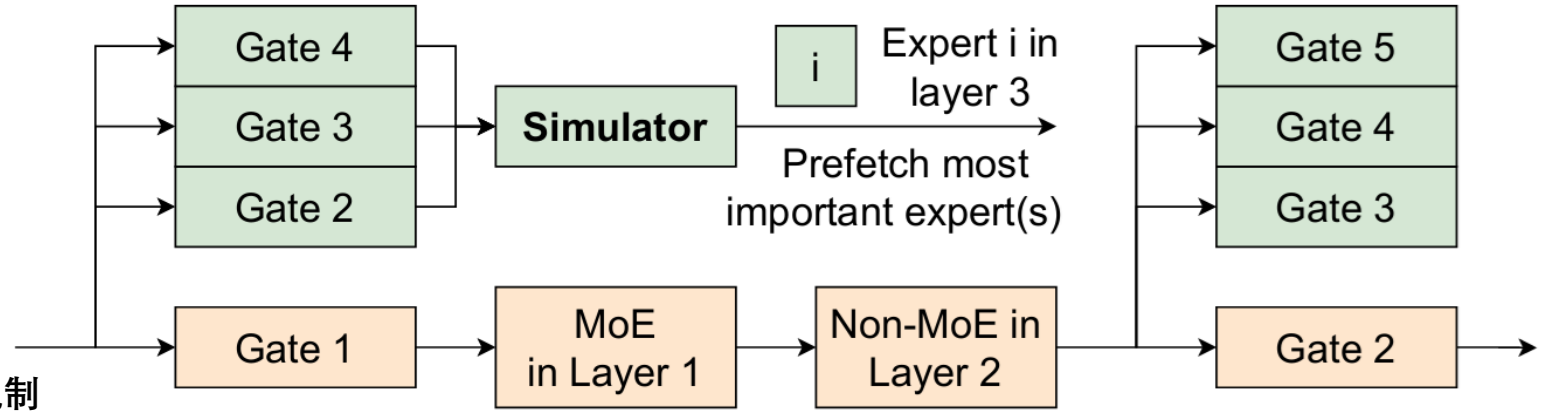


图14: 预取机制

系统设计：③ VRAM缓存管理

- 背景知识
- 相关工作
- 研究动机和发现
- 系统设计
  - 混合调度策略
  - 预取机制
  - VRAM缓存管理
- 实验
- Idea

- ③ VRAM缓存管理
  - 原理：更高激活分数的专家更有可能在下一次迭代中被重用
  - 提出：MRS (Minus Recent Score) 的缓存替换策略
    - 卸载分数最低的专家
  - 分数的更新

$$S = \alpha \times TopP(s) + (1 - \alpha) \times S$$

平均系数　原始分数　累计分数

保留原始分数最高的K个专家，其余专家s=0

系统设计：小结

- 背景知识
- 相关工作
- 研究动机和发现
- 系统设计
  - 混合调度策略
  - 预取机制
  - VRAM缓存管理
- 实验
- Idea

特性	系统设计
CPU在处理连续小规模的专家时，效率会比较高。GPU更适合处理高负载的专家。	**混合调度策略：** 尽量将小负载的专家放在CPU上， 大负载的专家放在GPU上执行。
模型在相邻层之间表现出激活值的高度相似性	**预取机制：** 使用本层激活值作为后续几层的近似， 估计各专家的预取收益
具有更高激活分数的专家更有可能在下一次迭代中被重用	**VRAM缓存管理：** 基于专家历史累积的激活分数来估计未来被重用的概率，卸载分数低的专家

实验：实验设置

- 背景知识
- 相关工作
- 研究动机和发现
- 系统设计
- 实验
  - 实验设置
  - 端到端性能
  - 缓存策略有效性分析
- Idea

- 硬件：
  - 1 * NVIDIA RTX A6000 (48GB VRAM)
  - 1 * Intel Xeon Gold 5220R (限制使用10核心来模拟算力限制)
- LLM模型：
  - 少量专家： Mixtral-8x7B (8专家, 2激活)
  - 大量专家： Qwen2-57B-A14B (64专家, 8激活, 1共享) / DeepSeek-V2-Lite (64专家, 6激活, 2共享)
- 对比方法：
  - llama.cpp: 通用的CPU-GPU混合推理框架。采用的是静态的层映射策略。
  - AdapMoE: 当前GPU进行MoE推理的SOTA，具备自适应的预取和缓存，但不利用CPU计算。
  - kTransformers: 当前CPU-GPU混合MoE推理的SOTA，根据专家使用频率决定缓存位置。
- 评估指标：
  - Prefill阶段： TTFT (Time To First Token)
  - Decode阶段： TBT (Time Between Tokens)

实验：端到端性能

- 背景知识
- 相关工作
- 研究动机和发现
- 系统设计
- 实验
  - 实验设置
  - 端到端性能
  - 缓存策略有效性分析
- Idea

- 在不同输入长度（32, 128, 512, 1024 tokens）和不同GPU缓存容量（25%, 50%, 75%）下，对Prefill和Decode阶段进行测试。

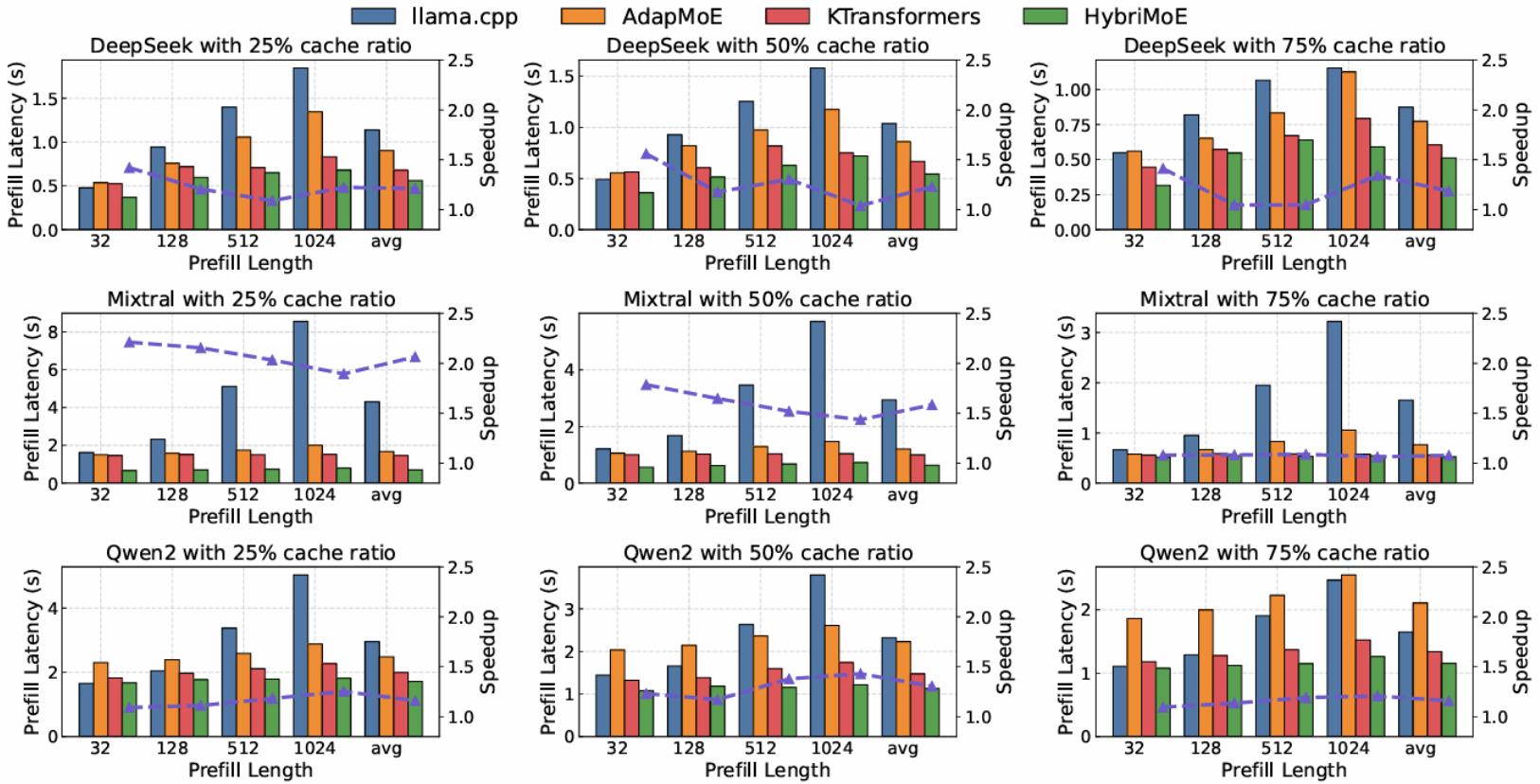


Fig. 7. Prefill stage performance comparison across different input lengths and cache ratios, highlighting relative speedups against kTransformers.

实验：消融实验

- 背景知识
- 相关工作
- 研究动机和发现
- 系统设计
- 实验
  - 实验设置
  - 端到端性能
  - 缓存策略有效性分析
- Idea

- 消融实验配置： Qwen2模型、25%缓存容量
- 结论：三个方法都对加速有贡献，并且混合调度的贡献最大

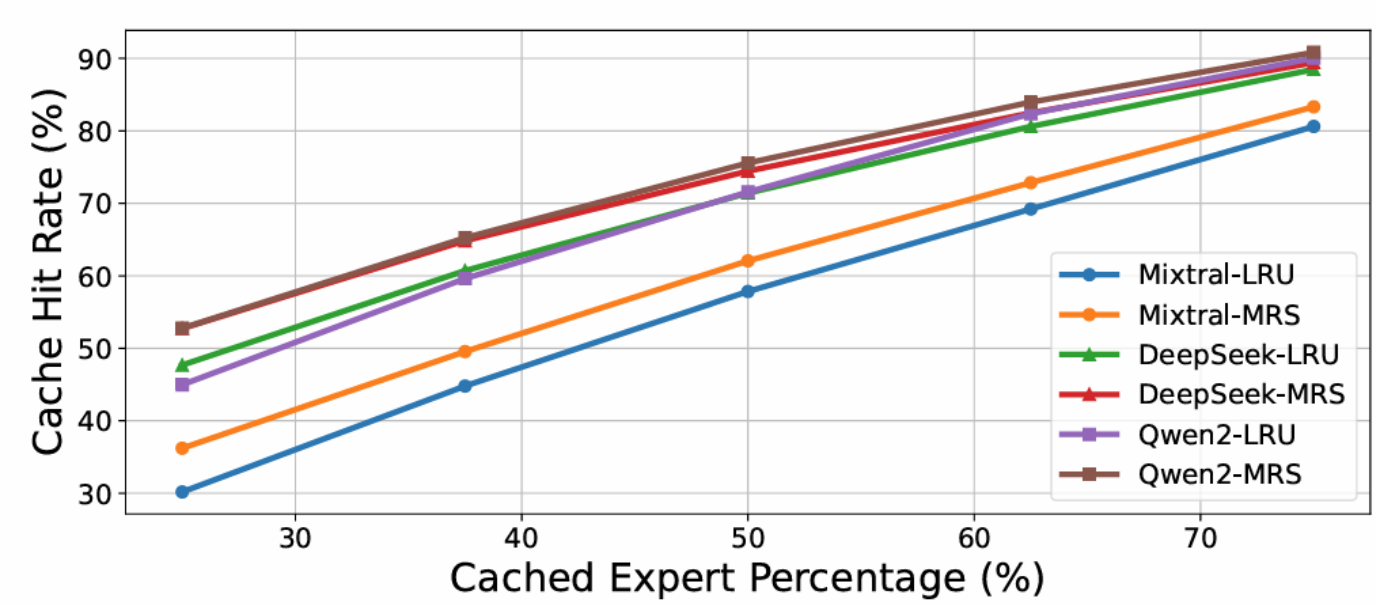
TABLE III
MoE INFERENCE SPEEDUP BREAKDOWN OF PROPOSED TECHNIQUES.

	Technique	Latency(s)	Speedup
Prefill	Baseline	1.47	
	Baseline+Scheduling	1.17	1.26×
	Baseline+Prefetching	1.39	1.06×
	All	1.13	1.31×
Decode	Baseline	0.21	
	Baseline+Scheduling	0.14	1.46×
	Baseline+Prefetching	0.18	1.15×
	Baseline+Caching	0.15	1.38×
	All	0.11	1.86×

实验：缓存策略有效性分析

- 背景知识
- 相关工作
- 研究动机和发现
- 系统设计
- 实验
  - 实验设置
  - 端到端性能
  - 缓存策略有效性分析
- Idea

- 对比本文提出的MRS缓存策略和传统的LRU策略的缓存命中率
- 结论：
  - 在不同缓存专家百分比下，MRS优于LRU
  - 缓存专家百分比越低 (即使用VRAM越少)，MRS优势越大



Idea

- 背景知识
- 相关工作
- 研究动机和发现
- 系统设计
- 实验
- Idea

- 基于此：在此之上，顺势思维

- 针对本文没有讨论的问题进行细化：
  - 为什么预取考虑3层
  - MRS计算得分是否应当考虑计算代价
  - 对PCIE传输时间为恒定值的假设是否可以细化
  - 能否优化当前固定的调度优先级（比如使用强化学习）
  - 能否扩展到多机情形

Idea

- 背景知识
- 相关工作
- 研究动机和发现
- 系统设计
- 实验
- Idea

- 与此类似：与此平行，发散思维
- 借鉴本文发现规律的思路：
  - MoE专家的激活模式在全局统计上是高度不稳定的，无规律可循。而作者从两个维度进行分解：
  - 时间维度：在相邻的推理迭代之间的相关性
  - 结构维度：相邻的网络层之间的相关性

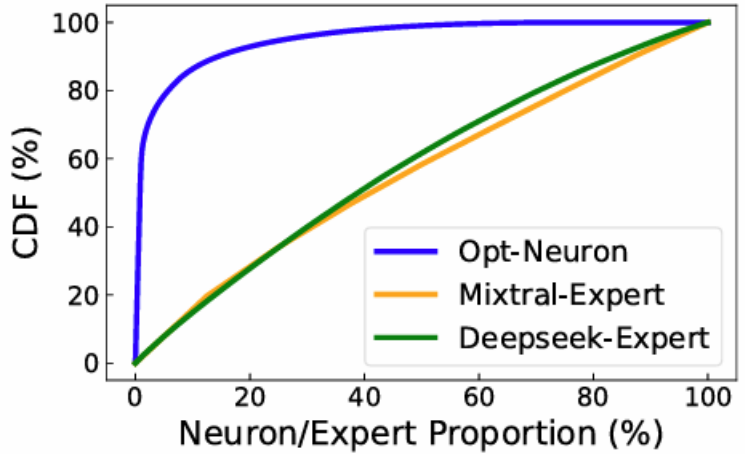


图5: 不同模型中的神经元(或专家)累计激活频率

  - 启发：找不到规律时，将其分解到更细粒度的维度（如时间、空间、层级等），去观察其“局部”或“相邻”单元之间的关联性和连续性。

Idea

- 背景知识
- 相关工作
- 研究动机和发现
- 系统设计
- 实验
- Idea

- 由此需要：在此之下，逆向思维
- 在训练中，专家的激活规律是可以被引导的（比如当前常用的专家负载均衡）
- 能否基于本文的结论，反向指导MoE模型的设计和训练，使得其专家的激活更加具有规律，提升推理性能
- 比如：
  - 增加时间局部损失（鼓励专家在相邻Token上复用）
  - 增加空间局部损失（鼓励专家在相邻层上复用）

请各位老师和同学批评指正

汇报人：姚昌硕

2025年9月11日