arXiv:2603.01058v1 [cs.AR] 1 Mar 2026

# TriMoE：通过卸载利用支持AMX的中央处理器和DIMM-NDP增强GPU，实现高吞吐量混合专家推理

Yudong Pan[1,2], Yintao He[1,2], Tianhua Han[1,2], Lian Liu[1,2], Shixin Zhao[1,2], Zhirong Chen[1,2], Mengdi Wang[1,2], Cangyuan Li[1,2], Yinhe Han[1,2], Ying Wang[1]，2，†[1]SKLP，中国科学院计算技术研究所，北京，中国[2]中国科学院大学，北京，中国
panyudong23@mails.ucas.ac.cn,heyintao19z@ict.ac.cn,wangying2009@ict.ac.cn

## 摘要

为了以经济高效的方式部署大型混合专家（MoE）模型，基于卸载的单GPU异构推理至关重要。尽管卸载冷专家的GPU–中央处理器架构受到主机内存带宽的限制，新兴的GPU-NDP架构利用DIMM-NDP卸载非热专家。然而，非热专家并非同质的内存受限群体：其中存在相当一部分温专家，它们受到较高GPU I/O延迟的严重影响，却能够耗尽NDP计算吞吐量，从而暴露出严重的计算缺口。我们提出了TriMoE，一种新型GPU–中央处理器–NDP架构，通过协同利用支持AMX的中央处理器，将热专家、温专家和冷专家精准映射到其最优计算单元上，以填补这一缺口。我们进一步引入了瓶颈感知的专家调度策略，以及预测驱动的动态重布局与再平衡方案。实验表明，与现有最先进的解决方案相比，TriMoE最高可实现2.83× 加速比。

## 1 引言

混合专家（MoE）模型通过为每个 token 稀疏激活一小组专家，改善了大语言模型（LLM）的效率与质量权衡 [13, 20]。然而，数百个专家的参数总量远超单个高端 GPU 的内存容量 [7, 29]。例如，部署 DeepSeek-V2 至少需要七块 80GB H100 GPU [9], ，成本约为 20 万美元。因此，将专家卸载到主机内存对于单 GPU 部署至关重要。

在混合专家卸载系统中，端到端延迟通常主要由通过 PCIe 获取 GPU 外专家权重时的数据移动所决定，而非纯粹的计算。如图1所示，先前的工作因此从仅 GPU 预取逐步发展到异构设计 [14]。由于 GPU-CPU 方案受主机内存带宽限制，当前系统倾向于采用 GPU-NDP [18],，通过将热专家保留在 GPU 上，并将低负载、内存受限的专家卸载到近存计算引擎，来利用路由稀疏性。

重新审视高吞吐量工作负载（例如，离线 [29] 或之字形批处理 [9]）下的专家激活后，我们发现非热专家并非同质。除了少数热专家和长尾的冷专家外，30% 的专家属于温专家（亦称中等强度专家）。将温专家映射到 GPU，会导致每个专家获得的词元数不足，无法实现高利用率，且无法与PCIe传输重叠；将其映射到近数据处理，则会超出有限的近数据计算预算，抵消卸载冷专家带来的收益。因此，将所有非热专家归为一类的二元 GPU-NDP划分范式并不奏效。

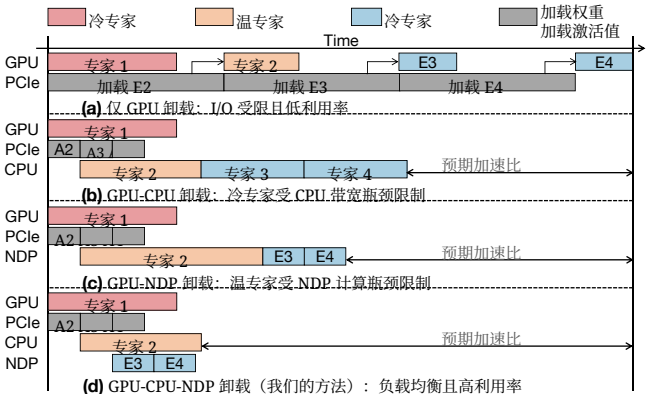



**图1：基线MoE卸载系统与我们所提出架构的执行时间线。**

这一分析促使我们重新审视主机侧计算选项。尽管 GPU-CPU 系统过去一直被认为受到限制，但我们认为，这是因为中央处理器被错误地分配给了带宽密集型冷专家。相比之下，配备矩阵扩展的现代服务器中央处理器（例如 Intel AMX 和 ARM SME）可提供数十万亿次浮点运算每秒 [2, 17],，恰好契合温专家的计算特征。与此同时，DIMM-NDP 仍是处理冷专家的理想选择，它利用高聚合内部带宽，其速度是主机内存系统的 4–8× 倍 [22, 23]。

因此，我们提出 **TriMoE**，这是一种单GPU、高吞吐量的混合专家推理架构，能够在单个系统中独特地协调三个异构计算域。TriMoE 将热专家（例如共享专家）保留在 GPU 上，以发挥峰值吞吐量；将温专家映射到支持AMX的中央处理器，以绕过 PCIe 瓶颈并满足计算需求；并将长尾冷专家卸载到 DIMM近数据处理引擎，以利用其高内部带宽。这种层次化设计有效地避免 GPU 遭受 PCIe 拥塞，并使中央处理器免于处理内存受限任务，从而确保每个计算域专注于其最擅长处理的工作负载。

简而言之，我们的贡献如下：

- **问题识别：**我们首次识别并形式化了高吞吐量混合专家服务中被忽视的温专家类别，并阐明了为何当前前沿的 GPU-NDP 卸载方式本质上效率低下。
- **TriMoE架构与调度器：**我们设计了 TriMoE，这是首个集成 GPU、支持AMX的中央处理器和 DIMM-NDP 的卸载架构，并配备了能够精准

† 通讯作者

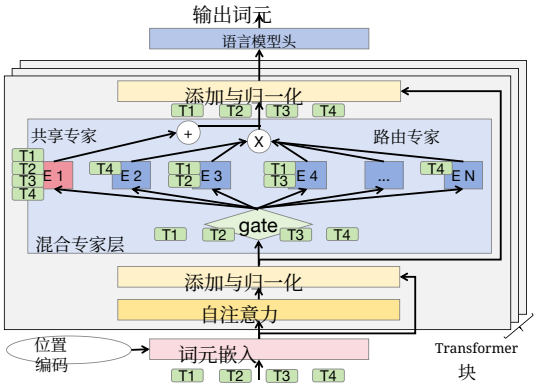



**图 2：包含共享专家和路由专家的混合专家架构。**

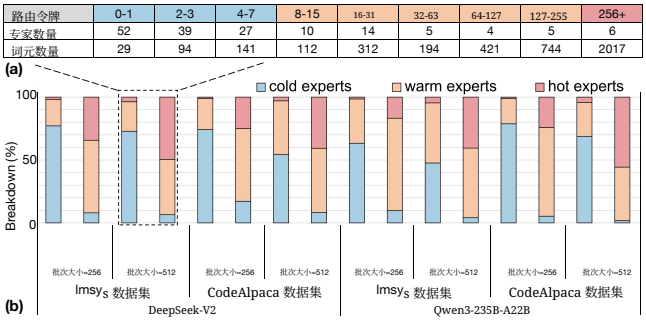



**图 3：不同批次大小、模型和数据集下的专家激活情况[1]。(a) 特定配置下的细粒度激活情况。(b) 所有配置下激活情况的汇总。**

将专家分配到最适合其运行的计算单元，从架构层面解决根本性的调度冲突。

- **动态重新布局与再平衡：**我们设计了一种由预测驱动的策略，通过动态专家重排解决相互冲突的数据布局偏好（GPU/CPU 与近数据处理），并通过冷专家重新平衡防止近数据处理负载偏斜。
- **端到端提升：** 在多个具有代表性的混合专家模型上，TriMoE 在解码阶段实现了 2.12–2.83× 倍加速比，相较于最先进的卸载系统，端到端吞吐量提升了 2.09–2.78× 倍。

# 2 背景

## 2.1 混合专家架构与推理

混合专家（MoE）模型通过稀疏激活部分专家子集，在不使计算成本线性增长的情况下扩展参数量 [13, 20, 21, 31]。如图 2 所示，混合专家层以一个门控函数和 N 个并行专家取代标准前馈层。在推理过程中，门控函数会计算每个输入词元分配给各个专家的概率，并选择 top-k 个专家进行处理。随后，这 k 个专家的输出通过加权和进行合并，形成该词元的最终输出，再传递给后续层。实际应用中，专家总数（N）通常从几十到几百不等，而激活的专家数量（K）通常是较小的整数。此外，DeepSeek [20] 等现代架构同时引入了共享专家和路由专家。例如，DeepSeek V2 使用 2 个由所有词元激活的共享专家，以及 160 个路由专家，并通过 top-k 门控从中选出 6 个。这种混合设计使模型能够在参数规模与计算成本之间实现良好平衡。

## 2.2 混合专家推理中的卸载

混合专家模型的参数规模庞大，且其架构具有稀疏性，因此通常需要成本高昂的多 GPU 部署。基于卸载的推理系统应运而生，成为一种经济高效的单 GPU 解决方案 [26, 30],，利用主机内存存储大部分（>90%）专家权重。然而，PCIe 总线（PCIe 5.0 的带宽为 64 GB/s）造成了严重的 I/O 瓶颈，因为其带宽远低于 GPU 的计算能力。

因此，最先进的卸载系统面向高吞吐量场景（离线或之字形批处理），以摊销 I/O 开销 [9, 29]。为尽量减少解码阶段专家激活高度偏斜时的专家数据移动，相关设计已从预取发展到异构计算范式。热专家常驻 GPU，而其他非热专家则利用主机可用的计算资源在主机上执行 [14, 37]。由于异构 GPU-CPU 系统仍受主机内存带宽限制，越来越多的方案开始采用 DIMM-NDP [18, 22]。例如，秩级 PU 利用比主机高 4-8×倍的聚合内部带宽，在近数据处处理专家 [23]，从而缓解 I/O 瓶颈并提高吞吐量。相反，在预填充阶段，专家被广泛激活，GPU 计算基本可以隐藏 I/O 开销。

# 3 动机

## 3.1 温专家的调度困境

当前最先进的 GPU-NDP 系统假设，所有非热专家都构成一个均匀的、受内存带宽限制的子集，适合由 DIMM-NDP 处理 [18, 28]。然而，我们的分析表明，这种假设过于简单。如图 3 所示，对多个具有代表性的混合专家模型和真实世界数据集进行激活研究后发现，非热专家之间存在显著差异：长尾冷专家占所有专家的 70% 以上，但仅处理 8% 的词元；而 20%–40% 的温专家处理多达 70% 的词元。这种差异给 GPU-NDP 带来了调度困境：（1）**GPU 侧 I/O 瓶颈**。这一瓶颈体现在两个方面。首先，如图 5(a) 所示，即使专家驻留在高带宽内存中，H100 GPU 也至少需要每个专家处理 256 个词元才能达到 30% 的利用率。温专家远低于这一阈值。更关键的是，这些未驻留的温专家必须通过 PCIe 获取。由于词元数量少，其计算时间过短，无法与显著的 I/O 延迟重叠，迫使 GPU 停顿。因此，GPU 总体利用率降至 8.6% 以下。（2）**NDP 侧计算瓶颈**。在 NDP 上调度温专家会超出其计算预算，使 NDP 执行时间增加 7×，抵消卸载冷专家带来的收益。

[1]此处的“批次大小”指离线大批次推理，或聚合多个小批次的之字形批处理。

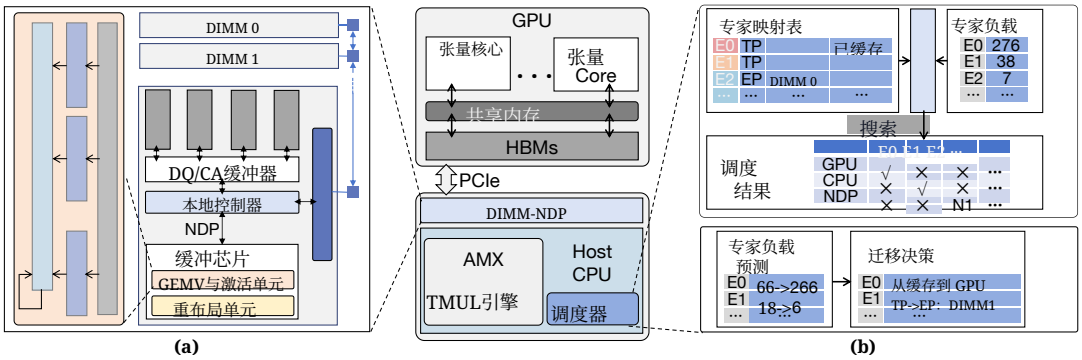



**图4：TriMoE概述。(a) DIMM-NDP架构。(b) 在线专家调度与离线专家重排-再平衡。**

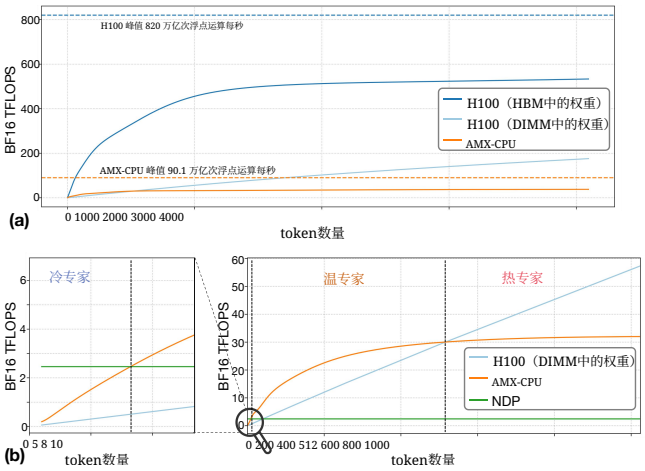



**图5：计算特性分析。(a) 实测吞吐量与令牌数量的关系。(b) 经验性GPU-CPU-近数据处理屋顶线。**

## 3.2 机遇：AMX-CPU弥合GPU与近数据处理之间的计算差距

第 3.1 节中的调度困境源于 GPU 的高吞吐量与 NDP 有限的计算能力之间存在显著差距。温专家正处于这一差距之中，因此促使我们重新审视 NDP 设计大多忽略的中央处理器能力。尽管第 2.2 节指出，GPU-中央处理器系统受内存带宽限制，但我们认为，问题的根源在于将中央处理器错误地分配给带宽密集型冷专家，而中央处理器并不适合处理这类专家。配备矩阵扩展指令集的服务器级中央处理器，例如 Intel AMX 和 ARM SME，在执行温专家（处理数十至数百个词元）时可提供 10–40 万亿次浮点运算每秒 [2],，如图 5(b) 所示，其性能实际上超过了受 PCIe 限制的 GPU 和计算能力有限的 NDP。这一发现促成了明确的三方协同设计：支持 AMX 的中央处理器负责处理温专家，以填补计算能力差距；双列直插式内存模块-NDP 单元利用其高内部带宽处理长尾冷专家，从而避免中央处理器的内存瓶颈；GPU 则继续高效处理热专家。这种分工让每种资源都承担最适合自身的任务。

**挑战：**然而，实现这一协同设计面临两项核心挑战：（*1*）编排复杂性：如何设计一个轻量级调度器，使其能够动态适应波动的专家负载，并在三个具有不同计算-I/O权衡的异构域之间优化放置。（*2*）数据管理冲突：如何设计一种自适应数据放置策略，以调和布局偏好之间的结构性冲突 [23, 35]（中央处理器/GPU采用条带化，而近数据处理采用局部化），同时缓解由不均衡的冷专家激活所导致的NDP负载偏斜 [17]。

## 4 TriMoE

本节详细介绍 TriMoE 的设计与实现。我们首先介绍整体架构（第 4.1 节），随后介绍两种应对这些挑战的新策略：瓶颈感知的贪心完工时间专家调度（第 4.2 节），以及预测驱动的专家重新布局与再平衡（第 4.3 节）。

## 4.1 TriMoE架构

图 4 展示了 TriMoE 的整体架构。该架构通过为服务器级单 GPU系统增配三种异构计算资源——高性能 GPU、支持AMX的中央处理器和 DIMM-NDP——实现高吞吐量的 MoE 卸载。

**高性能 GPU：**凭借数百 TFLOPS [3, 4], 的算力，GPU 是执行高密度任务的核心计算单元。在预填充阶段，它执行所有推理计算。在解码阶段，它专注于多层感知机、注意力，以及预取至 HBM 的热专家计算。在存储方面，多层感知机和共享专家权重存放于 GPU HBM 中，而庞大的键值缓存和所有路由专家则卸载至主机 DIMM [29]。

**支持AMX的中央处理器：**具备矩阵扩展的中央处理器拥有可观的计算能力。例如，英特尔至强蓝宝石急流（理论性能为 90.1 万亿次浮点运算每秒）的通用矩阵乘法吞吐量实测达到 A100 的 22% [17]。与依赖远程 PCIe 传输的 GPU 不同，中央处理器可直接处理驻留于主机的数据。TriMoE 利用这一特性，将 AMX-CPU 指定为温专家专用的计算域，最大限度地发挥主机侧资源的效用；若将这些资源分配给冷专家，则会受内存带宽限制。

**DIMM-NDP：**为了处理长尾冷专家，我们采用了基于中心缓冲区的近存计算，而非面积开销较高的存储体级近存计算 [1, 5, 15, 25]。如图 4(a) 所示，近存处理单元部署在

双列直插式内存模块的缓冲芯片上。这种设计实现了高效的权衡：它利用较高的总内部带宽（8× 主机带宽）高效处理冷专家，同时完全兼容常规内存访问，不会干扰中央处理器或 GPU 对双列直插式内存模块数据的正常访问 [1, 5]。

*GEMV*与激活单元：每个 DIMM-NDP 都配备了专用的通用矩阵向量乘法单元。该单元包含 256 个乘法器和一棵多级加法树。每个 128 位乘法器通常以位串行方式运行，[6],可同时处理 8 个半精度浮点数；此外，还使用一个 256KB 的内部缓冲区来存储中间激活值。为了支持混合专家层中的 SiLU 等非线性函数，该单元还集成了一个激活模块，主要由 256 个半精度浮点数指数与乘法单元组成。

重布局单元与 *DIMM-Link*：核心架构挑战在于数据布局偏好冲突和 NDP 负载不平衡。为解决这两项挑战，我们需要能够在 DIMM 之间快速进行数据重排和迁移，详见第 4.3 节。因此，我们采用 DIMM-Link [38] 并引入重布局单元。DIMM-Link 提供一条 25 GB/s 的无主机参与跨 DIMM 传输总线。重布局单元充当控制引擎，利用 DIMM-Link 高效执行两项关键后台任务：（1）重新布局：支持条带化布局与局部化布局之间的数据转换；（2）再平衡：支持在 DIMM 之间快速迁移局部化布局的冷专家。这种无主机参与的数据操作比由中央处理器驱动的数据移动效率高出 62× 。

**调度器：** 调度器统筹所有异构资源，执行在线专家调度以及动态专家重排和再平衡，具体内容将在后续章节中介绍。

## 4.2 瓶颈感知的贪心完工时间专家调度

在 TriMoE 的三路异构平台上进行调度颇具挑战。将专家映射到最适合的设备上，旨在最大化资源利用率，但这一理论上简单且最优的策略可能会造成全局瓶颈。关键在于平衡三个领域的负载，以最小化全局完工时间。我们提出了一种两阶段调度策略：首先执行基于成本模型的贪心初始分配，随后通过迭代式的瓶颈感知精化来优化最终完工时间。

**专家执行成本模型：** 第一步是估算任意专家 $E_i$ 在所有可能路径上的执行成本，并选择最合适的路径。该成本模型依赖于两个运行时输入，如图4(b)所示：来自门控函数的专家负载（令牌数量） $L_i$，以及专家的映射和数据布局，这些信息共同表示为 $M_i$。为了准确建模计算延迟，我们离线分析 GPU 和中央处理器在不同令牌数量下的性能，为 $f_{calc_gpu}(L_i)$ 和 $f_{calc_cpu}(L_i)$ 构建查找表 [16]。

*1) GPU*执行成本：我们评估两种路径。公式 1 描述缓存命中，此时专家驻留在 GPU HBM 中，成本主要由计算决定。公式 2 描述缓存未命中，需要从主机按需加载。其成本由 GPU 计算、PCIe 权重传输和主机DIMM权重读取决定（$T_{DRAM}$）。$T_{DRAM}$时间取决于专家权重的布局（$M_i$），布局可以是条带化，此时可使用总内存带宽，也可以是局部化，此时只能使用单个 DIMM 的带宽。

$$T_{GPU_Hit}(E_i) = f_{calc_gpu}(L_i) \tag{1}$$

$$T_{GPU_Miss}(E_i) = \max\left(f_{calc_gpu}(L_i), T_{PCIe}, T_{DRAM}(W_i, M_i)\right) \tag{2}$$

*2) CPU*执行成本：对于 AMX-CPU，其成本是计算时间与DIMM权重读取时间（$T_{DRAM}$）中的较大值，因为 $T_{DRAM}$同样取决于专家的布局。

$$T_{CPU}(E_i) = \max\left(f_{calc_cpu}(L_i), T_{DRAM}(W_i, M_i)\right) \tag{3}$$

*3) NDP*执行成本：与集中式GPU和中央处理器不同，DIMM-NDP单元以并行方式运行。我们严格限制此路径仅用于具有本地化布局的专家，以避免跨多个 DIMM进行输入广播和结果归约所带来的高开销。因此，只有存放其权重的特定 DIMM 才能执行专家 $E_i$。其成本是近存处理计算与通过DIMM内部带宽读取权重所需时间中的较大值。

$$T_{NDP}(E_i) = \max\left(f_{calc_ndp}(L_i), T_{Internal}(W_i)\right) \tag{4}$$

**总体完工时间建模：** 初始专家分配完成后，我们通过汇总各计算域内的成本来构建全局完工时间。令 $S_{GPU}$、$S_{CPU}$ 和 $S^d_{NDP}$分别表示分配给 GPU、中央处理器和第 $d$个DIMM-NDP单元的专家集合。

GPU 和中央处理器充当集中式串行处理器。它们的总执行时间 $T^{GPU}_{total}$ 和 $T^{CPU}_{total}$，只是各自分配到的专家的执行成本之和（由公式 1–3 推导得出）：

$$T^{GPU/CPU}_{total} = \sum_{E_i \in S_{GPU/CPU}} T_{GPU/CPU}(E_i) \tag{5}$$

相比之下，DIMM-NDP 单元并行运行，因此近存处理域的延迟由瓶颈 DIMM 决定。至关重要的是，我们的模型考虑了 内存访问争用：DIMM 不仅在执行本地 NDP 计算时处于忙碌状态，在为 GPU 或中央处理器处理权重获取请求时也同样如此。DIMM $d$ 的总活跃时间为：

$$T_{DIMM}(d) = \sum_{E_i \in S^d_{NDP}} T_{NDP}(E_i) + T^d_{contention} \tag{6}$$

其中，$T^d_{contention}$ 表示主机处理器针对 DIMM $d$ 发起条带化或局部化权重读取所导致的累计 DRAM 访问延迟。全局完工时间最终由三个域中最慢的一个决定：

$$T_{Makespan} = \max\left(T^{GPU}_{total}, T^{CPU}_{total}, \max_{d \in \mathbb{D}} T_{DIMM}(d)\right) \tag{7}$$

**瓶颈感知优化：** 初始专家分配可能导致全局负载不平衡，因此算法会进入迭代式瓶颈校正阶段。在每次迭代中，算法首先识别瓶颈设备（总耗时最长的设备），并选取该设备上成本最高的专家作为迁移候选项。随后，算法评估将该专家重新分配到另外两台设备的方案，并计算每种潜在迁移所产生的新全局完工时间。算法采用贪心策略，选择能够使新全局完工时间最短的方案。如果两种迁移方案产生的新完工时间相同，则选择接收设备时间增量（delta）最小的方案，以打破平局。这种迭代精化会持续进行，直到

无法通过重新分配进一步缩短全局完工时间，或达到最大迭代次数限制。

## 4.3 预测驱动的专家重新布局与再平衡

在线专家调度的效果受到专家权重放置方式的制约。尽管我们通过离线轨迹分析设置了优化的初始布局（例如，将冷专家集中放置在单个双列直插式内存模块上），但由于动态专家激活 [11, 32], 这种静态放置在运行时会变得不再理想。这会引发两个核心问题。首先，专家身份可能发生变化，从而产生**布局偏好冲突** 。例如，冷专家可能变为热专家，导致其布局与中央处理器偏好的条带布局不匹配。其次，当冷专家在各双列直插式内存模块上的激活不均时，会导致某些 DIMM-NDP 负载过重，形成瓶颈，造成**NDP负载不平衡** 。为解决这些问题，我们设计了一种预测驱动策略，利用专家激活的时间局部性，执行异步重新布局和再平衡 [8, 30]。

**专家负载预测器：** 如图4(b)所示，我们引入了一种轻量级专家负载预测器。该预测器为每个专家的负载维护一个指数移动平均（EMA） [10, 34],，并在每个解码步骤后更新：

$$EMA_e(t) = \alpha \cdot F_e(t) + (1 - \alpha) \cdot EMA_e(t-1) \tag{8}$$

其中， $F_e(t)$表示专家在步骤 $t$ 的实际负载， $\alpha = 0.3$ 经过经验调优，用于预测专家的负载趋势并抑制噪声，准确率达到 >78%。

**专家迁移决策：** 当某一层的专家计算完成后，预测器会估算下一层的负载趋势，并触发三类后台操作。*(1)* 热专家预取：如果预测某个专家为热专家，运行时会触发高优先级任务，通过 PCIe 将其权重预取至 GPU HBM。*(2)* 动态重新布局：如果专家的数据布局与其最佳执行单元不匹配，重布局单元会利用 DIMM-Link 将其数据布局在本地布局与条带布局之间转换。*(3)* 冷专家再平衡：为解决 NDP负载不平衡，系统会计算每个双列直插式内存模块上冷专家的预测负载总和。如果检测到负载倾斜，系统会通过 DIMM-Link，以贪心方式逐个将冷专家从负载最重的双列直插式内存模块迁移至最空闲的双列直插式内存模块。我们希望将这些迁移的开销隐藏在重叠窗口内；该窗口由当前层在 GPU 上的多层感知机和注意力计算提供。我们的策略会根据预测收益对所有可行的迁移任务进行排序，并按优先级贪心执行，直至这些任务的累计估计耗时填满窗口预算。这样可以确保有限的 DIMM-Link带宽用于最关键的任务。

我们的评估表明，该预测器的迁移决策准确率达到 78%，仅需 38 KB 内存开销来存储必要的元数据，端到端性能提升 1.16×。

# 5 评估

## 5.1 实验设置

*5.1.1 TriMoE 系统。*为评估 TriMoE，我们搭建了一个异构原型系统，其中包括一块配备 80GB 高带宽内存的 NVIDIA H100 PCIe GPU，以及一颗支持 AMX 的 Intel Xeon Platinum 8470 中央处理器（8 通道内存）和 16 个可提供高内部带宽的 DIMM-NDP。详细配置见表1。我们采用 PCIe 5.0，为主机到 GPU 的数据传输提供 64GB/s 的单向带宽。在软件栈方面，我们使用 vLLM 0.8.1 [19] 和 KTransformers [2]，分别为 GPU 和 AMX-CPU 实现高效内核。我们基于 Ramulator 2.0 [24] 开发了一个周期精确模拟器，用于评估 DIMM-NDP 的性能。对于近存处理单元，我们使用寄存器传输级（RTL）实现，并采用台积电7纳米工艺，通过 Synopsys Design Compiler [27] 进行综合。

**表1：系统配置。**

| | |
|---|---|
| **GPU** | H100 80GB：819.6 万亿次浮点运算每秒（BF16）、2.04 TB/s 带宽、64 GB/s PCIe带宽 |
| **CPU** | 内存：8 个通道 × 2 双列直插式内存模块，容量为 2TB<br>AMX：90.1 万亿次浮点运算每秒（BF16），307.2 GB/s 带宽 |
| **DIMM-NDP** | **DIMM参数**<br>DDR5-4800，128GB/双列直插式内存模块，4 个存储器秩 × 8个存储体组 × 4 个存储体<br>CCDL=12:CCDS=8:RRDS=8:RRDL=12:FAW=32:RP=34:BL=8:CL=40:WR=72<br>**NDP**<br>256 GFLOPS（BF16）和 153.6 GB/s 带宽，每个近数据处理单元的 1.13$mm^2$ 面积开销<br>**DIMM-Link 参数**<br>25Gb/s/通道，8 × 个通道（每条链路 25GB/s） |

*5.1.2 基线。* 我们将 TriMoE 与三种先进的混合专家卸载系统进行比较。(1) Klotski [9]: 仅 GPU。该基线通过预取热专家，并在热专家计算期间加载冷专家权重，最大限度地提高 PCIe 重叠；(2) EnhancedKtrans- former [2]: GPU-CPU。我们在其使用 AMX-CPU 处理所有非共享专家的基础上，增加了热专家的预取和按需加载，使其成为该架构下最强的基线 [37]；(3) MoNDE [18]: GPU-NDP。该系统通过对权重迁移（至 GPU）与激活迁移（至近数据处理）的成本权衡进行建模，在 GPU 和近数据处理之间平衡负载。

*5.1.3 工作负载与指标。* 我们选取了三种具有代表性的现代混合专家模型，其规格详见表 2，并使用 LMSys [36]和 CodeAlpaca [12]数据集提取真实的专家激活轨迹，以构建工作负载。由于卸载系统主要面向高吞吐量需求 [29], ，我们重点考察通过 zigzag 或离线批处理聚合而成的 256–768 大批次，并辅以小批次下的鲁棒性评估。鉴于混合专家解码阶段占推理时间的 90% 以上，我们将混合专家层延迟和端到端吞吐量作为核心指标。

**表 2：MoE 模型配置。**

| 模型 | 参数 | 共享专家 | 路由专家 | top-$k$ | 专家参数 |
|---|---|---|---|---|---|
| DeepSeek-V2 [20] | 236B | 2 | 160 | 6 | 422GB |
| Qwen3-235B-A22B [31] | 235B | 0 | 128 | 8 | 423GB |
| GLM-4.5-Air [33] | 106B | 1 | 128 | 8 | 190GB |

## 5.2 TriMoE 性能

*5.2.1 混合专家层解码加速。* 我们的评估针对推理中的主要瓶颈——混合专家解码阶段。如图 6 所示，在所有模型上，TriMoE 相较于表现最佳的基线实现了平均 2.12–2.83× 的加速比。为确定这些提升的来源，我们分析了资源利用率（表 3）。Klotski 受限于

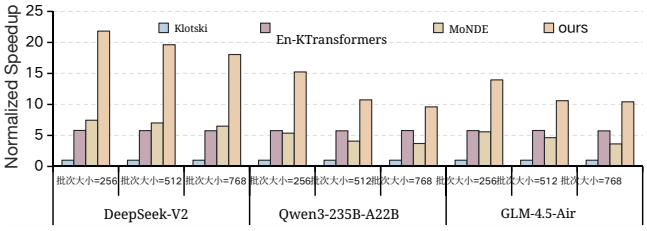



**图 6：相较于 Klotski 的混合专家解码加速。**

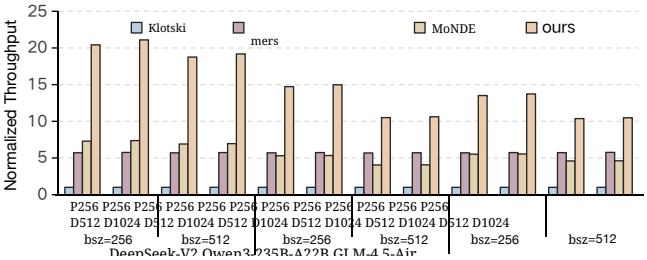



**图7：相较于 Klotski 的端到端吞吐量。**

严重的 PCIe 瓶颈，导致 GPU 利用率仅为 28.6%。En-KTransformers 在中央处理器上处理冷专家时受主机带宽限制，CPU 利用率最高仅为 42%。与此同时，MoNDE 的 NDP计算能力有限，迫使 GPU 承担大量专家计算，未能缓解 PCIe拥塞。相比之下，TriMoE 将温专家卸载到启用 AMX 的中央处理器，并将冷专家分配给近数据处理，使三个计算域的平均利用率达到 76.2%。

**表 3：资源利用率对比**

| 资源 | Klotski | En-KTransformers | MoNDE | **ours** |
|---|---|---|---|---|
| GPU计算利用率 | 28.6% | 57.6% | 33.9% | **66.0%** |
| CPU计算利用率 | × | 42% | × | **74.9%** |
| NDP计算利用率 | × | × | 70.1% | **87.8%** |

*5.2.2* 端到端性能。 我们进一步评估端到端性能。如图7所示，与三个模型中最强的基线相比，TriMoE 的吞吐量分别提高了 2.78×、2.22× 和 2.09× 。这证实了混合专家解码层的性能提升能够有效转化为系统级性能改进，并且随着输出长度增加，这种提升尤为显著。

## 5.3 消融实验

如图8所示，我们以**GPU-NDP** 基线为起点，通过消融实验量化各组件的贡献。首先，结合 AMX-CPU 与贪心调度（**+中央处理器**）可获得最显著的 1.75× 加速比。这验证了我们的假设：AMX-CPU 能有效处理温专家，从而打破 GPU I/O 与 NDP 计算瓶颈之间的僵局。接下来，启用瓶颈感知精化（**+精化**）可进一步提升 1.28×，证实该算法能够纠正贪心映射不平衡，从而尽可能缩短全局完工时间。最后，预测驱动的重新布局与再平衡（**+重新布局**）又带来 1.16×

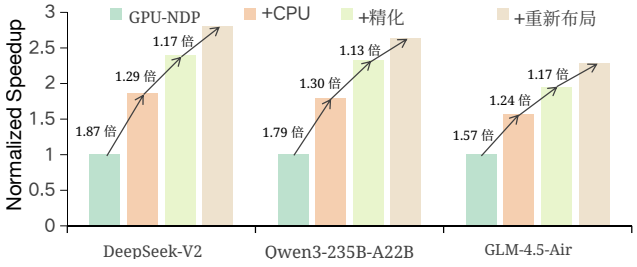



**图8：批次大小为 512 时 TriMoE 的消融实验。**

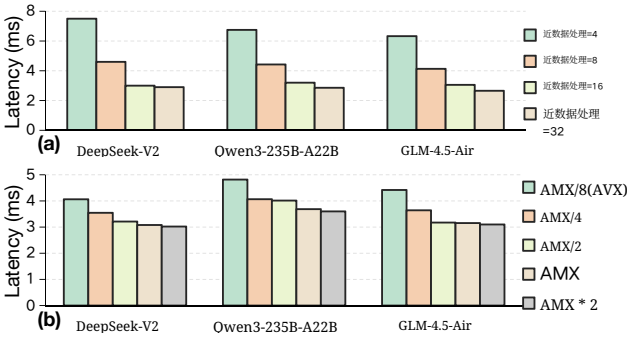



**图9：近数据处理数量和 CPU TFLOPS 的敏感性研究。**

提升。在预测准确率超过 78% 的情况下，该机制成功解决了动态激活导致的运行时布局冲突和 NDP负载偏斜，从而确保系统效率持续稳定。

## 5.4 敏感性分析

*5.4.1* 近数据处理器数量。图9(a)考察了配备近数据处理器的双列直插式内存模块数量。延迟在数量达到16个后趋于稳定。这反映了该工作负载的一个关键特征：冷专家虽然数量众多，但总计算密度较低。因此，配置足量近数据处理器即可提供充足带宽，以处理所有冷专家。

*5.4.2* 中央处理器计算能力。图9(b)将中央处理器的计算能力从旧款 AVX 水平（≈0.125×）逐步提升至 2.0× AMX。虽然 AVX 水平下的延迟较高，但当容量达到 AMX 基线的 0.5× 时，曲线便显著趋于平缓。这证实了现代 AMX 扩展具备充足的计算能力，使中端中央处理器也能有效弥补温专家缺口。

## 5.5 鲁棒性与开销

为使用通义千问模型在批次大小为 128, 64, 32, 2 72× 2 18× 1 82× 时评估稳健性，TriMoE相对于最强基线仍保持.、.和.的加速比，证实其在不同吞吐量需求下的有效性。我们还分析了运行时开销。借助 DIMM-Link无需主机参与的并行DIMM间传输能力，我们发现，迁移至多四个专家的 ~0.63毫秒开销完全被 ~0.68毫秒的并发GPU计算所掩盖。鉴于预测器准确率超过78%（这使得如此大规模的传输并不频繁），测得的总体在线迁移开销仍限制在3.3%以下。

## 6 结论

TriMoE从专家异构性的视角重新审视MoE卸载，并揭示了二元GPU–近数据处理设计在处理温专家时的根本低效。通过联合利用GPU处理热专家、支持AMX的中央处理器处理温专家以及DIMM-NDP处理冷专家，TriMoE将每一类专家匹配到最适合的计算域。结合我们的瓶颈感知调度以及预测驱动的重新布局与再平衡，这一三域架构相对于最先进的卸载系统实现了高达2.83× 的加速比。

## 参考文献


[1] Mohammad Alian、Seung Won Min、Hadi Asgharimoghaddam、Ashutosh Dhar、Dong Kai Wang、Thomas Roewer、Adam McPadden、Oliver O'Halloran、Deming Chen、Jinjun Xiong 等。2018。具备内存通道网络的应用透明型近内存处理架构。载于*2018*年第*51*届*IEEE/ACM*微体系结构国际研讨会（*MICRO*）。IEEE，第802–814页。[2] Hongtao Chen、Weiyu Xie、Boxin Zhang、Jingqi Tang、Jiahao Wang、Jianwei Dong、Shaoyuan Chen、Ziwei Yuan、Chen Lin、Chengyu Qiu 等。2025。KTransformers：释放面向 MoE 模型的 CPU/GPU混合推理的全部潜力。载于*ACM SIGOPS*第*31*届操作系统原理研讨会论文集。第1014–1029页。[3] Jack Choquette。2023。Nvidia Hopper H100 GPU：扩展性能。*IEEE Micro* 43卷，3期（2023），第9–17页。[4] Jack Choquette、Wishwesh Gandhi、Olivier Giroux、Nick Stam 和 Ronny Krashinsky。2021。Nvidia A100 Tensor Core GPU：性能与创新。*IEEE Micro* 41卷，2期（2021），第29–35页。[5] Jason Cong、Zhenman Fang、Michael Gill、Farnoosh Javadi 和 Glenn Reinman。2017。AIM：通过可扩展且非侵入式的加速器介入式内存加速计算基因组学。载于国际内存系统研讨会论文集。第3–14页。[6] Fabrice Devaux。2019。真正的存内处理加速器。载于*2019*年*IEEE Hot Chips*第*31*届研讨会（*HCS*）。IEEE计算机学会，第1–24页。[7] Artyom Eliseev 和 Denis Mazur。2023。通过卸载实现混合专家语言模型的快速推理。*arXiv*预印本 *arXiv:2312.17238*（2023）。[8] Zhiyuan Fang、Zicong Hong、Yuegui Huang、Yufeng Lyu、Wuhui Chen、Yue Yu、Fan Yu 和 Zibin Zheng。2025。通过跨层门控实现混合专家推理中的精准专家预测。*arXiv*电子印本（2025），arXiv–2502。[9] Zhiyuan Fang、Yuegui Huang、Zicong Hong、Yufeng Lyu、Wuhui Chen、Yue Yu、Fan Yu 和 Zibin Zheng。2025。Klotski：通过专家感知多批次流水线实现高效的混合专家推理。载于第*30*届*ACM*编程语言与操作系统架构支持国际会议论文集，第*2*卷。第574–588页。[10] David Haynes、Steven Corns 和 Ganesh Kumar Venayagamoorthy。2012。一种指数移动平均算法。载于*2012*年*IEEE*进化计算大会。IEEE，第1–8页。[11] Siqi He、Haozhe Zhu、Jiapei Zheng、Lizhou Wu、Bo Jiao、Qi Liu、Xiaoyang Zeng 和 Chixiao Chen。2025。Hydra：利用专家流行度，在Chiplet系统上实现高效的混合专家推理。载于*2025*年第*62*届*ACM/IEEE*设计自动化会议（*DAC*）。IEEE，第1–7页。[12] Hugging Face H4。2023。CodeAlpaca_20K。https://huggingface.co/datasets/ HuggingFaceH4/CodeAlpaca_20K。[13] Albert Q Jiang、Alexandre Sablayrolles、Antoine Roux、Arthur Mensch、Blanche Savary、Chris Bamford、Devendra Singh Chaplot、Diego de las Casas、Emma Bou Hanna、Florian Bressand 等。2024。Mixtral of experts。*arXiv*预印本 *arXiv:2401.04088*（2024）。[14] Keisuke Kamahori、Tian Tang、Yile Gu、Kan Zhu 和 Baris Kasikci。2024。Fiddler：用于混合专家模型快速推理的CPU-GPU编排。*arXiv*预印本 *arXiv:2402.07033*（2024）。[15] Liu Ke、Udit Gupta、Benjamin Youngjae Cho、David Brooks、Vikas Chandra、Utku Diril、Amin Firoozshahian、Kim Hazelwood、Bill Jia、Hsien-Hsin S Lee 等。2020。Recnmp：利用近内存处理加速个性化推荐。载于*2020*年*ACM/IEEE*第*47*届计算机体系结构国际研讨会（*ISCA*）。IEEE，第790–803页。[16] Liu Ke、Udit Gupta、Mark Hempstead、Carole-Jean Wu、Hsien-Hsin S Lee 和 Xuan Zhang。2022。Hercules：面向大规模个性化推荐的异构性感知推理服务。载于*2022*年*IEEE*高性能计算机体系结构国际研讨会（*HPCA*）。IEEE，第141–154页。[17] Hyungyo Kim、Nachuan Wang、Qirong Xia、Jinghan Huang、Amir Yazdanbakhsh 和 Nam Sung Kim。2025。LIA：通过协同AMX启用CPU-GPU计算和CXL卸载实现单GPU大语言模型推理加速。载于第*52*届计算机体系结构国际研讨会论文集。第544–558页。[18] Taehyun Kim、Kwanseok Choi、Youngmock Cho、Jaehoon Cho、Hyuk-Jae Lee 和 Jaewoong Sim。2024。MoNDE：面向大规模稀疏模型的近数据专家混合。载于第61届ACM/IEEE设计自动化会议论文集。第1–6页。[19] Woosuk Kwon、Zhuohan Li、Siyuan Zhuang、Ying Sheng、Lianmin Zheng、Cody Hao Yu、Joseph Gonzalez、Hao Zhang 和 Ion Stoica。2023。使用PagedAttention实现大语言模型服务的高效内存管理。载于第29届操作系统原理研讨会论文集。第611–626页。[20] Aixin Liu、Bei Feng、Bin Wang、Bingxuan Wang、Bo Liu、Chenggang Zhao、Chengqi Dengr、Chong Ruan、Damai Dai、Daya Guo 等。2024。DeepSeek-V2：一种强大、经济且高效的混合专家语言模型。arXiv预印本 arXiv:2405.04434（2024）。[21] Aixin Liu、Bei Feng、Bing Xue、Bingxuan Wang、Bochao Wu、Chengda Lu、Chenggang Zhao、Chengqi Deng、Chenyu Zhang、Chong Ruan 等。2024。DeepSeek-V3技术报告。arXiv预印本 arXiv:2412.19437（2024）。[22] Lian Liu、Shixin Zhao、Bing Li、Haimeng Ren、Zhaohui Xu、Mengdi Wang、Xiaowei Li、Yinhe Han 和 Ying Wang。2025。让每个人都能负担得起大语言模型推理：用NDP-DIMM扩展GPU内存。载于2025年IEEE高性能计算机体系结构国际研讨会（HPCA）。IEEE，第1751–1765页。[23] Qingyuan Liu、Liyan Chen、Yanning Yang、Haocheng Wang、Dong Du、Zhigang Mao、Naifeng Jing、Yubin Xia 和 Haibo Chen。2025。L3：面向可扩展长上下文大语言模型推理的DIMM-PIM集成架构与协同机制。arXiv预印本 arXiv:2504.17584（2025）。[24] Haocong Luo、Yahya Can Tuğrul、F Nisa Bostancı、Ataberk Olgun、A Giray Yağlıkçı 和 Onur Mutlu。2023。Ramulator 2.0：一种现代化、模块化且可扩展的DRAM模拟器。IEEE计算机体系结构快报23卷，1期（2023），第112–116页。[25] Jaehyun Park、Jaewan Choi、Kwanhee Kyung、Michael Jaemin Kim、Yongsuk Kwon、Nam Sung Kim 和 Jung Ho Ahn。2024。AttAcc！释放存内计算在基于Transformer的批量生成模型推理中的潜力。载于第29届ACM编程语言与操作系统架构支持国际会议论文集，第2卷。第103–119页。[26] Ying Sheng、Lianmin Zheng、Binhang Yuan、Zhuohan Li、Max Ryabinin、Beidi Chen、Percy Liang、Christopher Ré、Ion Stoica 和 Ce Zhang。2023。FlexGen：使用单个GPU实现大语言模型的高吞吐量生成式推理。载于国际机器学习会议。PMLR，第31094–31116页。[27] 新思科技。2025。Design Compiler。https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/design-compiler.html。[28] Lizhou Wu、Haozhe Zhu、Siqi He、Xuanda Lin、Xiaoyang Zeng 和 Chixiao Chen。2025。PIMoE：通过感知节流的任务卸载，实现MoE Transformer在NPU-PIM系统上的高效部署。载于2025年第62届ACM/IEEE设计自动化会议（DAC）。IEEE，第1–7页。[29] Tairan Xu、Leyang Xue、Zhan Lu、Adrian Jackson 和 Luo Mai。2025。MoE-Gen：通过基于模块的批处理，在单GPU上实现高吞吐量混合专家推理。arXiv预印本 arXiv:2503.09716（2025）。[30] Leyang Xue、Yao Fu、Zhan Lu、Luo Mai 和 Mahesh Marina。2024。MoE-Infinity：面向高效卸载的MoE模型服务。arXiv预印本 arXiv:2401.14361 （ 2024 ）。[31] An Yang 、 Anfeng Li 、 Baosong Yang 、 Beichen Zhang、Binyuan Hui、Bo Zheng、Bowen Yu、Chang Gao、Chengen Huang、Chenxu Lv 等。2025。Qwen3技术报告。arXiv预印本 arXiv:2505.09388（2025）。[32] Hanfei Yu、Xingqi Cui、Hong Zhang 和 Hao Wang。2025。fMoE：面向大规模混合专家服务的细粒度专家卸载。arXiv预印本 arXiv:2502.05370（2025）。[33] Aohan Zeng、Xin Lv、Qinkai Zheng、Zhenyu Hou、Bin Chen、Chengxing Xie、Cunxiang Wang、Da Yin、Hao Zeng、Jiajie Zhang 等。2025。GLM-4.5：具备智能体、推理和编程（ARC）能力的基础模型。arXiv预印本 arXiv:2508.06471（2025）。[34] Yan Zeng、Chengchuang Huang、Yipeng Mei、Lifu Zhang、Teng Su、Wei Ye、Wenqi Shi 和 Shengnan Wang。2025。EfficientMoE：通过自适应负载均衡优化混合专家模型训练。IEEE并行与分布式系统汇刊（2025）。[35] Yilong Zhao、Mingyu Gao、Fangxin Liu、Yiwei Hu、Zongwu Wang、Han Lin、Ji Li、He Xian、Hanlin Dong、Tao Yang 等。2024。UM-PIM：采用统一共享内存空间的基于DRAM的存内计算。载于2024年ACM/IEEE第51届计算机体系结构国际研讨会（ISCA）。IEEE，第644–659页。[36] Lianmin Zheng、Wei-Lin Chiang、Ying Sheng、Tianle Li、Siyuan Zhuang、Zhanghao Wu、Yonghao Zhuang、Zhuohan Li、Zi Lin、Eric P. Xing、Joseph E. Gonzalez、Ion Stoica 和 Hao Zhang。2023。LMSYS-Chat-1M：一个大规模真实世界大语言模型对话数据集。arXiv预印本 arXiv:2309.11998（2023）。[37] Shuzhang Zhong、Yanfan Sun、Ling Liang、Runsheng Wang、Ru Huang 和 Meng Li。2025。HybriMoE：面向高效混合专家推理的CPU-GPU混合调度与缓存管理。arXiv预印本 arXiv:2504.05897（2025）。[38] Zhe Zhou、Cong Li、Fan Yang 和 Guangyu Sun。2023。DIMM-Link：实现高效的双列直插式内存模块间近内存处理通信。载于2023年IEEE高性能计算机体系结构国际研讨会（HPCA）。IEEE，第302–316页。