MoQAE: 通过量化感知专家混合实现长上下文 LLM 推理的混合精度量化

Wei Tao[♠♡], **Haocheng Lu**[♠♡], **Xiaoyang Qu**[♡*], **Bin Zhang**[♠♡], **Kai Lu**[♠*], **Jiguang Wan**[♠], **Jianzong Wang**[♡]
[♠]Huazhong University of Science and Technology,
[♡]Ping An Technology (Shenzhen) Co., Ltd.
Correspondence: quxiaoy@gmail.com, kailu@hust.edu.cn

Abstract

在优化大语言模型（LLMs）以进行长上下文推理时，主要挑战之一在于键值（KV）缓存的高内存消耗。现有的方法，如量化，已在减少内存使用方面展示出有前景的结果。然而，目前的量化方法无法同时兼顾有效性和效率。在本文中，我们提出了 MoQAE，这是一种通过量化感知专家混合的全新混合精度量化方法。首先，我们将不同的量化位宽配置视为专家，并使用传统的专家混合（MoE）方法来选择最佳配置。为避免传统 MoE 方法中将令牌一个接一个输入路由器所导致的低效率，我们将令牌成块输入路由器。其次，我们设计了一个轻量化的仅路由器微调过程，用于训练 MoQAE，并使用全面的损失来学习模型精度和内存使用之间的权衡。最后，我们引入了一种路由冻结（RF）和路由共享（RS）机制以进一步减少推理开销。对多个基准数据集的广泛实验表明，我们的方法在效率和有效性上都优于最新的 KV 缓存量化方法。

1 介绍

近年来，大型语言模型（LLM）已经成为许多领域的基石，包括自然语言处理、计算机视觉、时间序列数据等等。随着这些模型的不断演进，处理更长和更复杂文本的需求也显著增长。一些复杂的任务常常需要能够处理跨越数千个标记的扩展上下文的模型。尽管最新的 LLM 可以处理多达 200 万个输入标记，长上下文推理在内存消耗和计算效率上仍然面临重大挑战。我们在图中绘制了 Llama2-13B 模型的内存使用构成与上下文长度的关系（超出设备内存限制的部分是我们的估算）。权重所占的内存是固定的，而键值（KV）缓存所占的内存与上下文长度成正比。当上下文长度较小时，内存使用仍然由权重主导。然而，随着上下文长度的增加，内存使用很快转向由 KV 缓存主导。最终，当上下文长度达到 128k 时，KV 缓存的内存使用可能达到 100GB，远远超出大多数商用 GPU 的内存容量。显然，在长上下文推理期间，内存使用的主要瓶颈在于 KV 缓存。此外，CPU 和 GPU 内存之间频繁传输大型 KV 缓存以进行计算更加剧了问题，导致显著的推理延迟。

研究人员提出了各种方法来优化 LLM 用于长上下文推理，包括剪枝、知识蒸馏和量化。其中，量化是最容易实现的方法，并且可以最大限度地减少内存消耗。一些研究人员提出将模型均匀量化到低位宽，这在内存减少上取得了很好的表现，但可能导致精度急剧下降。其他研究人员设计了混合精度量化，以保持高位宽的重要 tokens 来维持模型精度。然而，这些混合精度方法需要复杂且耗时的量化搜索过程来确定位宽配置。受 MoICE (Lin et al., 2024a) 的启发，它在专家模块混合专家 (MoE) 中使用专家作为旋转位置嵌入 (RoPE) 的基础，我们利用混合专家 (MoE) 方法快速训练和推理速度的优势，提出了 MoQAE，这是一种通过量化感知专家混合的新颖混合精度 KV 缓存量化方法。我们的主要创新是创造性地使用 MoE 技术来学习量化位宽配置。具体来说，我们的贡献包括三个部分。(1) 我们将每种量化比特宽度配置视为一个专家（这也是“量化感知专家”名称的由来），并利用 MoE 方法中的路由器选择最合适的量化比特宽度。即，我们将一个令牌输入到一个路由器中，路由器识别出该令牌最合适的专家。与该专家对应的量化比特宽度是我们需要量化该令牌所用的比特宽度。我们分块输入令牌，而不是在传统 MoE 方法中使用逐个令牌的方式。(2) 我们设计了一种轻量级的微调过程。我们不训练整个 LLM，而是冻结预训练的 LLM 参数，并对 MoE 路由器使用校准数据集进行微量微调。在微调期间，我们引入了一种综合损失，以平衡模型的准确性和内存使用。(3) 我们提出了一种冻结路由（RF）和共享路由（RS）机制。RF 机制冻结初始块的量化策略以保持模型的准确性，而 RS 机制允许量化策略在不同的 LLM 模块之间共享。

*Xiaoyang Qu (email: quxiaoy@gmail.com) and Kai Lu (email: kailu@hust.edu.cn) are the corresponding authors.

2 背景

2.1 预备知识

LLM 推理。现代 LLM 架构主要基于仅解码器结构，推理分为两个不同的阶段：预填充阶段和解码阶段。在预填充阶段，所有输入标记由 LLM 处理以生成第一个输出标记。随后，在解码阶段，LLM 处理包括所有输入标记和已生成标记的序列，以生成下一个输出标记。该过程重复迭代，每个新生成的标记被附加到序列中进行后续处理，直至整个输出序列完成。此方法的一个显著缺点是，在每个步骤中，与输入标记和所有先前生成标记对应的键（K）和值（V）矩阵必须重新计算，导致效率低下。为了解决这一问题，现代 LLM 使用 KV 缓存，它存储输入和生成标记的 K 和 V 矩阵，消除了冗余计算，大大降低了推理延迟。然而，当处理长输入文本时，KV 缓存的大小剧增，耗费大量 GPU 内存，使得模型在资源受限的硬件上难以部署。此外，随着 KV 缓存的大小增加，KV 缓存在 CPU 和 GPU 内存之间的频繁传输变得更加耗时，使得 KV 缓存成为推理延迟的瓶颈。

专家混合体。MoE 是一种模型架构，旨在通过一个路由机制将计算任务分配给多个专家（子模型）并动态选择子集来处理给定的输入。最近，MoE 架构被广泛应用于大型语言模型（LLM），如 Switch Transformer (Fedus et al., 2022) 和 GLaM (Du et al., 2022) 。传统上，MoE 将 LLM 中的每一层前馈网络（FFN）视为一个专家，并由路由器根据输入动态激活这些 FFN 层中的一小部分，而未激活的层则保持闲置。这种策略后来也扩展到了自注意力层 (Zhang et al., 2022) 。与密集模型相比，MoE 的稀疏激活机制在保持参数规模卓越的可扩展性的同时，显著减少了计算开销。在这项工作中，我们创新性地将 LLM 层视作专家，而不是简单地将量化比特宽度配置视作专家，并提出了量化感知性专家。

2.2 相关工作

KV 缓存优化。研究人员已提出了各种方法来优化 LLM 中的 KV 缓存。一些方法 (Zhang et al., 2023; Xiao et al., 2024; Han et al., 2024; Liu et al., 2024a; Ge et al., 2024; Pagliardini et al., 2023) 引入了修剪技术，以消除不太重要的令牌的 KV 缓存。例如，Zhang 等人提出了 H_2O (Zhang et al., 2023) ，该方法移除了在注意力权重矩阵中的垂直注意力分数总和最低的令牌。StreamingLLM (Xiao et al., 2024) 提出了一种“注意力吸收”机制，仅保留初始标记和最近的标记。其他研究人员 (Song et al., 2024; Xue et al., 2024; He and Zhai, 2024; Kwon et al., 2023; Dao et al., 2022; Yu et al., 2022; Cai et al., 2024; Jin et al., 2023) 专注于内存管理策略，从系统级别解决 KV 缓存碎片问题。例如，vLLM (Kwon et al., 2023) 构建了一个页表，将 KV 缓存的连续逻辑页映射到非连续的物理内存页，同时采用写时复制机制以减少内存使用。Jin 等人提出了 S3 (Jin et al., 2023) ，在推理过程中预测输出序列长度，并根据预测结果分配 KV 缓存内存空间，以避免因过度分配 KV 缓存空间造成的内存浪费。此外，量化 (Liu et al., 2024b; Hooper et al., 2024; Zhao et al., 2024; Frantar et al., 2023; Yang et al., 2024; Kim et al., 2024) 被探索作为一种将 KV 缓存数据从高精度格式转换为低精度格式的有前途的方法，从而节省内存。KIVI (Liu et al., 2024b) 识别出键缓存中存在许多异常通道。因此，提出对键缓存进行每通道量化，而值缓存则以标准的每标记方式进行量化。Atom (Zhao et al., 2024) 对 KV 缓存应用非对称和 4 位分组量化，并在 KV 缓存与查询向量计算前进行反量化。这些方法中，量化显得尤为有效且简单。然而，传统量化往往会导致显著的性能下降。本文提出了一种新的混合精度量化方法，实现了几乎无损的模型性能，解决了现有技术的局限，同时优化了 KV 缓存的内存使用。

混合精度量化。为缓解量化导致的性能下降，研究人员提出了混合精度量化方法 (Hooper et al., 2024; Yang et al., 2024; Zhang et al., 2024b; Kim et al., 2024; Lin et al., 2024b; Tao et al., 2025) 。这些方法为更重要的标记分配较高的比特宽度，为不太关键的标记分配较低的比特宽度，从而更有效地保持模型性能。最初，研究人员将混合精度量化应用于 LLM 的权重和激活值。例如，SqueezeLLM (Kim et al., 2024) 将 LLM 的权重分为一个稠密矩阵和一个稀疏矩阵，然后对稀疏矩阵使用 INT8 量化，同时保持稠密矩阵的精度为 FP16。AWQ (Lin et al., 2024b) 提出了一个激活感知的权重量化，通过激活值的分布找到重要权重的 1 % ，并重新排列权重以确保硬件效率。逐渐地，随着 KV 缓存问题变得愈发突出，混合精度量化也扩展到了 KV 缓存中。例如，MiKV (Yang et al., 2024) 使用与 H_2O 相同的方法来确定重要的标记，但使用较低比特量化而不是驱逐它们。KVQuant (Hooper et al., 2024) 在量化过程中保持 KV 缓存中异常值（大幅度值）的高精度，并设计了一种新的数据类型 nuqX 来表示混合精度量化后的 KV 缓存。然而，大多数这些方法需要极长的搜索时间来确定量化比特宽度。在本文中，我们通过量化感知专家提出了一种新颖的混合精度量化方法。该方法采用了 MoE 方法

中的高效路由器，能够快速有效地学习 KV 缓存的最佳量化配置。

2.3 概述

图 **??** 展示了 MoQAE 的概览。输入文本首先被划分为几个等长度的块，然后被 LLM 处理。在 LLM 的每个模块中，我们使用量化搜索模块来确定输入块的量化策略（即量化位宽配置）。随后，这些块使用刚刚确定的位宽配置进行量化，并继续进行模块中的正式计算（注意力和前馈计算）。最后，输出块被传递到下一个模块，重复这一过程。值得注意的是，我们对第一个块应用了一种路由冻结机制，防止其进入路由器并将其位宽固定为 FP16。此外，我们在模块之间采用了路由共享机制，允许不同的模块使用相同的量化策略。

在量化搜索模块中，我们引入了一个路由器和多个注意力感知专家。这些专家表示不同的量化位宽配置，例如 FP16、INT4、INT2 等等。输入文本被分成几个等长的块，对于不满足块大小的剩余部分，我们直接保持其精度为 FP16。在每个 LLM 的块内，这些块首先传递到一个路由器，其中路由器网络使用一个具有

$$\mathcal{P} = f(CW_1 \cdot CW_2)W_3 \tag{1}$$

函数的 MLP 实现。

这里，$C \in \mathbb{R}^{N\times D}$ 是输入块，$f()$ 是激活函数，$W_1, W_2 \in \mathbb{R}^{D\times M}$ 和 $W_3 \in \mathbb{R}^{D\times M}$ 是权重矩阵，其中 D 是每个注意力头中的嵌入维度大小，N 是块大小，M 是专家数量。输出 $\mathcal{P} \in \mathbb{R}^{N\times M}$ 反映了所有块关于选择哪个专家的概率。

对于块中的每个令牌，选择概率最高的专家作为该令牌的选定专家。随后，我们找出在该块中被选定次数最多的专家，并将其表示为整个块的量化策略。方程如下：

$$\mathcal{R} = \underset{1\le k\le M}{\arg\max}\left(\sum_{i=1}^{N}\mathbb{I}\left(\underset{1\le j\le M}{\arg\max}\, p_j^i = k\right)\right) \tag{2}$$

其中 $\mathcal{R} \in \{1, 2, ..., M\}$ 是量化策略，p_j^i 表示选择专家 j 用于块 i 的概率，$\mathbb{I}()$ 操作符表示如果条件满足结果为 1，否则为 0。最后，我们整合所有选定的专家，生成所有块的量化策略，并使用该量化策略对输入文本进行量化。

2.4 微调过程

为了加速训练过程，我们设计了一种高效的训练方法：冻结 LLM 本身的参数，只微调路由器的参数。此外，我们的微调是在称为校准数据集的原始数据集子集上进行的。

我们进一步在微调过程中设计了一种新颖的损失函数。该损失的目标是在长上下文推理期间实现大型语言模型的准确性与内存使用之间的权衡。该损失的设计细节如下：

一方面，为了优化模型的准确性，我们将模型的负对数似然损失 L_{nll} 作为最终损失的一部分。然而，我们不能直接应用 L_{nll}，因为它不涉及与路由器权重直接相关的操作，无法用于训练路由器的权重。因此，我们定义了一种新的损失称为 L_{model}，它是通过将 L_{nll} 乘以路由器输出的专家选择概率的平均值获得的。为了反映不同专家对模型准确性的不同重要性，我们对该乘积的每个成分应用一个惩罚项。L_{model} 最终计算如下：

$$L_{model} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\left(\underset{1\le k\le M}{\arg\max}\, p_k^i = j\right)\cdot\frac{p_j^i \cdot L_{nll}}{B_j} \tag{3}$$

，其中 p_k^i 表示为块 i 选择专家 k 的概率，$1/B_j$ 是专家 j 的惩罚项，B_j 表示专家 j 对应的比特宽度。我们选择 $1/B_j$ 作为惩罚项，因为更低比特宽度的数据导致更高的模型损失。

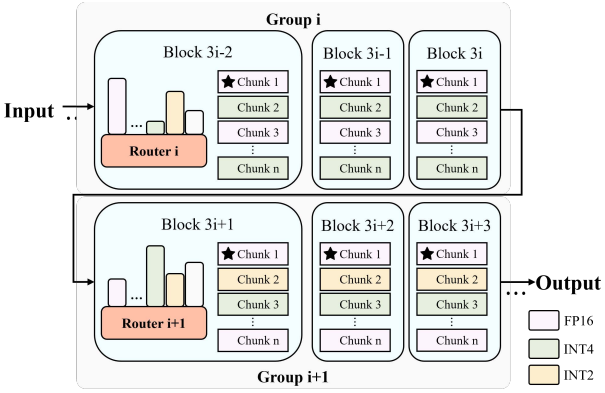

Figure 1: 路由共享机制。

另一方面，为了确保我们的方法也优化内存使用，我们引入了内存损失 L_{mem}。L_{mem} 的目的是鼓励路由器优先选择表示较低位宽的专家，从而减少模型的 GPU 内存使用。我们还计算了 L_{mem}，它是专家选择概率平均值的加权和，但惩罚项是以倒置方式应用的：在这里我们选择 $\frac{16}{B_j}$ 作为惩罚项。这是因为带有更高位宽的数据会导致更多的内存消耗。

最后，我们的损失定义如下：

$$L = \lambda L_{model} + (1-\lambda)L_{mem} \tag{4}$$

其中 λ 是一个预定义的超参数，用于控制模型准确性和内存使用之间的权衡。我们将在第 **??** 节中讨论 λ 对模型性能的影响。

2.5 路由冻结和路由共享

之前的研究者 (Xiao et al., 2024) 已经证明，LLM 初始位置的 token 在模型性能中起着关

键作用，显著影响其准确性。在我们的研究中，我们也通过实验探讨了 LLM 中不同层初始 token 的注意力权重。如图 **??** 所示，我们观察到初始位置的 token 的注意力权重相对于后续位置的 token（除了前两层）是相对较高的。这个发现强烈表明序列开头的 token 影响力很大，在决定模型输出中起着至关重要的作用。这些初始 token 似乎捕捉到了重要的上下文信息，然后这些信息传播到序列的其余部分。

针对这些观察结果，我们引入了一种路由冻结机制，以确保初始位置的关键标记在量化过程中不被损害。具体而言，我们防止第一个标记块被传递到路由器，并将其限制为选择 FP16 量化配置。这样的方法保证了序列开始处的标记以更高的精度保留，并且不会被量化为较低的位宽，从而保护模型的准确性。

此外，我们提出了一种路由共享机制，以进一步优化推理过程。我们的灵感来自 CLA (Brandon et al., 2024)，这表明在不同注意力层之间共享键和值头以减少计算开销是可行的。如图 1 所示，在这种机制中，我们将 LLM 中的不同模块分成若干组。在每组中，其他模块共享第一个模块的量化策略。其他模块中的路由器也被移除。通过路由共享机制，我们可以有效减少因路由器过多而导致的内存使用以及因大多数模块中的路由器计算而导致的延迟。虽然在不同模块间共享路由策略可能会导致模型精度的轻微损失（因为一个模块中的 KV 缓存的量化策略可能不适用于下一个模块），但这种损失不是很严重（我们将在第 **??** 节中证明）。同时，路由共享机制可以显著减少内存使用和计算延迟。因此，我们认为这种损失是可以接受的。我们还在第 **??** 节中探讨了组大小对模型性能的影响。

3 评估

3.1 实验设置

基准测试。我们对六个广泛使用的开源模型进行 MoQAE 基准测试：Llama-7B、Llama-13B (Touvron et al., 2023a) 、Llama2-7B、Llama2-13B (Touvron et al., 2023b) 、Llama3-8B (Dubey et al., 2024) 和 Mistral-7B (Jiang et al., 2023) 。为了评估性能，我们在 WikiText2 (Merity et al., 2017) 数据集上评估 MoQAE 的困惑度。我们还采用 LongBench (Bai et al., 2024) 进一步评估我们的方法和基线的长上下文生成性能。我们从 LongBench 中的四种不同任务类型中选择了八个子集作为我们的实际数据集。它们是单文档 QA 任务（Qasper）、摘要任务（QMSum、MultiNews）、少样本学习任务（TREC、TriviQA、SAMSum）和代码补全任务（LCC、RepoBench-P）。F1 分数用于评估 Qasper 和 TriviaQA，而 ROUGE 分数用于评估 QMSum 和 MultiNews，相似性分数用于评估 LCC 和 RepoBench-P。只有 TREC 使用分类分数作为评估指标。上下文的最大长度分别为 Llama 的 2048、Llama-2、Llama-3 的 4096 和 Mistral 的 8192。

基线。我们将 MoQAE 与 FP16 全精度模型以及其他九种最新的 KV 缓存量化方法进行比较，作为基线：① INT，表示统一整数量化。② NF，表示 NormalFloat 量化。③ KVQuant (Hooper et al., 2024)，保留高位宽的异常值。KVQuant-[x]b-1 % 表示 1 % 的 tokens 保持 FP16 精度。④ CQ (Zhang et al., 2024a)，将多个键/值通道结合在一起，以利用它们的相互依赖性。CQ-[x]c[y]b 表示每组有 x 个通道，并且每组量化码中有 y 位。⑤ Atom (Zhao et al., 2024)，在注意力头的粒度上使用非对称统一量化。⑥ QoQ (Lin et al., 2025)，缩放查询和键，以减少由在键缓存中量化异常值引起的损失。⑦ AWQ (Lin et al., 2024b)，对 KV 缓存应用统一 4 位量化。⑧ MiKV (Yang et al., 2024)，通过计算每个 token 的注意力分数总和，对那些低注意力分数总和的进行低位宽量化，而保持其余的在高位宽。⑨ KIVI (Liu et al., 2024b)，对键缓存使用每通道量化，并对值缓存使用每 token 量化。每个 token 的量化位宽根据其显著性分配。在这些方法中，① 、② 、④ 、⑤ 、⑥ 、⑦ 、⑨ 是统一量化；③ 、⑧ 是混合精度量化。方法名称中的后缀"gs" 表示组的大小，其他不含"gs" 的方法名称表示这些方法不使用组量化。

实现。我们在一台包含 96 GB 内存的 NVIDIA H20-NVLink GPU 上进行实验，并配备了 25 核 AMD EPYC 7T83 CPU 和 100GB 的 RAM。块大小设置为 32，λ 设置为 0.5。路由共享机制中的组大小设置为 3。路由器由一个 2 层的 MLP 组成，隐藏维度为专家数量。我们使用 SiLU 作为激活函数，并使用 top-1 专家选择作为路由机制。路由器参数的内存使用约为 1.6KB。关于训练，我们使用完整训练集的 5 % 作为校准数据集。我们使用 AdamW 作为优化器，学习率为 3e-4，批量大小为 8。

3.2 性能

我们首先在 Wikitext2 数据集上评估困惑度。结果如表 **??** 所示。我们还测试了 λ 为 0.3 的情况。从表中可以看出，简单的量化到极低的比特宽度（2 位）会导致显著的准确性损失。即使采用精心设计的量化方法，随着比特宽度的减少，模型的准确性也会迅速下降。与其他方法相比，MoQAE 能够在保持模型良好准确性的同时，将模型的平均比特宽度降低到相对较

Table 1: MoQAE 和基线方法在 LongBench 数据集上的表现，数值越高越好。

Method	Qasper ↑	QMSum ↑	MultiNews ↑	TREC ↑	TriviaQA ↑	SAMSum ↑	LCC ↑	RepoBench-P ↑
FP16	9.52	21.28	3.51	66.00	87.72	41.69	66.66	59.82
KIVI-2b ⑧	9.26	20.53	0.97	66.00	87.42	42.61	66.22	59.67
CQ-4c8b ④	9.58	20.87	1.93	66.00	87.72	41.13	66.57	59.75
MiKV ⑧	9.14	20.63	0.85	65.88	87.21	41.44	66.18	59.55
MoQAE	9.79	21.23	3.47	66.00	87.89	41.37	66.53	59.94

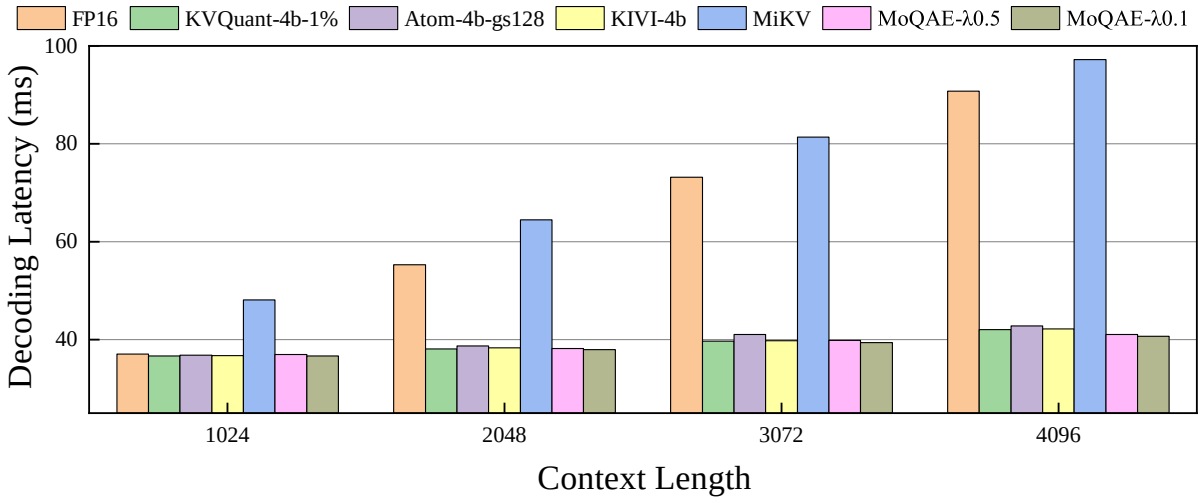

Figure 2: 不同上下文长度下，MoQAE 和基线方法的解码延迟。

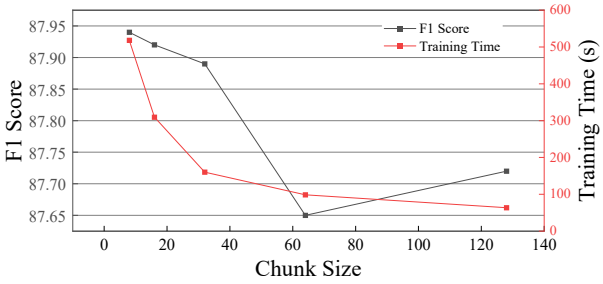

Figure 3: 块大小对模型性能和训练时间的影响。

低的水平。在 4-16 位的方法中，MoQAE- λ 0.5 实现了最低的困惑度，同时与基准方法具有相似的平均比特宽度。MoQAE- λ 0.5 的困惑度平均只比 FP16 模型多 0.08。MoQAE- λ 0.3 在大多数模型中也优于 2-4 位的方法。

我们还比较了 MoQAE 和其他 方法在 Long-

Table 2: 数据块大小对解码延迟的影响。

Chunk Size	8	16	32	64	128
Decoding Latency/ms	24.85	24.26	23.86	23.59	23.01

Bench 数据集上的表现。如表 1 所示，MoQAE 在大多数数据集上取得了最佳表现。MoQAE 在 SAMSum 和 LCC 数据集上的表现仅比基线方法稍差。

此外，我们在不同的上下文长度下对 MoQAE 和其他方法的内存占用和解码延迟进行了评估，批量大小为 8。我们在两种 λ 下测试 MoQAE。如图 **??** 和图 2 所示，MoQAE- λ 0.1 在所有上下文长度中实现了最低的内存使用和解码延迟。

与最先进的量化方法（SOTA）相比，MoQAE 平均可以减少 0.79GB 的内存使用，并减少 0.44ms 的解码延迟。MoQAE- λ 0.5 的效率不如 MoQAE- λ 0.1，但其仍然平均减少了 FP16 模型的内存使用 2.99GB，并且在解码延迟上优于大多数基准方法。

我们探讨了区块大小对模型性能的影响。结果如图 3 和表格 2 所示。随着区块大小的增加，训练时间显著减少，解码延迟也随之减少。模型准确性呈现出先下降后轻微上升的趋势。这是因为当区块大小变大时，一些重要的标记信息会被包裹在更多不重要的标记信息中。这样的区块可能被路由器错误识别为 INT2 量化，从而导致重要信息的丢失。当区块大小大时，由于我们将第一个区块固定为 FP16，更多的重要信息得以保存，这略微提高了模型的准确性。

我们进一步对超参数 λ 进行了消融实验。如

表 **??** 所示，随着 λ 的增加，模型准确性提高（当 λ 大于 0.5 时，准确性达到上限），同时平均比特数和内存使用量减少。这个结果表明，λ 可以有效平衡模型的准确性和内存使用量。

我们还测试了路由冻结和路由共享机制的影响。从表 **??** 可以看到，当从 MoQAE 中移除路由冻结时，准确性和推理延迟都有所降低。这是因为某些块的第一个块可能从原来的固定 FP16 更改为其他较低的比特宽度。当移除路由共享时，解码延迟显著改善，而准确性略微提高。这是因为在移除路由共享后，我们需要进行更多的路由计算，但计算出的比特宽度配置也会更为准确。同时，我们测试了路由共享机制中不同组大小的影响。可以看到，随着组大小的增加，解码延迟显著减少，但准确性也略有下降。

在本文中，我们介绍了 MoQAE，一种基于量化感知专家混合的新型混合精度量化方法。首先，我们将不同的量化位宽配置视为专家，并应用传统的 MoE 方法来选择最佳配置。为了避免传统 MoE 方法中逐个输入标记的低效，我们将标记分块输入到路由器中。其次，我们提出了一种轻量级的仅限路由器的微调过程，并设计了一种新的损失函数，使模型能够学习模型准确性和内存使用之间的权衡。最后，我们引入了 RS 和 RF 机制，进一步减少了由路由器引起的推理开销。在基准数据集上的大量实验表明，我们的方法在效率和效果方面均优于 SOTA 的混合精度量化技术。

因为我们的方法在 LLM 中引入了额外的路由器，这些路由器的参数将占用一些内存，并且路由器的计算也会减慢模型的推理时间。虽然我们采用了输入分块和路由共享等方法进行优化，但这些开销仍然存在。

此外，为了确保注意力计算结果的准确性，由于 softmax 在计算注意力权重时具有高精度要求，我们将量化的键向量解量化为 FP16，并与 FP16 查询向量一起进行计算。这个解量化操作也会导致额外的延迟。

该工作得到了广东省重点研发计划 (No.2021B0101400003)、国家重点研发计划 (No.2023YFB4502701)、国家自然科学基金 (No.62172175)、中国博士后科学基金 (No.2024M751011) 和湖北省博士后项目 (No.2024HBBHCXA027) 的资助。

References

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. 2024. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3119–3137.

William Brandon, Mayank Mishra, Aniruddha Nrusimha, Rameswar Panda, and Jonathan Ragan-Kelley. 2024. Reducing transformer key-value cache size with cross-layer attention. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple llm inference acceleration framework with multiple decoding heads. In *Proceedings of the 41st International Conference on Machine Learning*, pages 5209–5235.

Tri Dao, Daniel Y Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: fast and memory-efficient exact attention with io-awareness. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 16344–16359.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. 2022. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pages 5547–5569. PMLR.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. Gptq: Accurate post-training quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. 2024. Model tells you what to discard: Adaptive kv cache compression for llms. In *The Twelfth International Conference on Learning Representations*.

Chi Han, Qifan Wang, Hao Peng, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. 2024. Lm-infinite: Zero-shot extreme length generalization for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3991–4008.

Jiaao He and Jidong Zhai. 2024. Fastdecode: High-throughput gpu-efficient llm serving using heterogeneous pipelines. *arXiv preprint arXiv:2403.11421*.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun S Shao, Kurt Keutzer, and Amir Gholami. 2024. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *Advances in Neural Information Processing Systems*, 37:1270–1303.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Yunho Jin, Chun-Feng Wu, David Brooks, and Gu-Yeon Wei. 2023. S^3: Increasing gpu utilization during generative inference for higher throughput. *Advances in Neural Information Processing Systems*, 36:18015–18027.

Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W Mahoney, and Kurt Keutzer. 2024. Squeezellm: dense-and-sparse quantization. In *Proceedings of the 41st International Conference on Machine Learning*, pages 23901–23923.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.

Hongzhan Lin, Ang Lv, Yang Song, Hengshu Zhu, Rui Yan, et al. 2024a. Mixture of in-context experts enhance llms' long context awareness. *Advances in Neural Information Processing Systems*, 37:79573–79596.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024b. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems*, 6:87–100.

Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. 2025. Qserve: W4a8kv4 quantization and system co-design for efficient llm serving. In *Proceedings of Machine Learning and Systems*.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. 2024a. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems*, 36.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. 2024b. Kivi: a tuning-free asymmetric 2bit quantization for kv cache. In *Proceedings of the 41st International Conference on Machine Learning*, pages 32332–32344.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *International Conference on Learning Representations*.

Matteo Pagliardini, Daniele Paliotta, Martin Jaggi, and François Fleuret. 2023. Faster causal attention over large sequences through sparse flash attention. *arXiv preprint arXiv:2306.01160*.

Yixin Song, Zeyu Mi, Haotong Xie, and Haibo Chen. 2024. Powerinfer: Fast large language model serving with a consumer-grade gpu. In *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles*, pages 590–606.

Wei Tao, Bin Zhang, Xiaoyang Qu, Jiguang Wan, and Jianzong Wang. 2025. Cocktail: Chunk-adaptive mixed-precision quantization for long-context llm inference. In *2025 Design, Automation & Test in Europe Conference (DATE)*, pages 1–7. IEEE.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*.

Zhenliang Xue, Yixin Song, Zeyu Mi, Le Chen, Yubin Xia, and Haibo Chen. 2024. Powerinfer-2: Fast large language model inference on a smartphone. *arXiv preprint arXiv:2406.06282*.

June Yong Yang, Byeongwook Kim, Jeongin Bae, Beomseok Kwon, Gunho Park, Eunho Yang, Se Jung Kwon, and Dongsoo Lee. 2024. No token left behind: Reliable kv cache compression via importance-aware mixed precision quantization. *arXiv preprint arXiv:2402.18096*.

Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. 2022. Orca: A distributed serving system for {Transformer-Based} generative models. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pages 521–538.

Tianyi Zhang, Jonah Yi, Zhaozhuo Xu, and Anshumali Shrivastava. 2024a. Kv cache is 1 bit per channel: Efficient large language model inference with coupled quantization. *Advances in Neural Information Processing Systems*, 37:3304–3331.

Xiaofeng Zhang, Yikang Shen, Zeyu Huang, Jie Zhou, Wenge Rong, and Zhang Xiong. 2022. Mixture of attention heads: Selecting attention heads per token. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4150–4162.

Zhenyu Zhang, Shiwei Liu, Runjin Chen, Bhavya Kailkhura, Beidi Chen, and Atlas Wang. 2024b. Q-hitter: A better token oracle for efficient llm inference via sparse-quantized kv cache. *Proceedings of Machine Learning and Systems*, 6:381–394.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. 2023. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710.

Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. 2024. Atom: Low-bit quantization for efficient and accurate llm serving. *Proceedings of Machine Learning and Systems*, 6:196–209.