PR #22844 完整报告

sgl-project/sglang

[AMD] Optimize _append_shared_to_topk_output by a single fused Triton kernel for Qwen3.5

合并时间：2026-04-15 14:50

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/22844

执行摘要

- 一句话：为 AMD 平台 Qwen3.5 MoE 模型优化共享专家追加逻辑，用单个 Triton 内核融合 4 次内核启动以提升路由性能。
- 推荐动作：该 PR 值得精读，特别是对于关注高性能计算和 MoE 模型优化的工程师。重点关注新增的 Triton 内核设计，它展示了如何将多个独立操作融合为单次启动以减少开销，同时保留逐 token 权重的精度要求。此外，注意其平台特定性（AMD/AITER），这反映了项目中对不同硬件后端的差异化优化策略。

功能与动机

PR body 明确指出，_append_shared_to_topk_output 函数位于 MoE 路由的关键路径上，原先使用 4 次独立内核启动（2 次逐元素操作 +2 次拼接）来追加共享专家 ID 和权重，带来了不必要的启动开销。Qwen3.5 模型需要从学习的 shared_expert_gate 中获取逐 token 的共享权重，因此优化必须保留这种逐 token 的权重精度。

实现拆解

1. 新增融合内核函数：在 python/sglang/srt/layers/moe/fused_moe_triton/fused_moe_triton_kernels.py 中，新增 Triton JIT 内核 _fused_append_shared_experts_with_weights_kernel 及其包装函数 fused_append_shared_experts_with_weights。该内核在一个启动中完成：复制原始 topk_ids 和 topk_weights，追加共享专家 ID（基于 N_BASE 计算），并加载和追加逐 token 的共享权重。包装函数负责处理输入张量的形状转换（如将 1D 权重扩展为 2D）、计算合适的块大小，并调用内核。
2. 更新模型调用点：修改 python/sglang/srt/models/qwen2_moe.py 中的 _append_shared_to_topk_output 方法，移除原先手动创建共享 ID 张量、扩展权重并拼接的逻辑，改为导入并调用新的 fused_append_shared_experts_with_weights 函数。这简化了代码，并将计算委托给优化后的内核。
3. 测试与验证配套：PR body 提供了详细的准确性测试（GSM8K 准确率保持在 ~0.95）和性能基准测试结果（在不同并发度下，吞吐量提升 1.93%-4.16%，延迟指标改善 1.81%-3.99%），表明优化有效且无回归。本次改动未包含直接的单元测试文件变更，但依赖现有的模型测试和 CI 流程进行验证。

关键文件：

- python/sglang/srt/layers/moe/fused_moe_triton/fused_moe_triton_kernels.py（模块 MoE 层；类别 source；类型 core-logic；符号 _fused_append_shared_experts_with_weights_kernel, fused_append_shared_experts_with_weights）：新增了融合 Triton 内核及其包装函数，是性能优化的核心实现。
- python/sglang/srt/models/qwen2_moe.py（模块 模型层；类别 source；类型 data-contract；符号 _append_shared_to_topk_output）：修改了 MoE 模型中的关键路由方法，以使用新的融合内核，是优化的调用入口。

关键符号：_fused_append_shared_experts_with_weights_kernel, fused_append_shared_experts_with_weights, _append_shared_to_topk_output

关键源码片段

python/sglang/srt/layers/moe/fused_moe_triton/fused_moe_triton_kernels.py

新增了融合 Triton 内核及其包装函数，是性能优化的核心实现。

```
@triton.jit
def _fused_append_shared_experts_with_weights_kernel(
    topk_ids_ptr,
    topk_weights_ptr,
    shared_weights_ptr,
    out_ids_ptr,
    out_weights_ptr,
    N_BASE,
    K: tl.constexpr,
    S: tl.constexpr,
    BLOCK_K: tl.constexpr,
    BLOCK_S: tl.constexpr,
):
    pid = tl.program_id(0) # 每个程序ID处理一个token行
    ids_row_ptr = pid * K
    out_row_ptr = pid * (K + S)

    # 加载并存储原始的topk IDs和权重
    offs_k = tl.arange(0, BLOCK_K)
    mask_k = offs_k < K
    ids = tl.load(topk_ids_ptr + ids_row_ptr + offs_k, mask=mask_k)
    ws = tl.load(topk_weights_ptr + ids_row_ptr + offs_k, mask=mask_k)
    tl.store(out_ids_ptr + out_row_ptr + offs_k, ids, mask=mask_k)
    tl.store(out_weights_ptr + out_row_ptr + offs_k, ws, mask=mask_k)

    # 加载并存储共享专家的IDs和权重
    offs_s = tl.arange(0, BLOCK_S)
    mask_s = offs_s < S
    shared_ids = tl.cast(N_BASE + offs_s, ids.dtype) # 共享专家ID基于N_BASE偏移计算
    shared_ws = tl.load(shared_weights_ptr + pid * S + offs_s, mask=mask_s) # 逐token加载权重
    tl.store(out_ids_ptr + out_row_ptr + K + offs_s, shared_ids, mask=mask_s)
```

```
    tl.store(out_weights_ptr + out_row_ptr + K + offs_s, shared_ws, mask=mask_s)
```

python/sglang/srt/models/qwen2_moe.py

修改了 MoE 模型中的关键路由方法，以使用新的融合内核，是优化的调用入口。

```
def _append_shared_to_topk_output(
    self,
    topk_output: StandardTopKOutput,
    hidden_states: torch.Tensor,
) -> StandardTopKOutput:
    """Append shared expert ids and weights to topk output before fused MoE."""
    if not self.enable_shared_expert_fusion:
        return topk_output
    shared_weights = self._get_shared_expert_weights(hidden_states)
    if shared_weights is None:
        return topk_output

    # 导入并使用新的融合函数，替代原先的多步操作
    from sglang.srt.layers.moe.fused_moe_triton.fused_moe_triton_kernels import (
        fused_append_shared_experts_with_weights,
    )

    fused_topk_ids, fused_topk_weights = fused_append_shared_experts_with_weights(
        topk_output.topk_ids,
        topk_output.topk_weights,
        shared_weights,
        self.num_fused_shared_experts,
        N=self.num_experts, # 传递共享专家的基础ID
    )
    return StandardTopKOutput(
        topk_weights=fused_topk_weights,
        topk_ids=fused_topk_ids,
        router_logits=topk_output.router_logits,
    )
```

评论区精华

Review 评论非常简短，两位审核者（kkHuang-amd 和 HaiShaw）均快速批准。HaiShaw 的评论“LGTM, only AMD/AITER path relevant.”点明了本次优化的适用范围仅限于 AMD/AITER 路径，暗示了平台特定性，但未引发深入的技术讨论或争议。

- 优化适用范围确认 (design): 无争议，优化被认可为针对特定平台的有效改进。

风险与影响

- 风险：1. 平台兼容性风险：优化针对 AMD 平台（AITER 后端），可能未在其他硬件（如 NVIDIA）上充分测试，存在平台特定回归风险。 2. 数值精度风险：新内核涉及数据类型转换（如 shared_weights.to(topk_weights.dtype)）和扩展操作，虽然基准测试显示准确性无变化，但极端输入下可能引入细微数值差异。 3. 内核健壮性风险：新增的 Triton 内核使

用动态块大小（triton.next_power_of_2），对于非标准形状输入（如 k 或 s 非常大）可能影响性能或正确性。 4. 依赖变更风险：qwen2_moe.py 新增了从 fused_moe_triton_kernels 模块的导入，增加了模块间耦合，若内核函数接口未来变更，可能破坏模型代码。

- 影响：1. 性能影响：直接提升 Qwen3.5 MoE 模型在 AMD 平台上的推理性能，减少内核启动开销，在高并发场景下吞吐量提升可达 4.16%，降低端到端延迟。 2. 代码影响：简化了 _append_shared_to_topk_output 的实现逻辑，用单个函数调用替代多步张量操作，提高可读性和维护性。 3. 用户影响：最终用户（使用 AMD 硬件运行 Qwen3.5）将体验到更快的推理速度，无需更改任何配置或代码。 4. 团队影响：为 AMD 平台优化树立了范例，可能鼓励类似针对关键路径的融合内核优化。
- 风险标记：平台特定优化，内核健壮性依赖，缺少单元测试

关联脉络

- PR #20736 [AMD] Enable share expert fusion with router experts for Qwen3.5 BF16 & FP8: 同属 AMD 平台 Qwen3.5 MoE 模型优化系列，该 PR 启用了共享专家融合，而本次 PR 进一步优化了融合路径中的内核启动效率。
- PR #22725 [Misc] Use cache_once for is_arch_support_pdl in sgl-kernel: 同为性能优化类 PR，涉及内核层优化，但针对不同模块（sgl-kernel vs. MoE）。
- PR #22606 [serving] replace O(n²) stream_buffer string concat with integer offset: 同为性能优化 PR，通过减少操作复杂度（O(n²) 到 O(1)）提升性能，与本次内核融合减少启动开销的思路类似。