PR #25463 完整报告

sgl-project/sglang

[ROCm] Eliminate redundant contiguous copy in MLA attention on ROCm MXFP4

合并时间：2026-05-29 16:28

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25463

执行摘要

- 一句话：消除 MLA 注意力 MXFP4 路径冗余拷贝
- 推荐动作：建议合入。该 PR 是一个典型的零开销布局优化范例，通过改变分配顺序消除隐式拷贝，代码改动量小、收益明确、风险低。值得关注的是其利用 aiter 内核 stride 参数支持非连续输出的特性，可在类似场景复用。

功能与动机

在 ROCm MXFP4 的 MLA 解码路径上，batched_gemm_afp4wfp4_pre_quant 输出布局为 (heads, batch, v_head_dim)，后续的 .transpose(0,1).flatten(1,2) 因 transpose 导致非连续，flatten 触发隐式 contiguous() 拷贝，profile 显示为 direct_copy_kernel_cuda（elementwise_kernel_manual_unroll），每层 ~4μs，61 层 MoE 累加 ~268μs。

实现拆解

1. 预分配目标布局缓冲区：在 forward_absorb_core 函数的 _is_hip 分支下，当 _use_aiter_gfx95 and self.w_vc.dtype == torch.uint8 时，预分配形状为 (batch, heads, v_head_dim) 的连续张量 _bmm_buf，然后取其 .transpose(0,1) 视图作为 attn_bmm_output 传入 GEMM 内核。aiter 内核 (batched_gemm_a16wfp4) 通过显式 stride 参数支持非连续输出，因此能正确写入视图。
2. 零开销 flatten：GEMM 完成后，_bmm_buf 已是连续的 (batch, heads, v_head_dim) 布局，.flatten(1,2) 只是视图变换，不产生拷贝。
3. 针对不同 o_proj 量化类型的处理：后续的 o_proj 量化操作（MXFP4/fp8/bf16）直接使用 _bmm_buf，避免不必要的 transpose。
4. 添加 else 分支：对于非 MXFP4 路径，设置 _bmm_buf = None，保持原有逻辑不变。
5. 仅影响 MXFP4 路径：变更严格限定在 _use_aiter_gfx95 and self.w_vc.dtype == torch.uint8 条件内，对其他设备和数据格式无影响。

关键文件：

- python/sglang/srt/models/deepseek_common/attention_forward_methods/forward_mla.py（模块 注意力层；类别 source；类型 core-logic）：核心变更文件，修改了 MLA 注意力中 MXFP4 路径的 GEMM 输出缓冲分配策略，消除了冗余拷贝。

关键符号：forward_absorb_core

关键源码片段

python/sglang/srt/models/deepseek_common/attention_forward_methods/forward_mla.py

核心变更文件，修改了 MLA 注意力中 MXFP4 路径的 GEMM 输出缓冲分配策略，消除了冗余拷贝。

```
# python/sglang/srt/models/deepseek_common/attention_forward_methods/forward_mla.py
# 位于 forward_absorb_core 函数中，ROCm MXFP4 分支
if _use_aiter_gfx95 and self.w_vc.dtype == torch.uint8:
    x = attn_output.transpose(0, 1) # 输入形状 (heads, batch, kv_lora_rank)
    B_heads, M_batch = x.shape[0], x.shape[1]
    N_vdim = self.w_vc.shape[2]
    # 分配 (batch, heads, dim) 布局，使后续 flatten 为视图操作，不产生拷贝
    _bmm_buf = torch.empty(
        M_batch, B_heads, N_vdim,
        device=x.device, dtype=torch.bfloat16,
    )
    # 将 (heads, batch, dim) 视图传入 aiter 内核（内核通过 stride 支持非连续输出）
    attn_bmm_output = _bmm_buf.transpose(0, 1)
    batched_gemm_afp4wfp4_pre_quant(
        x,
        self.w_vc.transpose(-2, -1),
        self.w_scale_v.transpose(-2, -1),
        torch.bfloat16,
        attn_bmm_output,
    )
else:
    _bmm_buf = None # 非 MXFP4 路径保持原样
    # ... 其他分支不变 ...

# 后续处理：根据 o_proj 量化类型直接使用 _bmm_buf
if _bmm_buf is not None:
    # _bmm_buf 已经是 (batch, heads, dim) 连续，flatten 是零开销视图
    if self.o_proj.weight.dtype == torch.uint8:
        attn_bmm_output = fused_flatten_mxfp4_quant(_bmm_buf)
    elif self.o_proj.weight.dtype == torch.float8_e4m3fn:
        attn_bmm_output = fused_flatten_fp8_group_quant(
            _bmm_buf, group_size=128, dtype_quant=torch.float8_e4m3fn
        )
    else:
        attn_bmm_output = _bmm_buf.flatten(1, 2) # 视图，无拷贝
elif self.o_proj.weight.dtype == torch.uint8:
    # 原逻辑，保持向后兼容
    attn_bmm_output = attn_bmm_output.transpose(0, 1)
    attn_bmm_output = fused_flatten_mxfp4_quant(attn_bmm_output)
elif self.o_proj.weight.dtype == torch.float8_e4m3fn:
    attn_bmm_output = attn_bmm_output.transpose(0, 1)
```

```
        attn_bmm_output = fused_flatten_fp8_group_quant(
            attn_bmm_output, group_size=128, dtype_quant=torch.float8_e4m3fn
        )
    else:
        attn_bmm_output = attn_bmm_output.transpose(0, 1).flatten(1, 2)
```

评论区精华

PR 获得 3 位 reviewer 批准（ch-wan、1am9trash、HaiShaw），无 review 评论讨论。CI 状态中 amd-bot 报告了 3 个 AMD 失败（likely unrelated）和 13 个其他失败（0 likely related）。

- 暂无高价值评论线程

风险与影响

- 风险：低风险：变更范围极小，仅修改 forward_mla.py 中约 20 行代码，且仅在 _use_aiter_gfx95 and self.w_vc.dtype == torch.uint8 条件下生效；其他路径完全不变。GSM8K 准确率通过（0.932 ⩾ 0.90），无精度回归。aiter 内核的 stride 参数契约在 ROCm 生态中相对稳定，但若 aiter 后续更新改变了 stride 语义，可能导致写错位置，建议在 aiter 版本更新后回归该路径。
- 影响：影响范围：仅影响 MI355X (gfx950) 上使用 MXFP4 (Kimi-K2.6-MXFP4) 的 MLA 注意力解码路径。性能提升约 1-3% 端到端吞吐量（低并发时更明显，高并发时因其他瓶颈掩盖），TPOT 降低约 1-3%。无功能影响，无兼容性问题。
- 风险标记：核心路径变更，依赖于外部库 (aiter) 的 stride 契约

关联脉络

- PR #26672 [AMD] Work around HIP TPOT regression from Event.wait() in MTP seq lens resolution: 同为 AMD 平台性能优化，修改了 overlap_utils.py 中的同步逻辑，与本 PR 在 AMD 生态优化上是互补的。
- PR #24133 [NPU]: Optimize xgrammar token bitmask on NPU with AscendC: 同为针对特定硬件的性能优化，使用自定义内核消除 CPU fallback，与本 PR 思路相似。