PR #33166 完整报告

sgl-project/sglang

[AMD] DeepSeek-V4 MI355X: eliminate bpreshuffle fp8-scale copies at producer sites (MoE down, MLA o_proj bmm)

合并时间：2026-08-21 12:46

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33166

执行摘要

- 一句话：MI355X 上 DSV4 fp8 scale 逐层拷贝改零拷贝视图
- 推荐动作：值得精读。它展示了硬件专用性能优化如何在保持正确性的前提下安全落地：一是 zero-copy stride 重解释的布局契约设计（as_strided 交换 stride 恢复逻辑索引），二是 M>=2 发射 gate 对单 token 退化场景的处理，三是 review 中 " 测试必须命中真实 producer kernel 而非伪造布局 " 的验证方法论。对于后续在 gfx95 上扩展更多预量化站点或复用 fp8_utils helper 的工程师，本 PR 是很好的范本。

功能与动机

PR body 明确说明：若干 DeepSeek-V4 站点会预量化激活值并把 (fp8, scale) 元组交给下游 Linear；这些 scale 以行主序发射，随后用 materialize_bpreshuffle_fp8_scale 重新布局——在 MI355X（gfx950）上是每站点、每层一次的重排拷贝。本 PR 的目标是消除这些拷贝：让支持 transpose_scale 的 AITER 量化 kernel 直接发射 bpreshuffle 所需布局，再用 torch.as_strided 零拷贝视图替代。同时由于 fused_rms_fp8_group_quant 曾有不 honor transpose_scale 的前科（#31727 已修复），PR 用 producer 级 bit-exact 测试作为正确性保证，而不只依赖 GSM8K 数值。

实现拆解

1. 收敛 producer 中立的零拷贝契约：在 python/sglang/srt/layers/quantization/fp8_utils.py 中，将 main 上 #31727 引入的 view_aiter_fused_rms_transposed_fp8_scale 的文档契约升级为 producer 中立（描述 transpose_scale=True 的物理列主字节序与 stride 交换语义），并新增 view_aiter_fused_rms_transposed_fp8_scale_tuple（仅重解释 (q_input, x_scale, ...) 元组的 scale 槽位，其余按 identity 透传）与 emit_transposed_bpreshuffle_scale(m, on_bpreshuffle_gfx95=...)（统一 M>=2 且 gfx95 的发射 gate）。这为所有 producer site 提供了单一决策与单一视图入口。
2. MoE down-proj 输入 producer 改造：在 python/sglang/srt/models/deepseek_v2.py 的 MoE forward 中，fused_clamp_act_mul 调用点先通过 emit_transposed_bpreshuffle_scale 计算 _emit_bpre，以其作为 transpose_scale 参数；为真时用 view_aiter_fused_rms_transposed_fp8_scale(x_scale) 零拷贝替换原 materialize_bpreshuffle_fp8_scale(x_scale)，为假时维持 materialize 回退，消除每层 MoE 输入 scale 的重排拷贝。

3. MLA o_proj bmm producer 改造：在 forward_mla_rocm.py 的 rocm_absorb_v_bmm 中对两处 fused_flatten_fp8_group_quant 调用点做同等改造（transpose_scale=_emit_bpre + view_aiter_fused_rms_transposed_fp8_scale_tuple / materialize_bpreshuffle_fp8_scale_tuple 二选一）。此改动经历了路径迁移：上游将 ROCm MLA dispatch 拆分到 forward_mla_rocm.py 后，原在 forward_mla.py 的改动不再影响 MI355X 调用路径，作者将其搬到实际路径并确认 forward_mla.py 与 upstream 一致。
4. 测试与验证配套：CPU 侧在 test_fp8_bpreshuffle_scale.py 新增 producer no-copy 单测（非 2D scale 直通、tuple 助手仅重解释 scale 槽位）与 TestEmitTransposedBpreshuffleScaleGate（M>=2 边界、非 gfx95 恒 false）；GPU 侧新增 test_fp8_bpreshuffle_producer_mi35x.py，在真实 gfx95 内核上驱动两个真实 producer，断言两条路径量化输出 bit 一致、scale 值相等、(1, M) 列主 stride 与零拷贝存储共享，覆盖 M∈{1,2,8,16}。PR 还给出 GSM8K 双次运行与 TPOT 初步数据，但性能提升在噪声范围内，作者未将其作为确认收益。

关键文件：

- python/sglang/srt/layers/quantization/fp8_utils.py（模块 量化工具；类别 source；类型 core-logic；符号 view_aiter_fused_rms_transposed_fp8_scale, view_aiter_fused_rms_transposed_fp8_scale_tuple, emit_transposed_bpreshuffle_scale）：核心变更文件：新增 producer 中立的零拷贝 scale 视图 helper（含 tuple 变体）与统一的 M>=2 转置发射 gate，是所有 producer site 收敛的公共入口。
- python/sglang/srt/models/deepseek_common/attention_forward_methods/forward_mla_rocm.py（模块 MLA 前向；类别 source；类型 data-contract；符号 rocm_absorb_v_bmm）：ROCm MLA 的实际 MI355X 调用路径：rocm_absorb_v_bmm 内两处 fused_flatten_fp8_group_quant o_proj producer 应用零拷贝 scale 路径，是 review 中发起路径迁移修正的关键文件。
- python/sglang/srt/models/deepseek_v2.py（模块 MoE 层；类别 source；类型 data-contract；符号 forward）：MoE down-proj 输入 producer（fused_clamp_act_mul）的调用站点改造，将每层 MoE 输入 scale 的 relayout 拷贝替换为可选的零拷贝路径。
- test/registered/unit/layers/test_fp8_bpreshuffle_producer_mi35x.py（模块 测试；类别 test；类型 test-coverage；符号 TestBpreshuffleProducerScaleNoCopy, setUpClass, setUp, _run_fused_clamp_act_mul）：新增的 MI35X GPU 测试：直接调用两个真实 AITER producer（fused_clamp_act_mul、fused_flatten_fp8_group_quant），断言 transpose_scale=True 零拷贝路径与 transpose_scale=False 物化路径 bit 等价，是 PR 正确性论证的核心证据。
- test/registered/unit/layers/test_fp8_bpreshuffle_scale.py（模块 测试；类别 test；类型 test-coverage；符号 TestBpreshuffleScaleProducerNoCopy, test_nocopy_passthrough_for_non_2d_scale, test_tuple_helper_reinterprets_only_the_scale_slot, TestEmitTransposedBpreshuffleScaleGate）：CPU 侧单测：补充 producer no-copy 行为（非 2D scale 直通、tuple 助手仅重解释 scale 槽位）与 M>=2 发射 gate 的边界测试，固定布局契约。

关键符号：view_aiter_fused_rms_transposed_fp8_scale, view_aiter_fused_rms_transposed_fp8_scale_tuple, emit_transposed_bpreshuffle_scale, rocm_absorb_v_bmm, test_fused_clamp_act_mul_producer_paths_equivalent, test_fused_flatten_fp8_group_quant_producer_paths_equivalent

关键源码片段

python/sglang/srt/layers/quantization/fp8_utils.py

核心变更文件：新增 producer 中立的零拷贝 scale 视图 helper（含 tuple 变体）与统一的 M>=2 转置发射 gate，是所有 producer site 收敛的公共入口。

```
def view_aiter_fused_rms_transposed_fp8_scale(scale: torch.Tensor) -> torch.Tensor:
    """Zero-copy view of a ``transpose_scale=True`` fp8 group scale.

    与 ``materialize_bpreshuffle_fp8_scale`` 对应的零拷贝路径：当 AITER 量化
    kernel 以 ``transpose_scale=True`` 发射 scale 时，物理字节序已是
    ``[num_groups, tokens]`` 列主布局，只是被包装成了 ``[tokens, num_groups]``
    行主视图；交换 stride 即可恢复逻辑 ``[M, G]`` 索引，全程不复制。
    """
    if scale.dim() != 2:
        return scale # 非 2D scale（如 per-tensor）不做重解释，直接透传
    # as_strided 用 (1, M) stride 覆盖 [M, G] 形状：第 0 维步长 1、第 1 维步长
    # M，正好把物理列主字节映射回逻辑行主索引，即 bpreshuffle GEMM 的消费布局。
    return torch.as_strided(scale, scale.shape, (1, scale.shape[0]))


def view_aiter_fused_rms_transposed_fp8_scale_tuple(
    value: Tuple[torch.Tensor, ...],
) -> Tuple[torch.Tensor, ...]:
    """零拷贝重解释 FP8 ``(q_input, x_scale, ...)`` 元组中的 scale 槽位。"""
    # 仅处理第 1 个槽位（scale），其余元素按原对象透传，保持引用语义不变。
    return (value[0], view_aiter_fused_rms_transposed_fp8_scale(value[1]), *value[2:])


def emit_transposed_bpreshuffle_scale(m: int, *, on_bpreshuffle_gfx95: bool) -> bool:
    """统一决策 producer 是否直接发射转置（列主）布局的 fp8 scale。

    返回 True 时 producer 以 ``transpose_scale=True`` 发射并配合上面的零拷贝
    视图；返回 False 时保持行主发射 + ``materialize`` 拷贝路径。仅在 gfx95
    bpreshuffle 且 M >= 2 时走零拷贝：M == 1 时 ``[1, G]`` 与 ``[G, 1]`` 字节序
    重合，转置发射没有收益，统一回退 materialize。
    """
    return on_bpreshuffle_gfx95 and m >= 2
```

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

ROCm MLA 的实际 MI355X 调用路径：rocm_absorb_v_bmm 内两处 fused_flatten_fp8_group_quant o_proj producer 应用零拷贝 scale 路径，是 review 中发起

路径迁移修正的关键文件。

```
elif _is_block_scale_fp8(attn.o_proj):
    # 统一 gate：gfx95 bpreshuffle 且 M >= 2 时让 producer 直接发射转置
    # scale，否则保持行主 + materialize 拷贝（M == 1 或非 gfx95 回退）。
    _emit_bpre = emit_transposed_bpreshuffle_scale(
        _bmm_buf.shape[0], # token 数 M
        on_bpreshuffle_gfx95=_use_aiter_bpreshuffle_gfx95,
    )
    attn_bmm_output = fused_flatten_fp8_group_quant(
        _bmm_buf,
        group_size=128,
        dtype_quant=torch.float8_e4m3fn,
        transpose_scale=_emit_bpre,
    )
    if _emit_bpre:
        # 零拷贝路径：对元组的 scale 槽位做 stride 交换，不复制字节。
        attn_bmm_output = view_aiter_fused_rms_transposed_fp8_scale_tuple(
            attn_bmm_output
        )
    elif _use_aiter_bpreshuffle_gfx95:
        # 行主发射的回退：仍需要 relayout 才能被 bpreshuffle GEMM 消费，
        # 但只在 gfx95 上才需要这一额外拷贝步骤。
        attn_bmm_output = materialize_bpreshuffle_fp8_scale_tuple(
            attn_bmm_output
        )
```

test/registered/unit/layers/test_fp8_bpreshuffle_producer_mi35x.py

新增的 MI35X GPU 测试：直接调用两个真实 AITER producer（fused_clamp_act_mul、fused_flatten_fp8_group_quant），断言 transpose_scale=True 零拷贝路径与 transpose_scale=False 物化路径 bit 等价，是 PR 正确性论证的核心证据。

```
@classmethod
def setUpClass(cls):
    # 设计要点：测试直接导入真实 AITER producer，而不是用模拟布局，
    # 这样才能发现内核忽略或误实现 transpose_scale 的回归。
    try:
        from aiter import dtypes # noqa: F401
        from aiter.ops.triton.fused_fp8_quant import ( # noqa: F401
            fused_flatten_fp8_group_quant,
        )
        from aiter.ops.triton.fusions.fused_clamp_act_mul import ( # noqa: F401
            fused_clamp_act_mul,
        )
    except Exception as err: # 环境缺 aiter 时跳过，避免在非 gfx95 上误报
        raise unittest.SkipTest(f"aiter producers unavailable: {err}")
    cls.device = "cuda" # torch 的 cuda 设备在 ROCm 上映射为 HIP 设备

def setUp(self):
```

```
        torch.manual_seed(0) # 固定种子，保证两条路径输入完全一致

    def _run_fused_clamp_act_mul(self, m, transpose_scale):
        # MoE down-proj 输入 producer：gate_up 先做 SiLU 门控激活再 fp8 量化。
        inter = 4 * _GROUP_SIZE # 中间维度 = 4 组，G = 4
        gate_up = torch.randn(m, 2 * inter, device=self.device, dtype=torch.bfloat16)
        q, scale = fused_clamp_act_mul(
            gate_up,
            swiglu_limit=7.0,
            activation="silu",
            dtype_quant=dtypes.fp8,
            transpose_scale=transpose_scale,
        )
        return q, scale, gate_up

    def _run_fused_flatten_fp8_group_quant(self, m, transpose_scale):
        # MLA o_proj 的 producer：bmm 缓冲先量化为 fp8 再交给下游 GEMM。
        heads, dim = 8, _GROUP_SIZE # heads * dim = 1024 -> G = 8 组
        buf = torch.randn(m, heads, dim, device=self.device, dtype=torch.bfloat16)
        out = fused_flatten_fp8_group_quant(
            buf,
            group_size=_GROUP_SIZE,
            dtype_quant=torch.float8_e4m3fn,
            transpose_scale=transpose_scale,
        )
        return out[0], out[1], buf
```

评论区精华

Review 的核心交锋集中在 " 零拷贝布局优化必须证明真实内核等价 " 这一评审标准上：

1. kkHuang-amd 首轮指出 CPU 测试用 _simulate_transpose_scale_emit 构造的是 " 伪造布局 "，无法发现真实 producer 忽略或错误实现 transpose_scale（并提醒 fused_rms_fp8_group_quant 曾有此缺陷），要求补真实 producer 测试；作者新增 test_fp8_bpreshuffle_producer_mi35x.py 直接驱动 fused_clamp_act_mul 与 fused_flatten_fp8_group_quant。
2. 关于 helper 命名，作者原计划引入新的 producer 中立名称，后按 review 合并到 main 上 #31727 已有的 view_aiter_fused_rms_transposed_fp8_scale，并补充 _tuple 变体，删除重复的 bpreshuffle_fp8_scale_nocopy。
3. kkHuang-amd 提出两个 blocker：ROCm MLA 已 dispatch 到 forward_mla_rocm.py，改 forward_mla.py 不再影响 MI355X 路径；以及 M == 1 时 contiguous [1, G] tensor 的 stride 断言不正确。作者分别以迁移优化到 rocm_absorb_v_bmm 与修正 M == 1 断言（(G, 1) stride、materialize no-op 语义）回应。
4. 合并前 reviewer 要求 rebase 到最新 main 并重跑 CPU 与 MI35X 测试；1am9trash 确认 base-a-test-cpu 与 stage-b-test-1-gpu-small-amd-mi35x-rocm720 通过，并注明 MI300 的 test_fp32_lm_head.py 与 Qwen disaggregation 失败与本 PR 无关。

- 测试未覆盖真实 producer kernel (testing): 作者新增 test_fp8_bpreshuffle_producer_mi35x.py，在真实 gfx95 内核上验证 transpose_scale=True + 零拷贝 与 transpose_scale=False + materialize 两条路径 bit 等价。
- ROCm MLA dispatch 路径迁移 (correctness): 作者将 o_proj producer 优化移到 forward_mla_rocm.py 的 rocm_absorb_v_bmm，并确认 forward_mla.py 与 upstream 一致。
- M == 1 stride 断言正确性 (correctness): 作者修正断言，明确 M == 1 走 materialize fallback 且保持 (G, 1) stride、共享存储。
- helper 命名与合并 (design): 统一为单一 helper，三个 producer 站点（fused-RMS、MoE down、MLA o_proj）共用同一 API。
- 合并前 rebase 与 CI 刷新 (question): 作者刷新分支，1am9trash 确认 base-a-test-cpu 与 stage-b-test-1-gpu-small-amd-mi35x-rocm720 通过后合并。

风险与影响

- 风险：正确性风险：整个优化依赖 AITER producer 真正实现 transpose_scale=True 的 " 物理列主字节序 " 语义，任何实现偏差都会让零拷贝路径静默产出错误布局；测试仅覆盖当前 aiter 版本与 gfx95，跨版本内核行为变化无法由本 PR 保证。M==1 特判风险：emit_transposed_bpreshuffle_scale 保证生产路径在单 token 时走 materialize（(G, 1) stride 的共享存储 no-op），但直接调用 helper 且 m==1 会得到 (1, M) stride 视图，契约文档已声明仅 M>=2 有效。性能声明强度：TPOT 数据为单次运行（Δ ≈ ±0.4–0.8%），在 decode 噪声范围内，不能作为确认的加速结论。回归面：仅影响 gfx95 + _use_aiter_bpreshuffle_gfx95 的 DeepSeek-V4 路径，NV 与其它 ROCm 卡不受影响；但 forward_mla_rocm.py 近期被上游独立改动且本分支有多次 merge 历史，后续冲突风险略高。
- 影响：对用户：MI355X 上 DeepSeek-V4 每层少一次 scale relayout 拷贝，decode 延迟有轻微改善可能；GSM8K 显示 patched 低约 0.8–0.9 pt，作者归因于 fp4-MoE 噪声，与 producer 级 bit-exact 测试结论一致。对系统：fp8_utils.py 成为 bpreshuffle scale 布局的唯一仲裁点，后续新增 producer 只需遵循 emit_transposed_bpreshuffle_scale 与视图 helper。对团队：确立了 " 布局优化必须以真实内核等价测试验证 " 的评审标准，该标准在 #33165 与 #33166 之间互相印证，并推动 ROCm MLA 优化落在实际 dispatch 路径 forward_mla_rocm.py 上。
- 风险标记：依赖 aiter 内核 transpose_scale 契约，M==1 特判走拷贝回退，性能收益仅单次采样未确认，仅 gfx95 AMD 路径生效，多次 rebase/merge 历史复杂

关联脉络

- PR #33165 [AMD] dense-linear bpreshuffle scale no-copy（前序 PR，标题未在材料中提供）：本 PR 是其 direct follow-up：PR body 明确说 Follow-up to the dense-linear bpreshuffle scale no-copy，1am9trash 在 review 中也以 Same logic to #33165 背书。
- PR #31727 （引入 view_aiter_fused_rms_transposed_fp8_scale 的 PR）：本 PR 将自身新增的 bpreshuffle_fp8_scale_nocopy 合并到 #31727 已有的

view_aiter_fused_rms_transposed_fp8_scale，并依赖其修复的
fused_rms_fp8_group_quant transpose_scale 支持。