PR #50137 完整报告

vllm-project/vllm

[Bugfix] Don't transpose fused MoE quantization scales in `RoutedExperts.load_weights`

合并时间：2026-07-31 23:46

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/50137

执行摘要

- 一句话：修复融合 MoE 量化 scale 被误转置导致的 TP>1 加载回归
- 推荐动作：值得精读。改动小但处于核心加载路径，_orient_fused_weight 的判定条件与测试覆盖展示了如何用严格条件收窄启发式逻辑、避免误伤无 hidden 维张量。建议关注其已知局限（转置 + block 量化无法解析）以及未来用量化配置元数据替代形状推断的方向；合并前应补齐 PR 中承诺的端到端模型评估。

功能与动机

PR body 明确指出："Fixes a regression from #47058 that breaks loading of fused, per-channel-quantized MoE checkpoints with TP > 1."。#47058 重构后，RoutedExperts.load_weights 为兼容 Qwen3 VL MoE 这类转置存储的融合权重，引入按形状归一化方向的逻辑：对 3D 张量仅凭 shape[-1] != unpadded_hidden（w1/w3）或 shape[-2] != unpadded_hidden（w2）决定转置。该条件只排除了 " 恰好是 hidden 维 " 的情况，任何没有 hidden 维的 3D 辅助张量（如 per-channel scale [E, 2*I, 1]、block scale）都会被无条件转置，导致后续 chunk(2, dim=1) 与 TP 分片作用在错误轴上，报出 output with shape [0, 1] doesn't match the broadcast shape [0, 2048] 等错误。

实现拆解

1. 在 vllm/model_executor/layers/fused_moe/routed_experts.py 中新增静态方法 _orient_fused_weight(fused_weight, shard_id, unpadded_hidden)，将方向归一化逻辑从 load_weights 内联代码中抽出。方法按 shard_id 区分 w2（hidden 轴在 -2）与 w1/w3（hidden 轴在 -1），仅当 hidden_axis 尺寸不等于 unpadded_hidden 且 intermediate_axis 尺寸恰好等于 unpadded_hidden 时才 transpose(-1, -2)。相比旧逻辑是严格收窄：标准方向不受影响，转置 checkpoint 仍被归一化，无 hidden 维的量化张量原样返回。
2. 改写 load_weights 的 fused 分支：删除 w1/w3 与 w2 各自的内联 shape 判断，统一调用 _orient_fused_weight；w1/w3 仍走 chunk(2, dim=1) 拆分，w2 直接使用归一化后的张量。局部副本语义保留，避免转置影响 w3 的第二次迭代。
3. 修正 unpadded_hidden 取值：由直接读取 self.moe_config.hidden_dim_unpadded 改为 self.moe_config.hidden_dim_unpadded or self.moe_config.hidden_dim，防止部分配置未设置 unpadded 值时出错。

4. 测试配套：在 tests/kernels/moe/test_moe_weight_loading_padded.py 新增 TestOrientFusedWeight，覆盖 w13/w2 的标准与转置方向归一化、w13/w2 的 per-channel scale 与 block scale 不被误转置；其中 per-channel 用例在 main 分支会失败。本地 36 个测试全部通过，pre-commit（含 mypy-3.10）通过。
5. 未完成项：PR body 明确说明模型评估 still outstanding，计划用 Qwen3.5-122B-A10B-W8A8 TP=4 做端到端加载与评测后再合并。

关键文件：

- vllm/model_executor/layers/fused_moe/routed_experts.py（模块 专家加载；类别 source；类型 core-logic；符号 _orient_fused_weight, load_weights）：核心源码文件：新增 _orient_fused_weight 并重构 load_weights 的 fused 分支，是本次回归修复的关键。
- tests/kernels/moe/test_moe_weight_loading_padded.py（模块 测试覆盖；类别 test；类型 test-coverage；符号 TestOrientFusedWeight, test_w13_standard_orientation_is_untouched, test_w13_transposed_checkpoint_is_normalised, test_w2_standard_orientation_is_untouched）：测试配套：新增 TestOrientFusedWeight 覆盖两种方向与量化 scale 场景，其中 per-channel 用例在 main 分支会失败，直接锁定了回归。

关键符号：_orient_fused_weight, load_weights

关键源码片段

vllm/model_executor/layers/fused_moe/routed_experts.py

核心源码文件：新增 _orient_fused_weight 并重构 load_weights 的 fused 分支，是本次回归修复的关键。

```
# 方向归一化： w1/w3 期望 (..., intermediate, hidden)，w2 期望 (..., hidden, intermediate)。
# 旧实现只按 shape[-1] != hidden_size 判断转置，会把没有 hidden 维的量化 scale 也转置掉。
@staticmethod
def _orient_fused_weight(fused_weight: torch.Tensor, shard_id: str, unpadded_hidden: int) ->
torch.Tensor:
    if shard_id == "w2":
        hidden_axis, intermediate_axis = -2, -1
    else:
        hidden_axis, intermediate_axis = -1, -2
    # 严格收窄： 只有“当前 hidden 轴不对、而另一个数据轴正好是 hidden_size”时才转置，
    # 这样 per-channel scale (E, 2 * I, 1) 与 block scale 都原样返回。
    if (fused_weight.shape[hidden_axis] != unpadded_hidden
            and fused_weight.shape[intermediate_axis] == unpadded_hidden):
        return fused_weight.transpose(-1, -2)
    return fused_weight


def load_weights(self, weights: Iterable[tuple[str, torch.Tensor]]) -> Iterable[str]:
    expert_mapping = self.get_expert_mapping(include_fused=True)
    # 兜底： 部分配置未提供 unpadded 值，退回完整 hidden_dim
```

```
unpadded_hidden = self.moe_config.hidden_dim_unpadded or self.moe_config.hidden_dim
for expert_name, loaded_weight in weights:
    ...
    if is_fused:
        # 统一交给 _orient_fused_weight 归一化方向
        fused_weight = self._orient_fused_weight(loaded_weight, shard_id, unpadded_hidden)
        if shard_id in {"w1", "w3"}:
            # w1/w3 共用一张融合权重，按 intermediate 维拆成两份
            experts_shard = fused_weight.chunk(2, dim=1)[expert_id]
        else:
            experts_shard = fused_weight
        start = 0
    ...
```

tests/kernels/moe/test_moe_weight_loading_padded.py

测试配套：新增 TestOrientFusedWeight 覆盖两种方向与量化 scale 场景，其中 per-channel 用例在 main 分支会失败，直接锁定了回归。

```
class TestOrientFusedWeight:
    """覆盖两种写法的 w13/w2 以及不含 hidden 维的量化 scale。"""

    HIDDEN = 3072

    def test_w13_per_channel_scale_is_untouched(self):
        # 融合 per-channel scale 形状为 (E, 2 * I, 1)，没有 hidden 维；
        # 一旦被转置成 (E, 1, 2 * I)，chunk(2, dim=1) 就会切错轴。
        scale = torch.randn(8, 2048, 1)
        result = RoutedExperts._orient_fused_weight(scale, "w1", self.HIDDEN)
        assert result.shape == (8, 2048, 1)
        # chunk 仍按 intermediate 轴正常切分
        assert result.chunk(2, dim=1)[0].shape == (8, 1024, 1)
```

评论区精华

评审评论为空。claude[bot] 仅提示 "This pull request is from a fork — automated review isdisabled."；维护者DarkLight1337、Isotr0py、tlrmchlsmth均直接批准，无实质技术讨论。PR body 中作者预先做了 "Not a duplicate" 检查，并完整记录了测试计划、已知局限与 AI 辅助开发声明。

- 评审结论 (other): 获得批准并合并

风险与影响

- 风险：
  1. 核心路径变更：load_weights 是所有 fused MoE 模型加载权重的必经路径，判定条件的严格化直接影响真实权重与量化辅助张量两条分支。新旧条件在 " 两个轴恰好都等于 hidden_size" 时行为一致，但若未来出现 I == H 的模型且 checkpoint 需要转置，仍无法仅凭形状归一化。

2. 已知局限：同时 " 转置 + block 量化 " 的 checkpoint（[E, 2*I/block, H/block]）两个轴都不等于 hidden_size，无法从形状判断方向，需要量化配置驱动的布局元数据；作者表示暂未发现此类 checkpoint，且旧代码对 block scale 是 " 一律转置 "，新代码是 " 一律不转置 "，仍需实际验证。
3. 验证缺口：目前只有单元测试，PR 合并前的端到端模型评估尚未完成；虽已获 3 位维护者批准，但 TP > 1 的 per-channel 量化模型（如 Qwen3.5-122B-A10B-W8A8）缺少真实加载验证，是残余风险。
4. 兼容性：对标准方向 checkpoint 无行为变化，对已受影响用户是直接修复；但启发式形状推断本身仍存在边界，长期应转向量化配置驱动。 - 影响：影响范围集中在 fused MoE 权重加载路径：所有使用融合权重 + per-channel/block 量化的 checkpoint 在 TP > 1 下都可能受此 bug 影响（例如 Qwen3 VL MoE、Qwen3.5-122B-A10B-W8A8）。修复后这些模型恢复可加载性，标准方向 checkpoint 行为不变。对团队而言，_orient_fused_weight 成为新的方向约定入口，后续 fused_moe 模块化内核演进（如 #50133）需要复用或对齐该约定。 - 风险标记：核心路径变更，缺少端到端模型评估，已知加载局限

关联脉络

- PR #47058 Remove more unnecessary load_weights methods: 本 PR 是 #47058 的直接回归修复：#47058 重构了 RoutedExperts.load_weights 并引入按形状转置融合权重的逻辑，本 PR 修复其误转置量化 scale 的问题。