PR #51632 完整报告

vllm-project/vllm

[ROCm] [Bugfix] Fix Triton fused shared expert alignment

合并时间：2026-08-20 00:14

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/51632

执行摘要

- 一句话：修复 Triton 融合共享专家对齐计数，解决 MoE 精度崩溃
- 推荐动作：建议精读本 PR，它揭示了一个关键的数据契约细节：融合共享专家后，物理专家数可能与 global_num_experts 不一致。值得关注的设计决策是：在无 expert map 时使用实际权重行数，有 expert map 时保持原逻辑，这平衡了正确性与 EP 兼容性。同时，文档或注释可增强对专家计数语义的说明。

功能与动机

PR body 指出：MoE 模型使用 Triton 后端的融合共享专家时，可能产生损坏输出并遭受严重精度损失。融合共享专家会附加额外的专家 ID 和权重行，而 global_num_experts 仍只表示路由专家数。Triton 在 token 对齐时使用这个较小的计数，导致附加的共享专家 ID 被视为无效。当融合启用时，独立的共享专家路径被禁用，其贡献丢失。受影响的模型包括 EmbeddedLLM/MiniMax-M3-FP8-dynamic 在 ROCm gfx942 上的运行。

实现拆解

1. 修改核心对齐计数逻辑：在 vllm/model_executor/layers/fused_moe/experts/triton_moe.py 的 TritonExperts.apply 和 TritonWNA16Experts.apply 两个方法中，分别将 global_num_experts 替换为 num_align_experts = w1.shape[0] if expert_map is None else global_num_experts。这样在无 expert map 时使用物理专家行数（包含共享专家），有 expert map（EP 场景）时保持原逻辑，因为 ID 需要经过 expert_map 重映射。
2. 新增回归测试：在 tests/kernels/moe/test_moe.py 中新增 test_fused_shared_expert_alignment，构造 8 个路由专家 + 1 个共享专家的场景，使用 topk_ids 引用共享专家 ID（8），对比 modular_triton_fused_moe 与参考实现 torch_experts 的输出，确保共享专家贡献被正确计算。
3. 配套说明：该修复与原生 MXFP8 路径的处理方式保持一致，保证了一致性。测试覆盖了无 expert map 的场景，未来若需覆盖 EP 场景，可进一步扩展。

关键文件：

- vllm/model_executor/layers/fused_moe/experts/triton_moe.py（模块 MoE 执行；类别 source；类型 core-logic）：核心修复文件，包含两个 apply 方法中对专家对齐计数的关键修正，影响 Triton FSE 的 token 对齐正确性。

- tests/kernels/moe/test_moe.py（模块 MoE 测试；类别 test；类型 test-coverage；符号 test_fused_shared_expert_alignment）：新增回归测试 test_fused_shared_expert_alignment，验证无 expert map 时共享专家贡献被正确计算。

关键符号：apply, moe_align_block_size, _prepare_expert_assignment

关键源码片段

vllm/model_executor/layers/fused_moe/experts/triton_moe.py

核心修复文件，包含两个 apply 方法中对专家对齐计数的关键修正，影响 Triton FSE 的 token 对齐正确性。

```
# vllm/model_executor/layers/fused_moe/experts/triton_moe.py
# 位于 TritonExperts.apply 方法中，对应 3xx 行附近的调用点。

# Include fused shared-expert rows while preserving EP remapping.
# 关键修正：当没有 expert_map 时，用 w1.shape[0] 作为对齐专家总数，
# 因为融合共享专家会把额外专家行直接追加到权重末尾，而 global_num_experts
# 仍只表示路由专家数，会导致共享专家 ID 被判为无效而丢弃贡献。
num_align_experts = w1.shape[0] if expert_map is None else global_num_experts
sorted_token_ids, expert_ids, num_tokens_post_padded = (
    _prepare_expert_assignment(
        topk_ids,
        config,
        num_tokens,
        top_k_num,
        num_align_experts, # 替换原来的 global_num_experts
        expert_map, # 存在时保持 EP 重映射逻辑
        use_int8_w8a16=self.quant_config.use_int8_w8a16,
        use_int4_w4a16=self.quant_config.use_int4_w4a16,
        block_shape=self.block_shape,
    )
)
```

tests/kernels/moe/test_moe.py

新增回归测试 test_fused_shared_expert_alignment，验证无 expert map 时共享专家贡献被正确计算。

```
# tests/kernels/moe/test_moe.py
# 新增回归测试：验证融合共享专家对齐计数修复。

def test_fused_shared_expert_alignment(workspace_init):
    set_random_seed(7)
    m, n, k = 4, 64, 128
    routed_experts = 8
    physical_experts = routed_experts + 1 # 模拟融合一个共享专家
    dtype = torch.bfloat16

    a = torch.randn((m, k), device=DEVICE_TYPE, dtype=dtype) / 10
```

```
    # w1 包含 physical_experts 行，而 global_num_experts 传 routed_experts，
    # 这正好触发原 bug 的条件：共享专家行不在 global_num_experts 范围内。
    w1 = torch.randn((physical_experts, 2 * n, k), device=DEVICE_TYPE, dtype=dtype) / 10
    w2 = torch.randn((physical_experts, k, n), device=DEVICE_TYPE, dtype=dtype) / 10
    topk_ids = torch.tensor(
        [[0, 8], [1, 8], [2, 8], [3, 8]], device=DEVICE_TYPE, dtype=torch.int32
    )
    topk_weights = torch.tensor(
        [[0.5, 1.0]] * m, device=DEVICE_TYPE, dtype=torch.float32
    )

    moe_config = make_dummy_moe_config(
        num_experts=physical_experts,
        experts_per_token=2,
        hidden_dim=k,
        intermediate_size=n,
        in_dtype=dtype,
        max_num_tokens=m,
    )
    modular_moe = modular_triton_fused_moe(moe_config, FUSED_MOE_UNQUANTIZED_
    CONFIG)

    with set_current_vllm_config(vllm_config):
        expected = torch_experts(a, w1, w2, topk_weights, topk_ids)
        actual = modular_moe.apply(
            hidden_states=a,
            w1=w1,
            w2=w2,
            topk_weights=topk_weights,
            topk_ids=topk_ids,
            activation=MoEActivation.SILU,
            global_num_experts=routed_experts, # 故意传入较小的路由专家数
            expert_map=None, # 无 EP 重映射，使用物理专家数
            apply_router_weight_on_input=False,
        )

    # 放宽容差以允许浮点误差，但核心是验证共享专家 ID 8 被正确计算。
    torch.testing.assert_close(actual, expected, atol=2e-2, rtol=0)
```

评论区精华

Review 中无实质讨论，仅有 bot 自动评论和批准。维护者 maeehart 独立验证了修复：在 MI325X 上复现了基线失败（134/512 差异，最大 2.515625），应用 PR 后通过；AITER 启用的 MiniMax-M3 TP4 测试中，8-shot GSM8K 从 0.00% 恢复到 92.19%（flexible）/93.75%（strict），与 FSE 禁用时 89.84% 相当。

- 独立验证结果 (testing): 批准合并，修复有效。

风险与影响

- 风险：风险较低，但需关注：
  - EP 场景影响：修复仅在 *expert_map is None* 时改变行为，有 *expert map* 时逻辑不变，降低了 EP 场景的回归风险，但需确认所有调用点传参正确。
  - 物理专家数不等于真实专家数：w1.shape[0] 可能包含 *padding* 或量化压缩导致的额外行，但无 *expert map* 时通常与 topk_ids 范围一致，测试验证了该假设。
  - 测试覆盖有限：新增测试仅覆盖无 *expert map* 场景和有共享专家的情况，未覆盖 EP 或更多共享专家数量，后续可补充。
  - 影响：影响范围聚焦于 ROCm 平台上使用 Triton 后端且启用融合共享专家的 MoE 模型，特别是 *MiniMax-M3-FP8-dynamic* 等模型。修复后这些模型从完全不可用状态变为正常可用，准确率与禁用 FSE 时持平。对其他平台或非融合路径无影响，因为修改仅在相关条件分支内生效。
  - 风险标记：EP 场景未覆盖，测试仅覆盖无 *expert_map*，物理专家数假设

关联脉络

- PR #52775 [Kernel] SM120: stop routing misaligned-M blockwise FP8 GEMMs to the small-M swapAB config: 同为 MoE/FP8 相关的性能与正确性修复，涉及 GEMM 路由，与 Triton 路径可能有共性。
- PR #52704 [Bugfix][Quantization] Fix OCP MX MoE emulation silently skipping mxfp6 activation QDQ: 同为 Quantization/MoE 路径的静默正确性问题修复，思路类似。