PR #46408 完整报告

vllm-project/vllm

[Bugfix] Support -1 (invalid/non-local) slots in topk_ids for Triton MoE

合并时间：2026-06-25 04:59

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/46408

执行摘要

- 一句话：修复 Triton MoE 对 -1 槽位的处理
- 推荐动作：建议精读，尤其是新引入的两个 Triton kernel 的实现。该 PR 展示了如何通过定制 kernel 处理边界值，对理解 Triton 编程和处理异步并行中的特殊标记值有参考价值。

功能与动机

PR body 明确指出在专家并行下 dispatch 产生的 topk_ids 中包含 -1 值，表示无效 / 非本地专家槽。原有代码通过 *expert_map*[topk_ids] 索引时，-1 会映射到 *expert_map*[-1]（即最后一个元素），导致路由错误；同时 moe_sum 对中间结果求和时未跳过 -1 槽位，造成未初始化垃圾值混入结果。

实现拆解

1. 新增 remap_topk_to_local kernel：将全局专家 ID 映射为本地专家 ID，并保留 -1 值不变。该 kernel 使用条件加载，对有效 ID 执行 *expert_map* 查找，无效 ID 直接输出 -1。相比原有 *expert_map*[topk_ids] 方式，避免了 -1 包装问题。
2. 新增 masked_moe_sum kernel：对中间结果按 token 进行求和，跳过 topk_ids 中值为 -1 的槽位。该 kernel 在循环中检查每个槽位是否有效，仅对有效槽位累加，确保结果不含未初始化数据。
3. 移除无用导入：删除了 from vllm import *_custom_ops* as ops 导入，因为不再使用 ops.moe_sum。
4. 修改 apply 方法：将原来 *expert_map*[topk_ids] 替换为 remap_topk_to_local(topk_ids, *expert_map*)，并将 moe_sum 替换为 masked_moe_sum。

关键文件：

- vllm/model_executor/layers/fused_moe/experts/gpt_oss_triton_kernels_moe.py（模块 MoE；类别 source；类型 core-logic；符号 _masked_topk_sum_kernel, masked_moe_sum, _remap_topk_to_local_kernel, remap_topk_to_local）：核心变更文件，新增两个 Triton kernel 并修改 apply 方法，修复 -1 槽位处理。

关键符号：_remap_topk_to_local_kernel, remap_topk_to_local, _masked_topk_sum_kernel, masked_moe_sum

关键源码片段

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

核心变更文件，新增两个 Triton kernel 并修改 apply 方法，修复 -1 槽位处理。

```
# vllm/model_executor/layers/fused_moe/experts/gpt_oss_triton_kernels_moe.py

@triton.jit
def _remap_topk_to_local_kernel(
    topk_ids_ptr, # [n] global expert IDs (-1 = invalid)
    expert_map_ptr, # [num_experts] global->local (-1 for non-local)
    out_ptr, # [n] int64 local expert IDs (-1 for invalid/non-local)
    n_elements,
    BLOCK: tl.constexpr,
):
    pid = tl.program_id(0)
    offs = pid * BLOCK + tl.arange(0, BLOCK)
    mask = offs < n_elements
    tid = tl.load(topk_ids_ptr + offs, mask=mask, other=-1)
    # Clamp invalid indices to avoid OOB, then select -1 for them.
    # This preserves -1, unlike plain expert_map[-1] which would wrap.
    valid = tid >= 0
    idx = tl.where(valid, tid, 0)
    local = tl.load(expert_map_ptr + idx, mask=mask, other=-1)
    out = tl.where(valid, local.to(tl.int64), -1)
    tl.store(out_ptr + offs, out, mask=mask)


def remap_topk_to_local(
    topk_ids: torch.Tensor, expert_map: torch.Tensor
) -> torch.Tensor:
    """Fused global->local expert-id mapping over a topk_ids tensor, preserving -1.
    Replaces ``torch.where(topk_ids >= 0, expert_map[topk_ids.clamp(min=0)], -1)``
    with one kernel. Returns a NEW int64 tensor."""
    out = torch.empty_like(topk_ids, dtype=torch.int64)
    n = topk_ids.numel()
    BLOCK = 512
    grid = (triton.cdiv(n, BLOCK),)
    _remap_topk_to_local_kernel[grid](topk_ids, expert_map, out, n, BLOCK=BLOCK)
    return out


@triton.jit
def _masked_topk_sum_kernel(
    inp_ptr, # (M, topk, K) contiguous
    topk_ids_ptr, # (M, topk) int: -1 marks an invalid / non-local slot
    out_ptr, # (M, K), same dtype as inp
    K,
    topk: tl.constexpr,
```

```
    BLOCK_K: tl.constexpr,
):
    pid_m = tl.program_id(0).to(tl.int64)
    k = tl.program_id(1) * BLOCK_K + tl.arange(0, BLOCK_K)
    k_mask = k < K
    base = pid_m * topk
    acc = tl.zeros((BLOCK_K,), dtype=tl.float32)
    for j in tl.static_range(topk):
        eid = tl.load(topk_ids_ptr + base + j)
        # Skip invalid slots to avoid reading uninitialized data
        if eid >= 0:
            x = tl.load(inp_ptr + (base + j) * K + k, mask=k_mask)
            acc += x.to(tl.float32)
    tl.store(out_ptr + pid_m * K + k, acc.to(out_ptr.dtype.element_ty), mask=k_mask)


def masked_moe_sum(
    intermediate: torch.Tensor, # (M, topk, K)
    topk_ids: torch.Tensor, # (M, topk) int, -1 = invalid / non-local slot
    output: torch.Tensor, # (M, K)
) -> None:
    M, topk, K = intermediate.shape
    BLOCK_K = 1024
    grid = (M, triton.cdiv(K, BLOCK_K))
    _masked_topk_sum_kernel[grid](
        intermediate, topk_ids, output, K, topk=topk, BLOCK_K=BLOCK_K
    )
```

评论区精华

该 PR 没有 review 评论。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 回归风险：对于非 EP 场景（expert_map 为 None），行为不变，但需要确认代码路径正确跳过 remap 并仍能正确调用 masked_moe_sum（使用全有效输入）。
  2. 性能风险：新增两个 Triton kernel 可能带来轻微额外开销，但原操作也涉及多步操作，融合后可能持平或更优。
  3. 兼容性：masked_moe_sum 输出类型由 intermediate dtype 决定，与原有 moe_sum 行为一致。 - 影响：影响范围限定于使用了 Triton MoE 的专家并行场景。对非 EP 用户无影响，对 EP 用户将修复路由和求和 bug，提升模型推理正确性。影响程度中等，因为涉及核心 MoE 计算路径。 - 风险标记：核心路径变更，缺少测试覆盖

关联脉络

- PR #46061 Add unit test for deep_gemm_utils.apply_expert_map: PR body 提及此 PR 仅测试已存在的 scalar device helper，而本 PR 引入了新的 vectorized kernel，两者不同但相关。
- PR #46406 [Bugfix] Support non-power-of-2 top_k in legacy triton_kernels routing: 同为修复 Triton MoE 相关 bug，修改同一文件的不同部分。