PR #24171 完整报告

sgl-project/sglang

[LoRA][MOE] Fix EP correctness in MoE LoRA slicing and virtual-experts kernels

合并时间：2026-05-01 13:42

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24171

执行摘要

- 一句话：修复 EP 下 MoE LoRA 切片与虚拟专家内核正确性
- 推荐动作：建议详细阅读该 PR，尤其是虚拟专家内核的哨兵处理技巧和 MoE 切片索引的修复逻辑，这些模式可复用于其他并行度组合的 LoRA 支持。测试设计精巧，值得参考。

功能与动机

修复两个在 EP 下 MoE LoRA 路径中的正确性问题：

1) tp_size>ep_size>1 时适配器加载崩溃，因为 load_lora_weight_to_buffer 传递了外部 tp_rank 而不是 moe_tp_rank； 2) 虚拟专家内核未正确处理 EP 分发后的 -1 哨兵 topk_ids，导致 OOB 访问。这些问题可复现于 Qwen/Qwen3-VL-30B-A3B-Instruct-FP8 等混合架构模型。

实现拆解

1. 修正 MoE 切片索引：在 mem_pool.py 的 load_lora_weight_to_buffer 方法中，将传递给 slice_moe_lora_{a,b}_weights 的秩参数从 self.tp_rank 改为 self.moe_tp_rank，使其与 moe_tp_size 定义的切分方式一致，修复 1<ep_size<tp_size 时切片越界错误。
2. 修正虚拟专家内核：在 virtual_experts.py 中，_fused_virtual_topk_ids_kernel 使用 tl.where(base<0, base, shifted) 保留负哨兵值；_align_block_size_torch 创建编号为 num_experts 的哨兵桶，将无效 ID 路由至此桶，确保其不被分配真实专家块。
3. 添加 active_target_modules 过滤：在 mem_pool.py 中，遍历当前层模块时记录实际激活的目标模块集合，后续缓冲区复制循环仅处理集合内的键，跳过混合架构中不存在的模块，避免冗余 GPU 零填充。
4. 添加 layer_id 守卫：在 lora_manager.py 的 init_lora_modules 中，对 FusedMoE 包装循环添加 if layer_id is None: continue，防止非标准层结构导致 self.lora_modules 索引错误。
5. 新增回归测试：test_virtual_experts_kernels.py 提供 8 个 GPU 测试覆盖哨兵保留、正 ID 重映射、无 LoRA token 行为及哨兵桶分配；test_mem_pool_ep_unit.py 新增一个 CPU 测试验证 load_lora_weight_to_buffer 在 tp=4 ep=2 时传递 moe_tp_rank。

关键文件：

- python/sglang/srt/lora/mem_pool.py（模块 内存池；类别 source；类型 core-logic；符号 LoRAMemoryPool.load_lora_weight_to_buffer）：核心修复：修正 MoE 切片索引为 moe_tp_rank，并引入 active_target_modules 过滤，避免混合架构模型的冗余零填充。
- python/sglang/srt/lora/triton_ops/virtual_experts.py（模块 虚拟专家内核；类别 infra；类型 infrastructure；符号 _fused_virtual_topk_ids_kernel, _align_block_size_torch）：核心修复：修正虚拟专家内核，保留负 sentinel 并添加 sentinel 桶处理 torch.compile 回退。
- python/sglang/srt/lora/lora_manager.py（模块 LoRA 管理器；类别 source；类型 core-logic；符号 init_lora_modules）：稳定性改进：在 FusedMoE 包装循环中添加 layer_id 守卫，防止 None 索引。
- test/registered/lora/test_virtual_experts_kernels.py（模块 虚拟专家测试；类别 test；类型 test-coverage；符号 TestFusedVirtualTopkIdsPreservesSentinels, setUpClass, test_negative_sentinels_preserved, test_positive_topk_remapped_correctly）：新增 GPU 回归测试，覆盖虚拟专家内核的所有修复路径，证明哨兵处理正确。
- test/registered/unit/lora/test_mem_pool_ep_unit.py（模块 内存池测试；类别 test；类型 test-coverage；符号 TestLoadBufferPassesMoeTpRankToSlice, _StopAfterCapture, test_moe_tp_rank_used_for_slicing_when_ep_lt_tp, capture_a）：新增回归测试，专门验证 load_lora_weight_to_buffer 在 ep<tp 时传递 moe_tp_rank。

关键符号：load_lora_weight_to_buffer, _fused_virtual_topk_ids_kernel, _align_block_size_torch, init_lora_modules

关键源码片段

python/sglang/srt/lora/mem_pool.py

核心修复：修正 MoE 切片索引为 moe_tp_rank，并引入 active_target_modules 过滤，避免混合架构模型的冗余零填充。

```
# 摘自 load_lora_weight_to_buffer 方法
active_target_modules: Set[str] = set()
cur_layer_modules = lora_modules[layer_id]
for module_name, module in cur_layer_modules.items():
    from sglang.srt.lora.layers import FusedMoEWithLoRA
    if isinstance(module, FusedMoEWithLoRA):
        moe_target_modules = ['gate_up_proj_moe', 'down_proj_moe']
        for target_module in moe_target_modules:
            active_target_modules.add(target_module)
            if temp_A_buffer.get(target_module) is not None:
                temp_A_buffer[target_module] = (
                    module.slice_moe_lora_a_weights(
                        temp_A_buffer[target_module],
                        self.moe_tp_rank, # 使用 moe_tp_rank 而非 tp_rank
                        target_module,
                    )
                )
            if temp_B_buffer.get(target_module) is not None:
```

```
                temp_B_buffer[target_module] = (
                    module.slice_moe_lora_b_weights(
                        temp_B_buffer[target_module],
                        self.moe_tp_rank, # 同上
                        target_module,
                    )
                )
            continue
        target_module = get_target_module_name(module_name, self.target_modules)
        active_target_modules.add(target_module) # 无权重也标记，确保零填充
        # ... 常规切片 ...

    # 跳过非活动模块的缓冲区复制
    for name, weights in temp_A_buffer.items():
        if name not in active_target_modules:
            continue
        # ... 写缓冲区 ...
```

python/sglang/srt/lora/triton_ops/virtual_experts.py

核心修复：修正虚拟专家内核，保留负 sentinel 并添加 sentinel 桶处理 torch.compile 回退。

```
# _fused_virtual_topk_ids_kernel 片段
base = tl.load(topk_ids_ptr + offs, mask=valid, other=0)
# 保留负 sentinel：若 base < 0，则保持不变
shifted = base + safe_lora * num_experts_for_weight
result = tl.where(base < 0, base, shifted)
tl.store(virtual_topk_ids_ptr + offs, result, mask=valid)

# _align_block_size_torch 片段
valid_mask = (flat_topk_ids >= 0) & (flat_topk_ids < num_experts)
safe_topk_ids = torch.where(
    valid_mask,
    flat_topk_ids,
    torch.full_like(flat_topk_ids, sentinel), # 无效 ID -> sentinel 桶
)
bucket_count = num_experts + 1 # 增加 sentinel 桶
sorted_order = torch.argsort(safe_topk_ids)
sorted_expert_ids = safe_topk_ids[sorted_order]
# sentinel 桶的块保持 -1 标记，消费内核会跳过
```

评论区精华

该 PR 无实质性 review 讨论，获得直接批准。设计决策（如与 #23632 的关系、性能影响评估）已在 PR body 中说明。主要考量包括兼容性（ep==tp 或 ep==1 时无变化）和虚拟专家内核的执行开销（一个额外 tl.where 和少量元素级操作，<1% 开销）。

- Review 讨论 (other): 无需变更，直接合并。

风险与影响

- 风险：变更涉及核心调度和内存路径，可能影响以下方面： 1) mem_pool.py 的 active_target_modules 过滤依赖 isinstance 判断，若存在未知的 FusedMoEWithLoRA 子类未覆盖，可能导致模块被意外跳过；但现有测试覆盖常规子类。 2) 虚拟专家内核的哨兵桶增加一个额外桶，但桶计数从 num_experts 变为 num_experts+1，消费者内核需支持此变化；根据现有 if off_expert == -1: return 逻辑，哨兵桶的块保持 -1 标记，不会引起问题。 3) 作者声明在 tp=4 ep=2 和 tp=4 ep=1/4 上已验证，但未在其他 GPU 架构测试，存在潜在兼容性风险。整体风险较低。
- 影响：影响范围： 1) 用户：使用 Expert Parallelism + LoRA 的用户，尤其是混合架构模型（如 Qwen3.5），将不再遇到适配器加载崩溃和推理错误。 2) 系统：新增一套 GPU 单元测试（约 12s），不增加运行时开销。 3) 团队：回退到旧逻辑的证据是测试覆盖的关键点，为后续维护提供信心。
  - 风险标记：核心路径变更，虚拟专家内核修正，新增过滤逻辑

关联脉络

- PR #23632 Qwen 3.5 LoRA support: 相关：该 PR 是 #23632 的子集，专注于 EP 正确性修复，独立于 Qwen3.5 包装器。
- PR #23594 LoRA support for qwen3.5 and nemotron3: 相关：#23594 已合并，提供了 Qwen3.5 的广义 n-slice 路径，使本 PR 无需额外包装器。