PR #23178 完整报告

sgl-project/sglang

[LoRA] Fix EP + per-expert MoE LoRA illegal memory access

合并时间：2026-04-23 05:22

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23178

执行摘要

- 一句话：修复 EP + MoE LoRA 非法内存访问
- 推荐动作：建议深入阅读此 PR，特别是 _get_moe_ep_context、_moe_runner_keeps_global_expert_ids 的异常安全设计和 _iter_local_expert_weights 的字典 / 张量统一处理模式。这是如何在多个模块（内存池、CUDA 内核、调度器）间协调 EP 行为的典型案例，值得在类似分布式重构中参考。

功能与动机

Two related bugs prevent --tp N --ep M --enable-lora --lora-backend triton --moe-runner-backend triton from running on any MoE model with per-expert LoRA adapters; both reliably crash in the first forward pass with CUDA error: an illegal memory access was encountered.

实现拆解

实现按以下步骤进行：

1. 捕获 EP 上下文并决定是否使用局部 ID：在 LoRAMemoryPool.__init__ 中新增 _get_moe_ep_context() 和 _moe_runner_keeps_global_expert_ids() 辅助函数，分别获取 EP 大小和 rank 以及当前 MoE 运行器是否保持全局专家 ID。仅当 ep_size > 1 且运行器为 Triton/DeepGEMM（会重置 topk_ids 为局部 ID）时，设置 moe_use_local_expert_ids = True；对于 FlashInfer 系列运行器或专家数不能整除的情况，保持全局 ID 索引。
2. 调整 LoRA 缓冲区分配粒度：新增 _get_num_local_experts() 方法，根据 num_experts 和 ep_size 计算出该 rank 实际拥有的专家数。在 get_lora_A_shape 和 get_lora_B_shape 中，MoE 模块的专家维度从 num_experts 改为 self._get_num_local_experts()，从而每个 rank 只分配自己实际需要的缓冲区。
3. 新增全局到局部专家 ID 的映射和权重加载辅助：新增 _global_to_local_expert_id() 方法将全局 expert ID 映射为该 rank 的局部索引，并检查所有权。新增 _iter_local_expert_weights() 方法，用于在加载 LoRA 权重时，从全局权重字典或 3D 张量中按局部索引提取对应专家的权重，并将不属于本 rank 的专家权重置零。该辅助方法在 load_lora_weight_to_buffer 的所有四个 per-expert 加载路径中使用。

4. 修复 CUDA 内核中的哨兵过滤：在 moe_lora_align_kernel.cu 的 _count_and_sort_expert_tokens 中，将过滤条件从 expert_id >= num_experts 扩展为 expert_id < 0 || expert_id >= num_experts，以排除 EP 写入的 -1 哨兵，防止越界的 atomicAdd 和后续内存访问。
5. 添加完整的单元测试：新增 test/registered/unit/lora/test_mem_pool_ep_unit.py，包含 22 个 CPU 可运行的测试用例，覆盖 _get_num_experts、_get_num_local_experts、_global_to_local_expert_id 和 _iter_local_expert_weights 在不同 EP 配置和运行器后端下的行为。这些测试通过 __new__ 直接构造 LoRAMemoryPool 实例，无需 CUDA 或分布式初始化。

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 _get_moe_ep_context, _get_moe_tp_context, _moe_runner_keeps_global_expert_ids, _get_num_local_experts）：核心修改文件，重写了 LoRAMemoryPool 在 EP 下的缓冲区分配策略，新增了 EP 上下文捕获、局部专家数计算、全局到局部 ID 映射、权重迭代辅助等关键函数。
- test/registered/unit/lora/test_mem_pool_ep_unit.py（模块 单元测试；类别 test；类型 test-coverage；符号 _make_pool, _make_fake_base_model, TestNumExpertHelpers, test_num_experts_read_from_config）：新增的单元测试文件，包含 22 个 CPU 可运行测试用例，覆盖了 EP 上下文捕获、局部专家数计算、ID 映射和权重迭代等所有新增功能的正确性，是保证修改质量的关键。
- python/sglang/jit_kernel/csrc/lora/moe_lora_align_kernel.cu（模块 JIT 内核；类别 other；类型 core-logic；符号 _count_and_sort_expert_tokens）：CUDA 内核修改，添加对 EP 哨兵值 -1 的过滤，是修复非法内存访问的第二部分。

关键符号：_get_moe_ep_context, _get_moe_tp_context, _moe_runner_keeps_global_expert_ids, _get_num_local_experts, _global_to_local_expert_id, _iter_local_expert_weights, _count_and_sort_expert_tokens

关键源码片段

python/sglang/jit_kernel/csrc/lora/moe_lora_align_kernel.cu

CUDA 内核修改，添加对 EP 哨兵值 -1 的过滤，是修复非法内存访问的第二部分。

```
// ... 在 _count_and_sort_expert_tokens 函数中
for (size_t i = tid; i < numel; i += stride) {
    int32_t expert_id = topk_ids[i];
    // Under EP, StandardDispatcher writes -1 for experts not owned by this
    // rank; must filter the sentinel before indexing cumsum/sorted buffers.
    if (expert_id < 0 || expert_id >= num_experts) {
        continue;
    }
    // ... 后续处理
```

评论区精华

Gemini Code Assist 机器人提出了两个反馈：

1. 高优先级：moe_use_local_expert_ids 的初始化未考虑专家总数不能被 EP 大小整除的情况。如果专家数不可整除，_get_num_local_experts 回退到全局计数，但 moe_use_local_expert_ids 仍为 True，导致 _global_to_local_expert_id 对 rank>0 产生错误的映射。
2. 中优先级：如果采纳前述建议在 __init__ 中加入整除检查，则 _get_num_local_experts 中的 if not self.moe_use_local_expert_ids or total % self.moe_ep_s... 的模运算变得冗余，可以简化逻辑。

作者在第二个 commit（63fdd61fb0d99517a45ad30d91f00b3f82e9edd7）中修复了第一个问题，在 __init__ 中添加了整除判断，同时保留了 _get_num_local_experts 中的模运算作为安全网。该修改获得了最终批准。

- moe_use_local_expert_ids 初始化缺少整除检查 (design): 作者在第二次提交中修复了此问题，在 __init__中添加了 num_experts_global % self.moe_ep_size == 0 检查，强制不可整除时 moe_use_local_expert_ids 为 False。同时保留了 _get_num_local_experts 中的模运算作为安全网。
- _get_num_local_experts 中冗余模运算 (style): 作者选择保留该分支作为安全网（defensive programming），未移除但接受了建议的方向。

风险与影响

- 风险：主要风险在于兼容性：
  - 纯 TP 路径：moe_ep_size=1 时 moe_use_local_expert_ids=False，缓冲区分配和权重加载行为与修改前完全一致，已通过 --tp 2 端到端验证，无回归风险。
  - 保持全局 ID 的运行器：FlashInfer CUTLASS/CuteDSL/TRTLLM-routed 等运行器不重置 topk_ids，_moe_runner_keeps_global_expert_ids() 返回 True，因此缓冲区保持全局键索引，与这些运行器的期望一致。
  - 不均匀专家分割：num_experts % ep_size != 0 时，moe_use_local_expert_ids 被强制为 False，避免了错误截断专家数。
  - 性能影响：引入了额外的函数调用和条件判断，但仅发生在 MoE LoRA 权重加载阶段（非热点路径），对推理时延无影响。
  - 影响：用户影响：之前所有在 TP+EP 下使用 per-expert LoRA 适配器（如 Qwen3-VL-30B-A3B-Instruct-FP8 通过 --tp 4 --ep 4）的部署均会立即崩溃；修复后可以正常完成首次前向传播并生成结果。仅影响同时使用 EP 和 MoE LoRA 的用户，纯 TP 或非 LoRA 用户无影响。系统影响：LoRAMemoryPool 的内存分配策略改变，每个 EP rank 只分配本地专家数的缓冲区，减少了显存占用（EP 下每 rank 只持有总专家的 1/ep_size 个专家权重）。团队影响：新增的单元测试框架允许不依赖分布式环境验证 EP 逻辑，降低了后续维护成本。
- 风险标记：核心内存分配逻辑变更，分布式 EP 路径依赖手工端到端验证，EP 专家数不均分降级逻辑

关联脉络

- PR #23594 LoRA support for qwen3.5 and nemotron3: 同一功能线（LoRA），都与 LoRAMemoryPool 和 LoRA 加载逻辑相关，本 PR 修复了该 PR 引入的 EP 兼容性问题。