PR #24231 完整报告

sgl-project/sglang

[Bug][VLM] Fix shared memory bug: deep copy mm_items to prevent cross…

合并时间：2026-05-20 11:06

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24231

执行摘要

- 一句话：修复 VLM 共享内存跨请求突变 bug
- 推荐动作：值得合并的紧急 bugfix。建议后续将深拷贝逻辑抽取为 _deep_copy_mm_items 方法以改善可维护性。

功能与动机

修复当 n > 1 时，因浅拷贝共享 mm_items 导致后续请求访问已删除的共享内存文件而崩溃（FileNotFoundError），如 PR body 中堆栈所示。

实现拆解

1. 定位问题位置：_handle_batch_request 方法中有两处通过 copy.copy 浅拷贝 tokenized_objs[i]。当 n > 1 且请求中包含多模态输入时，mm_inputs.mm_items 列表被多个副本共享。
2. 添加深拷贝逻辑：在两处浅拷贝后（第 1445 行和第 1465 行附近），对 tokenized_obj 的 mm_inputs 做 copy.copy，再对 mm_inputs.mm_items 中的每个 item 做 copy.copy，确保每个展开的请求具有独立的 mm_items 和 shm 引用。
3. 防御性检查：添加 hasattr(tokenized_obj, "mm_inputs") and tokenized_obj.mm_inputs 检查，避免非多模态请求不必要深拷贝。
4. 影响范围：只修改了 python/sglang/srt/managers/tokenizer_manager.py 一个文件，增加了 12 行代码，无删减。

关键文件：

- python/sglang/srt/managers/tokenizer_manager.py（模块 调度器；类别 source；类型 core-logic）：核心修复文件，在 _handle_batch_request 方法的两处展开点添加深拷贝逻辑，确保每个请求的 mm_items 独立。

关键符号：未识别

关键源码片段

python/sglang/srt/managers/tokenizer_manager.py

核心修复文件，在 _handle_batch_request 方法的两处展开点添加深拷贝逻辑，确保每个请求的 mm_items 独立。

```
# python/sglang/srt/managers/tokenizer_manager.py
# 在 _handle_batch_request 方法中，原本两处浅拷贝 tokenized_objs[i] 后直接使用，
# 当请求为多模态且 n > 1 时，mm_items 被多个副本共享，导致 shm 被提前 unlink。
# 修复：在每处浅拷贝后立即深拷贝 mm_inputs 和 mm_items。

# 第一处：缓存公共前缀阶段
for i in range(batch_size):
    tmp_obj = copy.copy(objs[i])
    tokenized_obj = copy.copy(tokenized_objs[i])
    # 确保每个 tokenized_obj 拥有独立的 mm_items，防止 wrap_shm_features 修改原始对象
    if hasattr(tokenized_obj, "mm_inputs") and tokenized_obj.mm_inputs:
        tokenized_obj.mm_inputs = copy.copy(tokenized_obj.mm_inputs)
        tokenized_obj.mm_inputs.mm_items = [
            copy.copy(item) for item in tokenized_obj.mm_inputs.mm_items
        ]
    tokenized_obj.rid = tmp_obj.regenerate_rid()
    tokenized_obj.sampling_params = copy.copy(tokenized_obj.sampling_params)
    tokenized_obj.sampling_params.max_new_tokens = 0
    tokenized_obj.stream = False
    self._init_req_state(tmp_obj)
    self._send_one_request(tokenized_obj)
    await self._wait_one_response(tmp_obj, request).__anext__()

# 第二处：正式展开并行采样
for i in range(batch_size):
    for _ in range(obj.parallel_sample_num):
        tmp_obj = copy.copy(objs[i])
        tokenized_obj = copy.copy(tokenized_objs[i])
        # 同上：深拷贝 mm_inputs.mm_items 确保独立
        if hasattr(tokenized_obj, "mm_inputs") and tokenized_obj.mm_inputs:
            tokenized_obj.mm_inputs = copy.copy(tokenized_obj.mm_inputs)
            tokenized_obj.mm_inputs.mm_items = [
                copy.copy(item) for item in tokenized_obj.mm_inputs.mm_items
            ]
        tokenized_obj.rid = tmp_obj.regenerate_rid()
        self._init_req_state(tmp_obj)
        tokenized_obj.time_stats = self.rid_to_state[tmp_obj.rid].time_stats
        self._send_one_request(tokenized_obj)
        generators.append(self._wait_one_response(tmp_obj, request))
        rids.append(tmp_obj.rid)
```

评论区精华

Review 中 gemini-code-assist[bot] 提出“深拷贝逻辑在两处重复，应抽取为辅助方法以减少重复和不一致性风险”，但该建议未被采纳。yhyang201 和 mickqian 均批准了该 PR。

- 深拷贝逻辑重复，建议抽取帮助方法 (design): 未采纳。PR 作者未回复该建议，且 reviewers 已批准 PR。

风险与影响

- 风险：低风险：
  - 仅对多模态请求场景新增深拷贝，非多模态请求路径无变化。
  - 深拷贝的 mm_inputs.mm_items 结构简单（可能是路径字符串或元数据），性能影响极小。
  - 未引入新的依赖或配置。
  - 潜在风险：若 mm_inputs.mm_items 中存在不可深拷贝的对象（如锁或特殊对象），copy.copy 可能失败，但目前 copy.copy 期望这些 item 是简单结构。
  - 影响：影响范围：所有使用并行采样（n > 1）的多模态推理请求。修复前会百分之百触发崩溃（FileNotFoundError），修复后可正常执行。用户影响：用户无需修改任何配置或代码，更新后可自动受益。系统影响：每个多模态请求展开时增加一次浅层深拷贝，开销可忽略。
- 风险标记：缺少测试覆盖

关联脉络

- 暂无明显关联 PR