PR #23346 完整报告

sgl-project/sglang

[Bug Fix] Preserve decode state across retract-resume of GLM-5.1

合并时间：2026-05-07 21:37

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23346

执行摘要

- 一句话：修复 GLM-5.1 PD retract-resume 状态别名 bug
- 推荐动作：该 PR 是一次关键的 bugfix，涉及分布式推理中状态一致性的核心问题，值得相关开发人员精读，尤其是理解状态别名和 CPU offloading 的设计权衡。推荐的关注点：clone() 对性能的影响、get_cpu_copy 与基类的一致性设计。

功能与动机

在 PD 分离部署中，retract-resume 操作会释放和重用槽位，但原始引用并未更新，导致 resume 后读取到其他请求的脏数据，产生乱码输出，甚至通过 radix tree 泄漏。该 PR 旨在修复这些状态别名问题，保证 decode 状态的正确保存和恢复。

实现拆解

1. 修复 MetadataBuffers.get_buf 数据别名：在 python/sglang/srt/disaggregation/utils.py 中，将 get_buf 返回的切片改为返回 .clone()，避免后续 set_buf 覆盖共享内存后导致 retract-resume 请求使用脏数据。
2. 释放 index_k_with_scale_buffer：在 python/sglang/srt/mem_cache/memory_pool.py 中，为 NSATokenToKVPool 添加 _clear_buffers 方法，确保在销毁时释放额外的 index_k_with_scale_buffer，与基类清理逻辑一致。
3. CPU 卸下 / 恢复 index_k_with_scale_buffer：覆盖 get_cpu_copy 和 load_cpu_copy，在离载时同步将 index_k_with_scale_buffer 逐层分块拷贝到 CPU，恢复时再写回 GPU，确保 resume 后 NSA 注意力计算使用正确的 key/scale 数据。

关键文件：

- python/sglang/srt/mem_cache/memory_pool.py（模块 内存池；类别 source；类型 core-logic；符号 _clear_buffers, get_cpu_copy, load_cpu_copy）：核心修复，为 NSATokenToKVPool 添加了 index_k_with_scale_buffer 的清理和 CPU offload/restore 逻辑，确保 retract-resume 时 NSA 注意力计算正确。
- python/sglang/srt/disaggregation/utils.py（模块 输出缓冲；类别 source；类型 core-logic）：修复 MetadataBuffers.get_buf 返回视图导致的数据别名问题，通过 clone() 确保 retract-resume 请求不会读到被覆盖的数据。

关键符号：_clear_buffers, get_cpu_copy, load_cpu_copy, get_buf

关键源码片段

python/sglang/srt/mem_cache/memory_pool.py

核心修复，为 NSATokenToKVPool 添加了 index_k_with_scale_buffer 的清理和 CPU offload/restore 逻辑，确保 retract-resume 时 NSA 注意力计算正确。

```
def get_cpu_copy(self, indices):
    # NSA keeps a page-indexed index_k_with_scale_buffer alongside kv_buffer.
    # Retract frees the slots/pages and they get reused by other reqs'
    # set_index_k_scale_buffer, so we must offload it here too -- otherwise
    # resume restores kv_buffer but leaves foreign index/scale in place and
    # NSA attention reads garbage at those token positions.
    kv_cache_cpu = super().get_cpu_copy(indices)
    page_indices = indices[:: self.page_size] // self.page_size
    torch.cuda.synchronize()
    index_k_cpu = []
    chunk_size = self.cpu_offloading_chunk_size
    page_chunk_size = max(1, chunk_size // self.page_size)
    for layer_id in range(self.layer_num):
        index_k_cpu.append([])
        for i in range(0, len(page_indices), page_chunk_size):
            chunk_page_indices = page_indices[i : i + page_chunk_size]
            idx_cpu = self.index_k_with_scale_buffer[layer_id][chunk_page_indices].to(
                "cpu", non_blocking=True
            )
            index_k_cpu[-1].append(idx_cpu)
        torch.cuda.synchronize()
    return {"kv": kv_cache_cpu, "index_k": index_k_cpu}

def load_cpu_copy(self, kv_cache_cpu_dict, indices):
    super().load_cpu_copy(kv_cache_cpu_dict["kv"], indices)
    page_indices = indices[:: self.page_size] // self.page_size
    index_k_cpu = kv_cache_cpu_dict["index_k"]
    torch.cuda.synchronize()
    chunk_size = self.cpu_offloading_chunk_size
    page_chunk_size = max(1, chunk_size // self.page_size)
    for layer_id in range(self.layer_num):
        for i in range(0, len(page_indices), page_chunk_size):
            chunk_page_indices = page_indices[i : i + page_chunk_size]
            idx_cpu = index_k_cpu[layer_id][i // page_chunk_size]
            assert idx_cpu.shape[0] == len(chunk_page_indices)
            idx_chunk = idx_cpu.to(
                self.index_k_with_scale_buffer[0].device, non_blocking=True
            )
            self.index_k_with_scale_buffer[layer_id][chunk_page_indices] = idx_chunk
        torch.cuda.synchronize()
```

python/sglang/srt/disaggregation/utils.py

修复 MetadataBuffers.get_buf 返回视图导致的数据别名问题，通过 clone() 确保 retract-resume 请求不会读到被覆盖的数据。

```
  def get_buf(self, idx: int):
      # Return a clone of each tensor instead of a view (previously stacked slice).
      # When a slot is freed and reused, set_buf overwrites the underlying storage.
      # Retract-resumed reqs run process_prebuild much later and would see foreign
      # data (EAGLE hidden states, topk, etc.) if we returned views.
      return (
          self.output_ids[idx].clone(),
          self.cached_tokens[idx].clone(),
          self.output_token_logprobs_val[idx].clone(),
          self.output_token_logprobs_idx[idx].clone(),
          self.output_top_logprobs_val[idx].clone(),
          self.output_top_logprobs_idx[idx].clone(),
          self.output_topk_p[idx].clone(),
          self.output_topk_index[idx].clone(),
          self.output_hidden_states[idx].clone(),
          self.bootstrap_room[idx].clone(),
      )
```

评论区精华

Review 中 gemini-code-assist[bot] 指出 NSATokenToKVPool.get_cpu_copy 返回字典类型，而基类和其他子类返回 List[List[Tensor]]，这种不一致可能在未来造成问题。建议考虑在基类中引入统一的多缓冲区离载接口。该问题未在 PR 中进一步讨论，但 PR 最终被 ShangmingCai 批准合并。

- get_cpu_copy 返回类型与基类不一致 (design): PR 被批准合并，未进一步讨论类型不一致问题。

风险与影响

- 风险：主要风险来自 get_cpu_copy 返回类型与基类不一致，如果未来有代码依赖基类返回格式，可能导致错误。此外，两个文件修改均处于核心路径（disaggregation 和 KV 缓存），任何遗漏都可能影响系统稳定性。当前无配套测试覆盖该场景。
- 影响：对用户：修复了 GLM-5.1 在 PD 部署下 retract-resume 时的输出乱码问题，提升了模型的可用性。对系统：改进了状态管理的正确性，降低了 radix tree 污染的风险。对团队：需要关注 get_cpu_copy 的返回类型一致性，未来可考虑重构基类接口。
- 风险标记：核心路径变更，返回类型不一致，缺少测试覆盖

关联脉络

- 暂无明显关联 PR