PR #22790 完整报告

sgl-project/sglang

Refactor streaming session abort handling

合并时间：2026-04-15 15:13

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/22790

执行摘要

- 一句话：重构流式会话中止处理，集中化清理逻辑并添加并发控制，修复内存泄漏和状态损坏。
- 推荐动作：建议工程师精读 session_controller.py 和 session_aware_cache.py，重点关注 finish_req 和 abort_req 的设计如何解耦令牌历史和 KV 状态，以及 match_prefix 中会话分离的逻辑。这些决策对于理解流式会话的健壮性改进至关重要。

功能与动机

根据 PR body 描述，流式会话缺乏适当的中止处理，中止可能导致会话状态损坏、KV 内存泄漏或服务器崩溃。此 PR 旨在修复五个独立问题，并确保会话在中断后能正确恢复，提升系统健壮性。

实现拆解

1. 入口层增强：在 python/sglang/srt/managers/session_controller.py 的 Session 类中添加 _inflight 布尔标志，用于防止流式会话上的并发请求；修改 create_req 方法，在流式会话中检查 _inflight，若已存在活动请求则预中止新请求。
2. 状态管理方法：新增 finish_req 和 abort_req 方法。finish_req 在请求成功完成时更新 req_nodes 字典，将上一个请求的会话置空并记录当前请求；abort_req 仅清除 _inflight 标志，保持 req_nodes 不变以保留历史。
3. 缓存层逻辑重构：在 python/sglang/srt/mem_cache/session_aware_cache.py 中，修改 match_prefix 方法，对于预中止请求（req.to_finish 非空），调用 req.session.abort_req() 并分离会话；修改 cache_finished_req 方法，对于处理中中止，创建临时会话槽并通过 release_session 清空所有 KV，同时调用新增的 _mark_kv_freed 静态方法设置簿记标志。
4. 测试配套更新：在 test/registered/sessions/test_streaming_session.py 和 test/registered/unit/mem_cache/test_streaming_session_unit.py 中添加新的单元和集成测试，如 test_nth_mid_abort_recovery 和 test_preabort_detaches_session_and_preserves_slot，覆盖各种中止场景，并调整测试常量以确保在严格内存检查下通过。
5. 配置与常量调整：根据提交历史，微调测试中的生成长度和令牌数以匹配新逻辑，避免上下文溢出。

关键文件：

- python/sglang/srt/managers/session_controller.py（模块 会话管理；类别 source；类型 entrypoint；符号 finish_req, abort_req, _all_requests_finished）：入口点，定义 Session 类并新增 _inflight 标志、finish_req 和 abort_req 方法，是并发控制和状态管理的核心。
- python/sglang/srt/mem_cache/session_aware_cache.py（模块 缓存层；类别 source；类型 core-logic；符号 _mark_kv_freed）：核心逻辑，修改 match_prefix 和 cache_finished_req 以处理流式会话中止，实现集中化清理和 KV 释放。
- test/registered/sessions/test_streaming_session.py（模块 会话测试；类别 test；类型 test-coverage；符号 test_nth_mid_abort_recovery, do_generate, test_first_mid_abort_recovery, test_preabort_recovery）：集成测试文件，新增多个中止恢复测试函数，验证流式会话在各种中止场景下的行为。
- test/registered/unit/mem_cache/test_streaming_session_unit.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_streaming_release_kv_cache_defers_tail_free, test_preabort_detaches_session_and_preserves_slot, test_first_mid_abort_nukes_ephemeral_slot）：单元测试文件，重构现有测试并新增针对预中止和临时槽清理的测试，验证缓存层逻辑。

关键符号：Session.finish_req, Session.abort_req, SessionAwareCache._mark_kv_freed, SessionAwareCache.match_prefix, SessionAwareCache.cache_finished_req

关键源码片段

python/sglang/srt/managers/session_controller.py

入口点，定义 Session 类并新增 _inflight 标志、finish_req 和 abort_req 方法，是并发控制和状态管理的核心。

```
def finish_req(self, req):
    """更新 req_nodes 当流式请求成功完成时。"""
    self._inflight = False # 清除进行中标志，允许新请求
    if self.req_nodes:
        [prev_node] = self.req_nodes.values()
        prev_node.req.session = None # 断开上一个请求的会话关联，避免内存泄漏
        self.req_nodes.clear() # 清空历史记录，仅保留当前成功请求
    self.req_nodes[req.rid] = SessionReqNode(req) # 记录新请求为会话历史，供后续恢复使用


def abort_req(self):
    """中止时清除 inflight 标志，req_nodes 保持不变以保留历史。"""
    self._inflight = False # 仅重置标志，不修改 req_nodes，确保会话历史不丢失
```

python/sglang/srt/mem_cache/session_aware_cache.py

核心逻辑，修改 match_prefix 和 cache_finished_req 以处理流式会话中止，实现集中化清理和 KV 释放。

```
def cache_finished_req(self, req: Req, is_insert: bool = True, **kwargs):
    if not _is_streaming(req):
        return self.inner.cache_finished_req(req, is_insert=is_insert, **kwargs)
```

```
from sglang.srt.managers.schedule_batch import FINISH_ABORT

session_id = req.session.session_id
slot = self.slots.get(session_id)
is_first = slot is None

if isinstance(req.finished_reason, FINISH_ABORT):
    # 处理中中止：清空所有 KV 并通过 release_session 删除槽
    if slot is None:
        # 首次请求中止：创建临时槽以处理清理，确保锁正确释放
        slot = SessionSlot(
            req_pool_idx=req.req_pool_idx,
            kv_allocated_len=req.kv_allocated_len,
            last_node=req.last_node,
            cache_protected_len=req.cache_protected_len,
            swa_uuid_for_lock=req.swa_uuid_for_lock,
        )
        self.slots[session_id] = slot
    slot.kv_allocated_len = max(slot.kv_allocated_len, req.kv_allocated_len)
    self.release_session(session_id) # 集中化清理路径，释放KV内存和池槽
    req.req_pool_idx = None
    req.session.abort_req() # 调用会话的中止方法，清除 inflight 标志
    self._mark_kv_freed(req) # 设置簿记标志，避免忙检查重复计数
    return
# ... 正常完成处理，调用 finish_req 更新会话历史
```

评论区精华

review 中仅有一个来自 gemini-code-assist[bot] 的评论，建议将 scheduler.py 中重复的会话回滚逻辑提取为助手方法以提高代码清晰度。该建议未在本次 PR 中实施，可能留待后续优化。

- 提取重复回滚逻辑 (design): 未在本次 PR 中实施，可能留待后续优化。

风险与影响

- 风险：技术风险包括：1) 回归风险，核心会话管理逻辑变更可能影响现有流式会话功能；2) 并发控制新增 _inflight 标志，若实现不当可能导致死锁或竞争条件；3) release_session 的集中化清理虽简化逻辑，但需确保所有中止路径都正确调用，否则可能遗漏内存释放；4) 测试覆盖全面，但极端情况如高并发中止场景可能未充分验证。
- 影响：影响范围：1) 用户：流式会话在中止后能正确恢复历史，提升体验和可靠性；2) 系统：减少内存泄漏和状态损坏风险，增强服务器稳定性；3) 团队：新设计引入 _inflight 和 finish_req/abort_req 模式，需工程师熟悉以维护相关代码。影响程度为中度，主要涉及流式会话模块，不改变外部 API。
- 风险标记：核心路径变更，并发控制新增，测试覆盖全面但仍需验证

关联脉络

- PR #22862 Streaming session: fix retract tail leak via _free_tail: 同样修复流式会话的内存泄漏问题，涉及 KV 缓存管理。
- PR #22753 Fix streaming session busy-check double-counting via active_pool_idxs: 修复流式会话的统计双计数，与本 PR 的并发控制改进相关。
- PR #22755 Rename _alive_streaming_session_count; use _is_streaming helper: 重构流式会话相关辅助函数，代码风格一致性。