PR #32700 完整报告

sgl-project/sglang

[Scheduler] Fix to restrict the SWA chunk-cap escape hatch to true head-of-line livelock

合并时间：2026-08-06 22:09

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/32700

执行摘要

- 一句话：SWA 逃生舱仅限真活锁，消除 transient 压力下的 retraction 风暴
- 推荐动作：值得精读。该 PR 是一个教科书式的"逃生舱过度触发"修复：用最小改动（新增一个判定函数 + 两处门控）恢复设计意图，并以三场景表格清晰论证边界，是调度策略类 bugfix 的典范。关注点：_swa_req_never_fits() 复用 _swa_budget_for_req 与 _swa_chunk_cap 的预算口径，保持了单一数据源；测试用 SimpleNamespace 模拟 tree_cache 与 token_to_kv_pool_allocator，验证了纯逻辑可测性。建议后续补充边界（等于 size_swa）与真实 allocator 的集成测试。

功能与动机

PR body 指出 #31681 的逃生舱本意是解决队头活锁（请求永远装不进 SWA 池），但实际在普通 transient SWA 压力下也会触发，"the scheduler keeps admitting prefill into the headroom that running decodes need, collapsing the SWA evictable cushion"，在高并发 hybrid-SWA 模型上观测到"roughly half the running requests got retracted and 40%+ of prefill compute was re-prefill rework"。PR 用三场景表格明确了修复边界：$B < R$ 正常准入、$R \leqslant B < C$ 应等待、$B \geqslant C$ 才允许逃生舱。

实现拆解

1. 变更入口：python/sglang/srt/managers/schedule_policy.py 中 PrefillAdder.add_one_req() 的 hybrid-SWA 分支。该分支在 swa_needed >= self.rem_swa_tokens 时此前会直接触发 _swa_chunk_cap 收缩 chunk 准入。
2. 新增判定函数：新增 _swa_req_never_fits(extend_input_len, max_new_tokens, swa_host_hit_length)，内部用 self._swa_budget_for_req(...) >= self.token_to_kv_pool_allocator.size_swa 判断请求预算是否超过整个 SWA 池容量，这是"无论池如何排空都装不下"的充要条件。
3. 门控接入：在 add_one_req() 的锁前与锁后两处 swa_needed >= self.rem_swa_tokens 分支内，先调用 _swa_req_never_fits()；返回 False 时直接 return AddReqResult.NO_TOKEN 等待，只有返回 True 才继续走 _swa_chunk_cap 收缩逻辑。两处保持对称，避免锁竞争导致的状态漂移。
4. 测试配套：test/manual/test_schedule_policy.py 新增 TestSwaChunkCapHatch 三个纯逻辑用例（transient 压力等待、预算超池触发、随 size_swa 翻转），通过 PrefillAdder.__new__ 只构造必要字段，无需 KV 池与 GPU；

test/registered/unit/managers/test_prefill_adder.py 给 mock token allocator 补上默认 size_swa = 1_000_000，否则 _swa_req_never_fits 访问 MagicMock 的 size_swa 会 TypeError。

5. 行为影响：无参数与配置变更，属调度准入行为修复；默认行为从“收缩准入”变为“transient 压力下等待”，从根本上消除 retraction 风暴。

关键文件：

- python/sglang/srt/managers/schedule_policy.py（模块 调度器；类别 source；类型 core-logic；符号 _swa_req_never_fits）：调度准入核心逻辑所在：新增 _swa_req_never_fits() 门控，并在 add_one_req() 锁前锁后两处收紧 _swa_chunk_cap 逃生舱触发条件，是本 PR 的修复主体。
- test/manual/test_schedule_policy.py（模块 调度器；类别 test；类型 test-coverage；符号 _swa_adder, TestSwaChunkCapHatch, test_transient_pressure_request_waits, test_request_larger_than_whole_pool_takes_hatch）：新增 TestSwaChunkCapHatch 测试类与 _swa_adder 轻量构造器，直接覆盖本次修复的三个关键行为：transient 压力等待、超池触发逃生舱、随池容量翻转判定。
- test/registered/unit/managers/test_prefill_adder.py（模块 准入控制；类别 test；类型 test-coverage）：为 mock token allocator 增加默认 size_swa = 1_000_000 属性，避免 _swa_req_never_fits 读取 MagicMock 属性时抛 TypeError，保证既有准入测试继续通过。

关键符号：_swa_req_never_fits, add_one_req, _swa_chunk_cap, _swa_adder, TestSwaChunkCapHatch

关键源码片段

test/manual/test_schedule_policy.py

新增 TestSwaChunkCapHatch 测试类与 _swa_adder 轻量构造器，直接覆盖本次修复的三个关键行为：transient 压力等待、超池触发逃生舱、随池容量翻转判定。

```
def _swa_adder(size_swa, sliding_window, page_size=16, rem_chunk_tokens=512):
    """返回一个只携带 `_swa_req_never_fits` 所需字段的 `PrefillAdder`。

    用 `__new__` 绕过完整初始化，不依赖真实 KV 池与 GPU，
    让逃生舱门控作为纯逻辑被单独测试。
    """
    adder = PrefillAdder.__new__(PrefillAdder)
    adder.page_size = page_size
    adder.rem_chunk_tokens = rem_chunk_tokens
    adder.tree_cache = SimpleNamespace(sliding_window_size=sliding_window)
    adder.token_to_kv_pool_allocator = SimpleNamespace(size_swa=size_swa)
    return adder


class TestSwaChunkCapHatch(CustomTestCase):
    """逃生舱必须只在真正队头活锁（预算 > 整个池）时触发。"""

    def test_transient_pressure_request_waits(self):
```

```
        # 预算远小于池容量：decode 释放窗口后必然能容纳，必须等待而非收缩准入
        adder = _swa_adder(size_swa=1024, sliding_window=128)
        self.assertFalse(
            adder._swa_req_never_fits(extend_input_len=256, max_new_tokens=64)
        )

    def test_request_larger_than_whole_pool_takes_hatch(self):
        # host-hit load-back 一次把预算推到池容量之上：无论池如何排空都装不下
        adder = _swa_adder(size_swa=1024, sliding_window=128)
        self.assertTrue(
            adder._swa_req_never_fits(
                extend_input_len=256, max_new_tokens=64, swa_host_hit_length=4096
            )
        )

    def test_decision_is_gated_by_pool_capacity(self):
        # 同一请求在不同池容量下结论翻转，验证比较对象是 size_swa 而非错误 accessor
        req = dict(extend_input_len=256, max_new_tokens=64, swa_host_hit_length=600)
        self.assertTrue(
            _swa_adder(size_swa=512, sliding_window=128)._swa_req_never_fits(**req)
        )
        self.assertFalse(
            _swa_adder(size_swa=4096, sliding_window=128)._swa_req_never_fits(**req)
        )
```

评论区精华

该 PR 没有形成实质 review 评论线程，reviewer ispobock 直接 APPROVED。核心设计论证全部在 PR body 的三场景表格中完成：明确区分了“当前能装下”（正常准入）、“当前装不下但排空后可装下”（必须等待）与“永远装不下”（逃生舱唯一适用）三种情形。Issue 评论区只有 gemini-code-assist[bot] 的停止通知与 pengwu22、ispobock 触发的 CI 命令，无技术讨论。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 核心调度路径变更：add_one_req() 是调度器准入主路径，新增门控影响所有 hybrid-SWA 模型；锁后分支同样加入 NO_TOKEN 返回，在锁竞争下行为更保守，但需关注极端压力下等待队列被 transient 请求长期占用的可能性。
  2. 边界条件：>= 比较意味着预算恰好等于 size_swa 时也走逃生舱；此时 _swa_chunk_cap 会把 chunk 收缩到当前 rem_swa_tokens 内，逻辑上安全，但边界场景缺少专门测试。
  3. 测试覆盖局限：TestSwaChunkCapHatch 是 CPU 纯逻辑测试，通过 __new__ 构造 adder，未覆盖真实 allocator 行为与多 rank 场景；test_prefill_adder.py 只补了 mock 属性，未新增准入断言。

4. 性能回归风险：每次触发分支多一次 _swa_budget_for_req 调用，属 O(1) 计算，风险极低。 - 影响：对用户与系统：hybrid-SWA 模型在高并发（约 0.9 池使用率）下 retraction/re-prefill 风暴被消除，decode 吞吐 +36%、p50 延迟 -41%，等效提升单机服务能力；巨大 prompt 或大 host-load-back 请求仍可通过逃生舱避免队头活锁。对团队：调度器核心准入逻辑行为变化，所有依赖 SWA 池的模型（如 sliding-window attention 系）都应回归验证；两个测试文件的新增模式（__new__ 构造纯逻辑 adder）为后续调度策略单测提供低成本模板。 - 风险标记：核心调度路径变更，回归修复（#31681 引入），测试仅 CPU 纯逻辑，边界条件未覆盖

关联脉络

- PR #31681 [Scheduler] Introduce SWA chunk-cap escape hatch (referenced in PR body): PR body 明确说明本修复针对 #31681 引入的 _swa_chunk_cap 逃生舱过度触发回归；#31681 不在本次提供的近期历史列表中，标题未知。