PR #33794 完整报告

sgl-project/sglang

Fix paged SWA retraction resume accounting

合并时间：2026-08-07 07:32

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33794

执行摘要

- 一句话：修复 SWA 预分配预算未按物理页对齐导致恢复记账错误
- 推荐动作：值得精读。关注两点：一是 _prealloc_required_tokens 与 pop_preallocated 中预算更新的一致性，理解“token 需求须与物理页计费对齐”的原则；二是回归测试构造物理页预算精确断言的写法，可作为同类内存预算修复的测试范本。

功能与动机

PR body 明确说明要 "align full and sliding-window preallocation requirements to physical allocator pages"，并 "keep the remaining SWA budget consistent when multiple retracted requests resume"。根因是预算按 token 计数、分配器按整页计费，两者不一致导致 retracted 请求恢复时预算判断失真。

实现拆解

1. 核心修复（python/sglang/srt/disaggregation/decode.py）：在 _prealloc_required_tokens 中读取 token_to_kv_pool_allocator.page_size，当 page_size > 1 时对 full_len 与 swa_len 用 ceil_align 向上取整，使预算需求与物理分配器的整页计费一致；同时保留 disable_radix_cache 时 swa_reserved 置 0 的原有分支。
2. 注释同步（同文件 pop_preallocated）：将 SWA 预算递减处的注释改为“按页对齐要求直接递减”，明确 swa_required 已按页对齐，避免后续维护误解。
3. 回归测试（test/registered/unit/disaggregation/test_decode_queue_cleanup.py）：新增 test_paged_swa_retraction_resume_uses_physical_page_budget，构造 page_size=128、fill_len=574、每请求物理 5 页、总预算 18 页的 4 个撤回请求，断言只恢复 3 个、剩余 3 页、_pre_alloc 调用 3 次，精确覆盖“物理页预算”边界。
4. 配套改动：仅新增导入 ceil_align，无配置、schema 或部署变更。

关键文件：

- python/sglang/srt/disaggregation/decode.py（模块 解码队列；类别 source；类型 core-logic；符号 _prealloc_required_tokens, _prealloc_kv_lens, pop_preallocated）：核心修复文件：在预分配 token 需求计算中加入物理页对齐，使预算与分配器行为一致，并同步更新 SWA 预算递减注释。
- test/registered/unit/disaggregation/test_decode_queue_cleanup.py（模块 解码队列；类别 test；类型 test-coverage；符号 test_paged_swa_retraction_resume_uses_physical_

page_budget, pre_alloc）：新增回归测试，精确覆盖物理页预算边界，验证多个撤回请求在近容量时只按整页预算恢复。

关键符号：_prealloc_required_tokens, resume_retracted_reqs, test_paged_swa_retraction_resume_uses_physical_page_budget

关键源码片段

python/sglang/srt/disaggregation/decode.py

核心修复文件：在预分配 token 需求计算中加入物理页对齐，使预算与分配器行为一致，并同步更新 SWA 预算递减注释。

```
def _prealloc_required_tokens(self, req: Req) -> Tuple[int, int]:
    # 先取得未对齐的全长与 SWA 尾部长度需求
    full_len, swa_len = self._prealloc_kv_lens(req)

    # 分配器按整页收费，这里将两侧需求统一向上取整到页边界
    page_size = self.token_to_kv_pool_allocator.page_size
    if page_size > 1:
        full_len = ceil_align(full_len, page_size)
        swa_len = ceil_align(swa_len, page_size)

    # radix cache 禁用时不再预留 SWA token，避免预算被额外占用
    swa_reserved = self.num_reserved_decode_tokens
    if self.scheduler.server_args.disable_radix_cache:
        swa_reserved = 0

    # 返回（全量预算，SWA 预算）供上层判断是否允许恢复撤回请求
    return (
        full_len + self.num_reserved_decode_tokens,
        swa_len + swa_reserved,
    )
```

test/registered/unit/disaggregation/test_decode_queue_cleanup.py

新增回归测试，精确覆盖物理页预算边界，验证多个撤回请求在近容量时只按整页预算恢复。

```
def test_paged_swa_retraction_resume_uses_physical_page_budget(self):
    page_size = 128
    fill_len = 574
    # 每个请求按页对齐后占用 5 页（ceil(574 / 128) = 5）
    physical_tokens_per_req = 5 * page_size
    # 可用物理页预算为 18 页，恰好只能容纳 3 个请求
    physical_available = 18 * page_size

    # 构造 4 个已撤回请求，origin_input_ids 长度模拟真实 prefill 长度
    reqs = [
        SimpleNamespace(
            rid=f"req-{i}",
            origin_input_ids=[0] * fill_len,
```

```
            output_ids=[],
            is_retracted=True,
            load_kv_cache=MagicMock(),
        )
        for i in range(4)
    ]

    queue = DecodePreallocQueue.__new__(DecodePreallocQueue)
    queue.retracted_queue = reqs.copy()
    queue.num_reserved_decode_tokens = 0
    queue.req_to_token_pool = SimpleNamespace(available_size=lambda: len(reqs))
    queue.token_to_kv_pool_allocator = SimpleNamespace(page_size=page_size)
    queue.scheduler = SimpleNamespace(
        sliding_window_size=2047,
        server_args=SimpleNamespace(disable_radix_cache=True),
    )
    queue._uses_swa_tail_prealloc = MagicMock(return_value=True)
    queue._swa_aware_allocatable_token_budgets = MagicMock(
        return_value=(physical_available, physical_available)
    )

    # 模拟物理分配：每恢复一个请求扣掉整页对齐的预算
    def pre_alloc(_req):
        nonlocal physical_available
        self.assertGreaterEqual(physical_available, physical_tokens_per_req)
        physical_available -= physical_tokens_per_req

    queue._pre_alloc = MagicMock(side_effect=pre_alloc)

    resumed = queue.resume_retracted_reqs()

    # 18 页预算只能恢复前 3 个请求，剩余 1 个留在 retracted_queue
    self.assertEqual(resumed, reqs[:3])
    self.assertEqual(queue.retracted_queue, reqs[3:])
    # 恢复 3 次后恰好剩余 3 页，验证按整页扣减且无透支
    self.assertEqual(physical_available, 3 * page_size)
    self.assertEqual(queue._pre_alloc.call_count, 3)
```

评论区精华

PR 无实质 review 讨论；4 条评论均为维护者触发 CI 的 /tag-and-rerun-ci，merrymercy 直接 APPROVED。技术上未出现公开争议，核心修复点（页对齐预算）由提交信息与 PR body 清晰说明。

- 无实质 review 讨论 (other): 无未解决的技术疑虑，直接合并。

风险与影响

- 风险：

1. _prealloc_required_tokens 向上取整后预算更保守，近容量时恢复的请求数可能比修复前略少，但这是修正过度订阅的正确行为。
2. 依赖 page_size <= 1 时保持原逻辑（跳过对齐），若未来分配器页大小动态变化，需同步检查此分支。
3. swa_allocatable_tokens -= swa_required 的正确性依赖 swa_required 已按页对齐，二者耦合在一处，后续若调整 _swa_aware_allocatable_token_budgets 或分配器计费方式需同步。
4. 当前仅有 CPU 单元测试，缺少真实多卡 PD 场景的集成测试，预算边界差异在端到端下未被直接验证。 - 影响：影响范围限定在“启用 SWA tail prealloc 且开启 PD 解耦 decode 侧”的场景（如 Qwen3、DeepSeek 等滑动窗口模型），修复后撤回请求恢复的预算判断与物理分配一致，避免恢复失败或内存透支。不启用 SWA 或 page_size=1 的配置不受影响。对团队而言，该修复补齐了 disaggregation 路径中一个隐蔽的正确性缺口，属于低风险高收益的 bugfix。 - 风险标记：预算对齐更保守，缺少集成测试，仅影响 SWA tail prealloc 路径

关联脉络

- PR #32700 [Scheduler] Fix to restrict the SWA chunk-cap escape hatch to true head-of-line livelock: 同为 SWA 调度预算与正确性修复，涉及 SWA 容量上限与调度策略，与本次预算记账修复同属 SWA 功能线。
- PR #30545 [Disagg][StagingBuffer][2/2] Support radix cache: 同为 disaggregation 模块的 KV 缓存与预算管理改动，涉及 decode 侧缓存与预算调整的关联路径。