PR #35286 完整报告

sgl-project/sglang

[Fix] Assert the page-aligned SWA evict floor at PD decode prealloc

合并时间：2026-08-19 04:32

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35286

执行摘要

- 一句话：PD 解码预分配新增 SWA 驱逐页对齐断言
- 推荐动作：值得快速阅读。本 PR 的代码量只有 4 行，价值在于把跨模块的页对齐不变量显式锚定在数据写入点——分配成功之后、写入 req.kv.swa_evicted_seqlen 之前校验。值得关注的设计决策是断言位置的选择：不放在 _swa_tail_len（约束来源处），也不放在 free_swa_segment（消费处），而是放在唯一写入口，使所有未来调用方都经过同一个校验关卡。对追踪 #35223 的读者来说，这是理解 I2（SWA liveness boundary invariant）的最小样例。

功能与动机

追踪 issue #35223 的目标是把 KV 分配与释放的 source of truth 从设备端迁移到主机端：分配在主机端按整页规划，每个请求记录页对齐的 kv_allocated_len，释放时天然拿到整页，不再需要 torch.unique、liveness 过滤等设备读回。SWA 一侧的 swa_evicted_seqlen 也必须页对齐。PR body 明确说明：_swa_tail_len already floors its window start to a page, so fill_len - swa_tail_len is a page multiple. Assert it here so any future caller bypassing _swa_tail_len fails loudly rather than silently poisoning the page-aligned liveness invariant (free_swa_segment's contract).

实现拆解

1. 变更入口：仅修改 python/sglang/srt/disaggregation/decode.py 中的 alloc_for_decode_prealloc_hisparse 函数，这是 PD（prefill-decode 分离）架构下 hisparse 路由的解码预分配入口。
2. 核心改造：在 uses_swa_tail 分支里，把原先直接赋给 req.kv.swa_evicted_seqlen 的表达式 fill_len - swa_tail_len 先存入局部变量 swa_evicted_seqlen，随后加入断言 swa_evicted_seqlen >= 0 and swa_evicted_seqlen % allocator.page_size == 0，再写入 req.kv.swa_evicted_seqlen。这样不变量在数据落盘前被校验，遵循“尽早失败”的防御原则。
3. 原因：_swa_tail_len 会把窗口起点向下取整到页，因此理论上 fill_len - swa_tail_len 必然是页的倍数；但该不变量只存在于调用链内部，缺乏本地约束。任何未来绕过 _swa_tail_len 的调用者都可能悄悄写入非页对齐值，进而污染 free_swa_segment 依赖的 liveness 记账，造成难以定位的 KV 损坏。
4. 测试与 CI 配套：本 PR 未新增单元测试文件；作者在 issue 评论中 rerun 了 test_disaggregation_dsv4.py 与 test_disaggregation_basic.py，分别在 8-gpu-h200 与

2-gpu-h100 上通过，随后 /tag-and-rerun-ci 触发完整 CI。

关键文件：

- python/sglang/srt/disaggregation/decode.py（模块 解码预分配；类别 source；类型 core-logic；符号 alloc_for_decode_prealloc_hisparse）：这是 PD 分离架构下 hisparse 解码预分配的唯一入口，直接承载 swa_evicted_seqlen 的页对齐不变量。本 PR 在这里把隐式约定改为显式断言，是追踪 issue #35223 的 I2 步骤的核心实现点。

关键符号：alloc_for_decode_prealloc_hisparse

关键源码片段

python/sglang/srt/disaggregation/decode.py

这是 PD 分离架构下 hisparse 解码预分配的唯一入口，直接承载 swa_evicted_seqlen 的页对齐不变量。本 PR 在这里把隐式约定改为显式断言，是追踪 issue #35223 的 I2 步骤的核心实现点。

```
def alloc_for_decode_prealloc_hisparse(
    allocator: BaseTokenToKVPoolAllocator,
    *,
    req: Req,
    fill_len: int,
    uses_swa_tail: bool,
    swa_tail_len: int,
) -> torch.Tensor:
    # PD decode 场景的 hisparse 预分配入口
    # req.kv 是主机端持有的 KV 生命周期容器（来自 #29427 系列），
    # kv_allocated_len 是页对齐后的分配长度
    if req.kv is None:
        req.kv = ReqKvInfo(kv_allocated_len=fill_len, swa_evicted_seqlen=0)
    else:
        req.kv.kv_allocated_len = fill_len

    device = allocator.device
    prefix_lens = torch.tensor([0], dtype=torch.int64, device=device)
    prefix_lens_cpu = torch.tensor([0], dtype=torch.int64)
    seq_lens = torch.tensor([fill_len], dtype=torch.int64, device=device)
    seq_lens_cpu = torch.tensor([fill_len], dtype=torch.int64)
    last_loc = torch.tensor([-1], dtype=torch.int64, device=device)

    if uses_swa_tail:
        # SWA tail 模式：alloc_extend_swa_tail 只按窗口尾部页分配，
        # 窗口起点已被 _swa_tail_len 向下取整到页边界
        kv_loc = allocator.alloc_extend_swa_tail(
            prefix_lens=prefix_lens,
            prefix_lens_cpu=prefix_lens_cpu,
            seq_lens=seq_lens,
            seq_lens_cpu=seq_lens_cpu,
            last_loc=last_loc,
```

```
            extend_num_tokens=fill_len,
            swa_tail_len=swa_tail_len,
        )
        # fill_len - swa_tail_len 即被驱逐的序列长度,
        # 必须是 page_size 的整数倍，否则会破坏 free_swa_segment
        # 依赖的页对齐 liveness 契约，导致主机端记账静默错位
        swa_evicted_seqlen = fill_len - swa_tail_len
        assert (
            swa_evicted_seqlen >= 0
            and swa_evicted_seqlen % allocator.page_size == 0
        )
        req.kv.swa_evicted_seqlen = swa_evicted_seqlen
    else:
        # 非 SWA 分支：整段逻辑分配，无驱逐下界需要记账
        kv_loc = allocator.alloc_logical_only(
            prefix_lens=prefix_lens,
            prefix_lens_cpu=prefix_lens_cpu,
            seq_lens=seq_lens,
            seq_lens_cpu=seq_lens_cpu,
            last_loc=last_loc,
            extend_num_tokens=fill_len,
        )
    return kv_loc
```

评论区精华

本 PR 没有收到任何 review 评论（review_comments_count = 0），核心讨论集中在 issue 评论区。作者 hnyls2002 主动 rerun 了 disaggregation 相关测试：/rerun-test test/registered/disaggregation/test_disaggregation_dsv4.py test/registered/disaggregation/test_disaggregation_basic.py。github-actions bot 随后报告两个测试全部通过（8-gpu-h200 与 2-gpu-h100）。作者还发起了 /tag-and-rerun-ci 触发全量 CI。此外 PR body 本身说明了设计权衡：与其静默接受非页对齐值，不如在入口处大声失败，保护后续 free_swa_segment 的 liveness 契约。

- CI 重跑 disaggregation 相关测试 (testing): disaggregation 相关测试全部通过，证明该防御性断言未破坏现有 PD 分离路径的行为；全量 CI 已触发且通过。

风险与影响

- 风险：
  - 断言覆盖不全：本次只在 hisparse 分支（alloc_for_decode_prealloc_hisparse）加断言，另一个入口 alloc_for_decode_prealloc 以及 EAGLE/DFLASH 相关路径尚未覆盖，后续可在 #35382 的“共享 decode alloc lens”中统一收敛。
  - 对 _swa_tail_len 的隐式依赖：断言本身并不校验页对齐的来源，它依赖 _swa_tail_len 已向下取整到页这一前提。若未来 swa_tail_len 语义变化（如允许部分页 tail），此断言可能误伤合法调用，取整逻辑与断言需保持同步演进。

- 回归风险低：断言为纯防御性，正常路径数值不变；CI 上 3 个 disaggregation 相关测试 job 均通过。
- 缺少单元测试：没有新增的针对性单测，只有集成测试覆盖。
- 影响：影响范围：主要影响 PD 分离模式下使用 hisparse 与 SWA tail 的解码预分配路径，对未启用 SWA 的常规路径无行为影响；由于断言不改变任何数值，正常运行无感知。对团队而言，这是 host allocator 迁移（#35223）的关键不变量锚点，把跨模块的页对齐约定下沉到代码入口，显著降低后续重构的排查与回归成本。对用户无直接可见影响，但提升了系统在异常调用下的可诊断性。
- 风险标记：核心路径变更，缺少单元测试覆盖，依赖 _swa_tail_len 页对齐前提

关联脉络

- PR #35265 [Spec] Page-align the DFLASH decode KV reservation: 同一追踪 issue #35223 下的姊妹 PR，同样为 kv_allocated_len 的页对齐不变量服务；本 PR 的断言为 DFLASH 路径的预留对齐提供了前置约束。
- PR #35382 share the page-aligned decode alloc lens between EAGLE and DFLASH: issue #35223 中明确提及的在途 PR，目标是把 decode 分配的页对齐视图统一到单一 writer；本 PR 的断言是其前置不变量。
- PR #35049 [PD] Deferred decode-side KV release for aborts mid-transfer: 同样修改了 python/sglang/srt/disaggregation/decode.py，是 PD 分离路径中 KV 生命周期治理的核心热点，与本次预分配层修改在代码相邻区域协同。
- PR #32701 free KV pages by segment in the paged allocator without a device sync: host allocator 路线的前期落地工作，本 PR 的页对齐断言为该路线后续铺路，属于同一功能演进脉络。