PR #36834 完整报告

sgl-project/sglang

[HiCache] buffer mode: decide staged-fetch fate against the live tree

合并时间：2026-08-29 16:51

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36834

执行摘要

- 一句话：HiCache buffer 模式改为按实时树裁决预取去留
- 推荐动作：值得精读。核心价值在于解决异步预取与实时缓存树状态不一致的经典问题：不信任会过期的 node id，而是在决策点对 live tree 重新匹配，且把决策前置到 IO commit 之前以保持撤销语义。消费端 trim 而非全有全无的设计也值得借鉴。建议读者重点看 pipeline.py 中 try_lock_anchor 的四态返回与 staged_splice_tokens 的三种不可用判定，以及 unified_radix_cache.py 的 IO commit 决策顺序；如后续要扩展 HiCache，需注意重匹配开销与取消后重试的竞态。

功能与动机

PR body 明确指出问题根源：Buffer-mode staged prefetches were launched on a blind gamble——try_lock_anchor 解引用携带的 anchor node id 会因 split 与 eviction 而过期，lookup miss 时静默跳过 pin 照常 fetch；生产审计发现数百个 span 从 L3 读回后在 consume 时因 splice base 不再存在而被丢弃，且这些 fetch 全部是未锁定下发的。同时 consumption 采用全有全无策略：device prefix 一旦增长超过 hold 的 matched_len，整个 staged span 即被丢弃，即使其尾部仍可拼接。

实现拆解

变更入口是 buffer-only 模式的预取下发与消费链路，共 4 个文件协同调整，按数据流顺序拆解如下：

1. 命名空间修正（unified_radix_cache.py + pipeline.py）：存储预取 key 的命名空间改从请求（extra_key + cache_salt）获取，而非 anchor node（root anchor 无命名空间会串 key）；set_prefix_ctx 的记录结构从 list[int] 扩展为 (prefix_tokens, extra_key, cache_salt) 三元组，为重匹配重建 key 提供依据。这是后续 live tree 重匹配的前提。
2. 锚点锁定改为 live tree 重匹配（pipeline.py 的 try_lock_anchor）：不再接收 node id 参数，而是在请求命名空间下按 prefix 对当前树做 O(prefix path) 匹配，返回 locked / no_anchor / cap_skip / anchor_lost 四种结果；prefetch_from_storage 下发时立即尝试一次锁定，IO commit 阶段（_try_alloc_storage_hit 内）再次调用作为最终裁决。
3. IO commit 决策前置（unified_radix_cache.py）：在 bounce alloc 之前检查锁定结果，anchor_lost 时计入 declined_anchor_lost 统计、把 req_id 重新加入 _storage_prefetch_missed_rids 触发 paced retry（span 仍在 L3，可从缩短后的匹配重

新取回）并取消本次取回；决策放在 alloc 前使取消保持纯 revoke 语义、已 park 的请求保留 pin。新增 declined_device_covered 统计覆盖 tree 已持有整个 span 的情形。

4. 消费端 trim 替代整体丢弃（pipeline.py 的 staged_splice_tokens / plan_staged_splice）：新增 staged_splice_tokens 计算 live device prefix 之后可拼接的尾部长度，处理三种不可用情形（prefix 缩到 span 之下、span 已完全 device-resident、trim 会切入 aux 尾部窗口——aux pool 只能整段拼接）；plan_staged_splice 取代 staged_prefetch_tokens，返回 (kv, swa) 双路 token 数并释放不可用 hold，消费时通过 trim_tokens 裁剪增长的 prefix，ack 时释放整个 bounce（含已裁剪头部）。
5. 调度计费对齐与测试配套（scheduler.py + 测试文件）：_get_new_batch_prefill_raw 中改用 plan_staged_splice(req.rid, len(req.prefix_indices)) 按 admission 同一 live prefix 计费，避免批分配 OOM 或占用虚高；测试新增 test_buffer_only_load_back_trims_head_published_by_sibling（sibling 发布 head 后消费 trim 回归）、test_buffer_only_plan_frees_covered_hold、test_buffer_only_hit_commit_cancels_device_covered_fetch 与 TestAnchorLockOutcomePolicy 四组用例，并在最终提交将新测试钉在 page_size=1、sliding_window_size=4 单一配置，防止参数化矩阵撑爆 CI GPU。

关键文件：

- python/sglang/srt/mem_cache/buffer_mode/pipeline.py（模块 缓存管线；类别 source；类型 core-logic；符号 staged_splice_tokens, try_lock_anchor, set_prefix_ctx, staged_span_covered）：本 PR 的核心逻辑所在：try_lock_anchor 改为对 live tree 重匹配并返回四态结果，新增 staged_splice_tokens 计算可拼接尾部，plan_staged_splice 取代 staged_prefetch_tokens 并释放不可用 hold，消费端由整体丢弃改为 trim 后拼接。
- python/sglang/srt/mem_cache/unified_radix_cache.py（模块 统一缓存；类别 source；类型 core-logic；符号 staged_prefetch_tokens, plan_staged_splice）：预取下发时记录请求命名空间并立即尝试锁定，IO commit 阶段在 bounce alloc 前裁决取消 anchor 丢失或被 tree 覆盖的 fetch，新增 declined_anchor_lost / declined_device_covered 结果统计。
- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 core-logic）：prefill batch 组装时改用 plan_staged_splice 按 admission 同一 live prefix 计费，仅对可拼接的 span 尾部计费，并同时取回 kv/swa 双路 token 数。
- test/registered/unit/mem_cache/test_unified_radix_cache_unittest.py（模块 缓存测试；类别 test；类型 test-coverage；符号 test_buffer_only_load_back_trims_head_published_by_sibling, test_buffer_only_plan_frees_covered_hold, test_buffer_only_hit_commit_cancels_device_covered_fetch, TestAnchorLockOutcomePolicy）：新增 4 组 buffer-only 行为回归测试：sibling 发布 head 后消费 trim、覆盖 hold 释放、覆盖 fetch 取消与锚点锁定四态策略，并钉在单一配置防 CI GPU 内存耗尽。

关键符号：try_lock_anchor, staged_splice_tokens, plan_staged_splice, set_prefix_ctx, release_staged_hold

关键源码片段

python/sglang/srt/mem_cache/buffer_mode/pipeline.py

本 PR 的核心逻辑所在：try_lock_anchor 改为对 live tree 重匹配并返回四态结果，新增 staged_splice_tokens 计算可拼接尾部，plan_staged_splice 取代 staged_prefetch_tokens 并释放不可用 hold，消费端由整体丢弃改为 trim 后拼接。

```
# 计算一个已 staged 的预取在实时 device prefix 之后还能拼接多少 token。
# 返回 0 表示该 hold 已不可用，调用方 plan_staged_splice 会据此释放它。
def staged_splice_tokens(f: _StagedPrefetch, device_prefix_len: int) -> int:
    # span 的完整 token 区间：[matched_len, matched_len + num_tokens)
    span_end = f.matched_len + f.num_tokens

    # 情形一 / 二：live prefix 缩到了 span 起点之下（splice base 已失效），
    # 或已推进越过 span 终点（整段已 device-resident），两种都不可拼接。
    if device_prefix_len < f.matched_len or device_prefix_len >= span_end:
        return 0

    # 情形三：prefix 在 span 内增长时，只拼接 prefix 之后的尾部；
    # prefix 越长，可拼接的尾部越短，但永远不丢弃已取回的数据。
    splice_tokens = span_end - device_prefix_len

    # aux 传输（SWA 等）只能整段拼接：若裁剪会切入其尾部窗口，
    # 则放弃本次拼接，保持整段语义（aux pool 不拆零拼接）。
    for t in f.aux_xfers:
        if t.host_indices is not None and t.host_indices.numel() > splice_tokens:
            return 0
    return splice_tokens
```

python/sglang/srt/mem_cache/unified_radix_cache.py

预取下发时记录请求命名空间并立即尝试锁定，IO commit 阶段在 bounce alloc 前裁决取消 anchor 丢失或被 tree 覆盖的 fetch，新增 declined_anchor_lost / declined_device_covered 结果统计。

```
# prefetch_from_storage 的 buffer_mode 分支（head 版本）：
# 锚点锁定改为基于 live tree 的重新匹配，因此必须把请求级命名空间
# 与匹配前缀一起记录，供 try_lock_anchor 重建匹配 key。
if buffer_mode:
    # 记录命名空间三元组（matched prefix + extra_key + cache_salt）。
    # 此前命名空间取自 anchor node，root anchor 无命名空间会串 key。
    self.buffer_pipeline.set_prefix_ctx(
        req_id,
        matched_prefix_tokens,
        extra_key=extra_key,
        cache_salt=cache_salt,
    )
    # 下发时立即尝试锁定：已发现延迟到 IO commit 再锁时锚点常被
    # split/eviction 删除（本 PR 修复的"静默未锁定"来源）。
    # IO commit 的第二次调用才是决定该 fetch 命运的地方。
    self.buffer_pipeline.try_lock_anchor(req_id)
```

评论区精华

该 PR 没有实质性 review 讨论（review 评论为空），仅有一条作者自己的 CI 指令评论 /tag-and-rerun-ci。PR 由作者自行合并，设计取舍主要沉淀在 PR body 与代码注释中，值得注意的点包括：在 IO commit 前裁决以保证取消是纯 revoke；aux 传输（SWA 等）只能整段拼接，trim 不得切入其尾部窗口；anchor_lost 后仍将 span 留在 L3 并触发 paced retry 而非直接丢弃。

- CI 重跑指令 (other): 无技术分歧，作者自行合并 PR。

风险与影响

- 风险：
  1. 热路径开销：try_lock_anchor 现在每次 IO commit 都要做一次 O(prefix path) 的前缀匹配。PR 声称该路径本就是热点，但未提供基准数据，高并发下需关注匹配耗时占比。
  2. 实时状态竞态：plan_staged_splice 在 admission 时基于 live prefix 计算可拼接尾部，但消费发生在稍后，期间 prefix 可能再次变化；trim 逻辑依赖 staged_splice_tokens 判定与最终消费严格一致，若出现竞态可能导致 splice 校验失败或 SWA 窗口语义破坏（aux 需整段拼接）。
  3. 重试依赖假设：anchor_lost 后 arm 的 paced retry 依赖“span 仍 L3-resident”的假设，若存储端已回收数据，重试会再次 miss，形成重复读。
  4. 缺少端到端验证：PR 明确无精度与速度基准，生产收益（消除数百次无效 L3 读）未被量化；测试集中在单一配置（page_size=1、sliding_window_size=4），跨配置布局（不同 page_size、SWA 窗口大小、aux 传输路径）验证有限。
  5. 行为变更面：release_aborted_staged 更名为 release_staged_hold 虽为内部符号，但 plan_staged_splice 会主动释放 hold，改变了调用方对 hold 生命周期的既有假设。 - 影响：影响范围限于 --hicache-host-memory-mode buffer_only 部署（HiCache 的 host RAM 暂存模式），默认 cache 模式与非 HiCache 场景不受影响。对相关部署的收益是消除无效 L3 存储读与消费期整段丢弃，降低存储 IO 压力和显存 / 主机内存暂存占用；调度器计费更贴近实际占用，降低批分配 OOM 风险。对团队而言，该 PR 确立了 staged 预取状态必须对 live tree 重新校验的原则，并扩展了 prefetch 结果统计（declined_anchor_lost / declined_device_covered），为后续 HiCache 演进（如 Mamba 计费、更多 aux pool）提供了基础设施。影响程度中等偏上，集中于 HiCache 功能域。 - 风险标记：核心路径变更，热路径新增前缀匹配，缺少端到端基准，测试仅覆盖单一配置，异步状态竞态

关联脉络

- PR #36958 [misc] Keep req.kv non-optional and key KV ownership on req_pool_idx: 同样改动 unified/radix cache 相关模块与 test/registered/unit/mem_cache/test_unified_radix_cache_unittest.py，属于 KV 生命周期与缓存状态管理的近期演进线，且都与调度器对缓存状态的依赖相关。