PR #27770 完整报告

sgl-project/sglang

[P/D disagg] Decode-side radix cache for SWA hybrid models (unified radix tree)

合并时间：2026-08-21 09:25

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27770

执行摘要

- 一句话：unified radix tree 上为 SWA 混合模型解锁 decode 侧 KV 前缀复用
- 推荐动作：值得精读。它是 SWA 混合模型接入 decode-side radix cache 的里程碑 PR，核心看点是 SWA 双生命周期 KV 与页对齐 radix key 的交互、margin 设计如何直接消除 return_full_match 的过度设计，以及 full/SWA 锁引用计数的精细管理。想理解后续 DSV4/MiMo 扩展（#27831）的读者，应先读透本 PR 的窗口裕量与准入逻辑。

功能与动机

P/D 分离下 decode worker 可以为未完成请求缓存设备端前缀，让后续请求只从 prefill 传输增量，但混合 SWA 模型此前在启动时被直接拒绝。PR body 明确指出：混合 SWA 模型有 full-attention 与 sliding-window 两套生命周期不同的 KV——full KV 可跨请求复用，SWA KV 只对当前窗口有效，因此 decode worker 应复用 full-attention 前缀、只请求新鲜 SWA 窗口。该功能门控到实验性 unified radix tree，并声明 supersede 走默认 SWARadixCache 的 #26218。

实现拆解

实现按以下步骤展开：

1. 启动期组合校验（kv_cache_builder.py 的 build_kv_cache）：原逻辑对 SWA 模型一律抛错；现在仅当 SGLANG_ENABLE_UNIFIED_RADIX_TREE 或 use_mlx() 时放行 --disaggregation-decode-enable-radix-cache + SWA 组合；仍拒绝 hierarchical cache、DeepSeek-V4 DSA 压缩 KV、SWA-compress（Gemma4 / MiMo-V2）以及 Mamba/SSM。
2. decode 侧窗口裕量与容量统计（decode.py）：_swa_tail_len 在启用 radix cache 时把窗口起点改为 seq_len - 1 - max(window_size, page_size) 后再页对齐，保证按页截断的 radix key 内仍包含完整 SWA 窗口；新增 _radix_full_evictable、_radix_full_protected、_radix_full_available 访问器，让混合 SWA 模型能分别统计 full-attention 与 SWA 池容量。
3. SWA 容量回收（decode.py 的 _reclaim_swa_tail_capacity）：按页对齐计算 SWA 尾部需求，不足时先从树驱逐 SWA 页，仍不足则返回错误串；pop_preallocated 接到错误后仅 abort 该请求（FINISH_ABORT 清理），而不是让调度线程崩溃。
4. 锁生命周期与去重转发（decode.py、unified_radix_cache.py）：_release_matched_prefix_lock 通过 DecLockRefParams(swa_uuid_for_lock) 和 skip_swa=True 区分 full/SWA 锁释放；cache_unfinished_req 在插入去重后重新匹配、

把 live 请求的 token table 重写到 cache-owned slots，并转移锁到最深 full-resident 节点。得益于 SWA 插入侧保留的窗口 margin，normal 匹配即可安全转发，最终无需引入 return_full_match 特例。

5. 测试与 CI 配套：单元测试覆盖窗口裕量、admission 预算、reclaim 成功 / 失败与单请求拒绝；新增 gpt-oss-20b NIXL 端到端测试（GSM8K 0.50 阈值 + 两遍非回归校验）；H20 8-GPU 测试移入 extra-b stage 缓解 runner 排队，校准混合缓存 CI 阈值，并修复 AMD 依赖脚本的引号参数传递。

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 _release_matched_prefix_lock, _reclaim_swa_tail_capacity, _radix_full_evictable, _radix_full_protected）：核心实现文件：SWA 窗口裕量计算、SWA 容量回收、full/SWA 锁释放与容量统计访问器全部在此，是 P/D 分离解码准入路径的关键改造。
- python/sglang/srt/mem_cache/kv_cache_builder.py（模块 缓存构建；类别 source；类型 dependency-wiring；符号 build_kv_cache）：启动期校验的开关所在：决定 SWA 模型何时允许启用 decode 侧 radix cache，并显式拒绝未支持组合。
- python/sglang/srt/mem_cache/unified_radix_cache.py（模块 前缀缓存；类别 source；类型 core-logic；符号 cache_unfinished_req）：cache_unfinished_req 是去重转发与锁转移的核心，本 PR 确认其匹配注释并依赖插入侧 margin 保证 normal match 安全。
- test/registered/unit/mem_cache/test_decode_radix_lock_ref.py（模块 锁引用；类别 test；类型 test-coverage；符号 test_swa_tail_len_keeps_page_aligned_matchable_window, test_swa_admission_counts_evictable_capacity, test_reclaim_swa_tail_capacity_page_rounds, test_reclaim_swa_tail_capacity_fails_before_allocation）：以 mock 方式覆盖锁引用计数平衡的关键 4 个场景，是验证 SWA 窗口裕量与 reclaim 逻辑正确性的主要单元测试。
- test/registered/disaggregation/test_disaggregation_decode_radix_cache_swa.py（模块 SWA 测试；类别 test；类型 test-coverage；符号 TestDisaggregationDecodeRadixCacheSWANixl）：新增端到端 NIXL 测试，在 8×H200 上以 gpt-oss-20b 验证多轮缓存命中与 GSM8K 两遍非回归，是功能合入的主要验收证据。
- test/registered/unit/disaggregation/test_decode_queue_cleanup.py（模块 队列清理；类别 test；类型 test-coverage；符号 test_swa_reclaim_failure_rejects_only_request）：验证 SWA 驱逐失败时只拒绝该请求、正确清理 kv_receiver 并 abort，覆盖 review 中要求的崩溃改拒绝行为。
- test/registered/disaggregation/test_disaggregation_decode_radix_cache.py（模块 解耦测试；类别 test；类型 test-coverage）：基础 decode radix cache 测试的阈值与预算校准，避免 SWA 改动影响既有非 SWA 用例。
- scripts/ci/amd/amd_ci_install_dependency.sh（模块 CI 脚本；类别 infra；类型 infrastructure）：部署脚本修复，保留带引号的 retry 参数，属于 CI 基础设施配套改动。
- test/registered/radix_cache/test_mamba2_extra_buffer_kl.py（模块 Mamba 测试；类别 test；类型 test-coverage）：Mamba 缓存测试的配置校准，确保新增校验逻辑不误伤 SSM 路径的既有覆盖。

- test/registered/unit/mem_cache/test_hisparse_allocator.py（模块 稀疏分配器；类别 test；类型 test-coverage）：稀疏分配器测试的极小校准，与 unified cache 体系相关。

关键符号：_release_matched_prefix_lock, _reclaim_swa_tail_capacity, _radix_full_evictable, _radix_full_protected, _radix_full_available, _swa_tail_len, cache_unfinished_req, build_kv_cache

关键源码片段

python/sglang/srt/disaggregation/decode.py

核心实现文件：SWA 窗口裕量计算、SWA 容量回收、full/SWA 锁释放与容量统计访问器全部在此，是 P/D 分离解码准入路径的关键改造。

```
# python/sglang/srt/disaggregation/decode.py
# SWA 混合模型在 decode 侧复用 full-attention 前缀时需要的辅助逻辑。
# 核心问题：full KV 生命周期长、可跨请求复用，SWA KV 只对当前窗口有效，
# 因此 radix 匹配、容量计算与锁释放都必须能区分这两部分。

    def _release_matched_prefix_lock(self, req: Req) -> None:
        # decode 侧只复用 full-attention 前缀，SWA 尾部每次新分配，
        # 所以释放已匹配节点锁时通过 swa_uuid_for_lock 区分两部分；
        # 若 SWA 部分已提前释放（swa_prefix_lock_released），
        # 则调用 skip_swa=True 只释放 full 部分的锁，避免锁计数失衡。
        params = DecLockRefParams(swa_uuid_for_lock=req.swa_uuid_for_lock)
        if req.swa_prefix_lock_released:
            self.tree_cache.dec_lock_ref(req.last_node, params, skip_swa=True)
            req.swa_prefix_lock_released = False
        else:
            self.tree_cache.dec_lock_ref(req.last_node, params)

    def _reclaim_swa_tail_capacity(
        self, swa_tail_len: int, req_id: str
    ) -> Optional[str]:
        # 按页对齐计算 SWA 尾部需求，不足时先尝试从 radix 树驱逐 SWA 页；
        # 仍不足则返回错误串，由调用方拒绝该请求（而不是在调度线程崩溃）。
        page_size = self.token_to_kv_pool_allocator.page_size
        required = ceil_align(swa_tail_len, page_size)
        available = self.token_to_kv_pool_allocator.swa_available_size()
        if available < required:
            self.tree_cache.evict(EvictParams(swa_num_tokens=required - available))
            available = self.token_to_kv_pool_allocator.swa_available_size()
        if available < required:
            return (
                f"SWA eviction insufficient: needed={required}, "
                f"available={available}, req={req_id}"
            )
        return None

    def _radix_full_evictable(self) -> int:
```

```
        # 混合 SWA 模型通过 full_* 访问器单独统计 full-attention 池容量，
        # 普通模型退化为原有的 evictable / protected / available 统计。
        if self.scheduler.tp_worker.is_hybrid_swa:
            return self.tree_cache.full_evictable_size()
        return self.tree_cache.evictable_size()

    def _swa_tail_len(self, seq_len: int) -> int:
        if not self._uses_swa_tail_prealloc() or seq_len <= 0:
            return max(seq_len, 0)
        window_size = self.scheduler.sliding_window_size
        if window_size is None or window_size <= 0:
            return seq_len
        page_size = self.token_to_kv_pool_allocator.page_size
        if getattr(
            self.scheduler.server_args,
            "disaggregation_decode_enable_radix_cache",
            False,
        ):
            # 保留窗口裕量：seq_len - 1 是最后一个已提交位置，
            # 窗口起点再前移 max(window_size, page_size) 并做页对齐，
            # 保证按页截断的 radix key 内仍包含完整 SWA 窗口，#29860 同款处理。
            # 否则会出现 insert 已复用 full KV、而 normal match 返回 0 的悬空问题。
            window_start = max(0, seq_len - 1 - max(window_size, page_size))
        else:
            window_start = max(0, seq_len - window_size)
        window_start = (window_start // page_size) * page_size
        return seq_len - window_start
```

python/sglang/srt/mem_cache/kv_cache_builder.py

启动期校验的开关所在：决定 SWA 模型何时允许启用 decode 侧 radix cache，并显式拒绝未支持组合。

```
# python/sglang/srt/mem_cache/kv_cache_builder.py
# build_kv_cache 中的启动期组合校验：
# decode-side radix cache 对 SWA 混合模型只允许走 unified radix tree，
# 并显式拒绝仍未适配的 hierarchical cache、DSA 与 SWA-compress 变体。
    if (
        get_disagg().disaggregation_decode_enable_radix_cache
        and get_disagg().disaggregation_mode == "decode"
    ):
        if is_hybrid_swa:
            if not (envs.SGLANG_ENABLE_UNIFIED_RADIX_TREE.get() or use_mlx()):
                raise ValueError(
                    "--disaggregation-decode-enable-radix-cache with sliding "
                    "window attention (SWA) models requires the unified radix "
                    "tree (set SGLANG_ENABLE_UNIFIED_RADIX_TREE=1)."
                )
            if enable_hierarchical_cache:
                raise ValueError(
```

```
                "SWA decode radix currently supports only device-resident "
                "cache and is incompatible with --enable-hierarchical-cache."
            )
        if getattr(model_config, "is_deepseek_v4_arch", False):
            raise ValueError(
                "decode radix cache does not support DeepSeek-V4 (DSA) "
                "compressed KV (c4 / c128 / indexer) yet."
            )
        if getattr(model_config, "is_hybrid_swa_compress", False):
            raise ValueError(
                "decode radix cache does not support SWA-compress models "
                "(e.g. Gemma4 / MiMo-V2) yet."
            )
    if is_hybrid_ssm:
        raise ValueError(
            "--disaggregation-decode-enable-radix-cache is incompatible "
            "with Mamba/SSM models."
        )
```

评论区精华

Review 中最重要的交锋是 return_full_match 是否必要：

- hzh0425 质疑："I think we don't actually need to add this new one. The decode-side insertions of 'full' and 'swa' should come in pairs."
- ishandhanani 先以 895 token、page_size=64、window=127 的实例反驳：insert 已去重 832 个 full tokens，而 normal match 因 SWA tombstone 返回 0，会触发 new_prefix_len <= len(new_indices) 检查失败。
- ispobock 指出 #29860 已在插入侧保留窗口 margin，作者随即承认："With the same margin, the 895/64/127 case starts SWA at 704 instead of 768 ... I will update the decode tail calculation to use that margin and remove return_full_match." 这一收敛避免了过度设计。

其他关键讨论：

- ShangmingCai 在 _release_matched_prefix_lock 处要求："Don't crush here. We could just reject this request instead of killing the main scheduler thread." 最终改为驱逐失败只拒绝该请求。
- ShangmingCai 询问是否应受 SGLANG_OPT_SWA_RELEASE_LEAF_LOCK_AFTER_WINDOW 控制，作者解释 decode 侧只复用 full 前缀、每次传输新鲜 SWA 尾部，gate 该路径会导致 protected SWA 页保留并复现 admission stall。
- ShangmingCai 质疑 gsm8k 0.45 阈值过低，作者上调至 0.50（实测 0.568 / 0.560）。
- hzh0425 要求合入前验证 PD + DSV4 + SWEBench 的命中率与稳定性，并批准后请 ShangmingCai 二次确认。
- return_full_match 是否必要 (design): 作者采纳 margin 方案，更新 decode 侧 tail 计算并移除 return_full_match，normal match 即可安全 repoint。

- SWA 驱逐失败时崩溃还是拒绝请求 (correctness): 实现改为 abort 并清理单个请求，新增 test_swa_reclaim_failure_rejects_only_request 覆盖。
- SWA 锁释放是否受 SGLANG_OPT_SWA_RELEASE_LEAF_LOCK_AFTER_WINDOW 控制 (design): 刻意不 gate，保持锁释放路径独立于该 flag。
- decode_hicache_mixin 是否需要同样处理 (question): 未在本 PR 内明确处理，属于遗留确认项。
- gsm8k 阈值 0.45 过低 (testing): 作者上调至 0.50，实测两遍 0.568 与 0.560。
- 测试移入 extra-b 队列 (testing): 作者移到 extra-b stage 以降低 base-c 队列压力。
- PD + DSV4 + SWEBench 验证要求 (testing): 合入前验证要求；DSV4 正式支持由 #27831 承接。

风险与影响

- 风险：主要风险集中在以下几点：
- 核心路径变更：decode.py 的 DecodePreallocQueue 是 P/D 分离解码的准入核心，本次修改对所有使用 decode radix cache 的用户生效；虽 _swa_tail_len 通过 getattr 默认关闭走旧分支，但非 SWA 模型仍建议回归。
- 锁引用计数平衡：_release_matched_prefix_lock 的 skip_swa 分支、cache_unfinished_req 的锁转移依赖 full/SWA 插入严格成对，一旦失衡会泄漏 SWA 页或提前释放锁；单元测试基于 mock，难以覆盖真实并发时序。
- 拒绝率变化：SWA 驱逐失败从崩溃改为拒绝请求，在内存压力场景下可能提升单请求失败率，需要观察生产指标确认对可用性的影响。
- 兼容性约束：hierarchical cache、DSA、SWA-compress、Mamba 组合会启动失败，报错信息明确但属于行为变化。
- CI 偶发失败：HiCache file backend restore 测试在 H20 上多次失败后通过，疑似 flaky，已通过移入 extra-b 缓解。
- 影响：对用户：gpt-oss 等 SWA 混合模型首次可在 P/D 分离下启用 decode 侧 radix 缓存，跨请求 / 跨轮复用 full-attention KV，减少 prefill 到 decode 的传输量并降低 decode 首 token 时延；功能默认关闭，需显式开启 unified radix tree。对系统：decode worker 的容量统计与锁管理需要区分 full/SWA 两个池，内存预算逻辑更复杂。对团队：本 PR 确立了 SWA 统一 radix 路线的基线，后续 Mamba（#26828）、DSV4/MiMo（#27831）扩展均以它为蓝本，review 中已经形成了明确的分工共识。
- 风险标记：核心路径变更，锁引用计数平衡，CI 偶发失败，实验性开关 gated，模型组合受限

关联脉络

- PR #19746 P/D disaggregation decode-side radix cache: PR body 明确本 PR 是 #19746 引入的 decode-side radix cache 在 SWA 混合模型上的扩展（标题为基于上下文的推断）。
- PR #26218 SWA decode radix cache via default SWARadixCache: PR body 声明 supersede #26218，同一功能统一改走 unified radix tree（标题为基于上下文的推断）。

- PR #26828 Linear-attention/Mamba *decode* radix *cache* + MTP: issue 评论中 laixinn 提到正在做线性注意力（mamba *cache*）与 MTP 支持，作者建议作为本 PR 之后的分工项（标题为基于上下文的推断）。
- PR #27831 DeepSeek-V4 *decode* radix *cache* + MTP support: 评论中 zhangxiaolei123456 与 TobyMint 提到基于本 PR 方法扩展到 DSV4、MiMo-V2、Gemma4，是直接的下游演进。
- PR #29860 Keep SWA window margin at insert boundary: review 中 ispobock 指出该 PR 已在插入侧保留窗口 margin，直接决定了本 PR 移除 return_full_match 的最终方案（标题为基于上下文的推断）。
- PR #28085 Related *decode* radix *cache* fix: ShangmingCai 在 review 中提示可能与 *decode*_hicache_mixin 相关的 PR，需要 double-check（标题未在材料中给出，为占位推断）。