PR #31443 完整报告

sgl-project/sglang

[HiCache]: Optimize hybrid/DSA L3 prefetch result sync and usable-prefix clamping

合并时间：2026-07-21 18:59

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/31443

执行摘要

- 一句话：优化 HiCache 预取收尾与可用前缀截断，修复 hybrid 模型乱码
- 推荐动作：值得精读。这是 HiCache 混合预取收尾逻辑的 canonical 实现，重点看三点：一是 helper 提取如何统一两套缓存树的收尾路径；二是 clampable 判定如何以 sidecar 的 hit_policy 与 source pool 属性区分 DSA 与 SWA/Mamba 场景；三是前导连续成功段统计（rs.index(False)）对安全前缀长度的影响。建议后续跟进单卡跳过 all_reduce 的优化、统一 PrefetchOperation 属性契约，并补充覆盖中间 gap 场景的单测。

功能与动机

修复 issue #30321：GLM-5.2 在启用 EAGLE/MTP 推测解码与 Mooncake 分层缓存（HiCache）时，长尾生成偶发乱码或损坏文本。PR body 明确指出三处根因：跨 ATTN group 同步与可用前缀判定内联在 check_prefetch_progress 中，导致两套缓存实现行为不一致；sidecar 命中页数用 sum() 统计，中间出现 gap 时会高估安全前缀；SWA/Mamba 等非 DSA 混合池短缺时未整体丢弃预取结果。stepinto 在 PR #31348 复现该问题并提供 GLM-5.2 SWE-bench 复现命令，本 PR 为直接修复。

实现拆解

1. 抽取预取收尾 helper：在 python/sglang/srt/mem_cache/unified_radix_cache.py 新增 _sync_and_check_hybrid_prefetch_result，在 python/sglang/srt/mem_cache/hiradix_cache.py 新增 _sync_and_clamp_prefetch_result，把原本内联在 check_prefetch_progress 中的 all_reduce 同步、sidecar 命中页数规约、可用前缀计算统一收口；调用方只需处理 None（全量丢弃）或截断后的 token 数，避免两条缓存树各自维护一套收尾逻辑。
2. 引入 DSA 风格 clamp 与 all-or-nothing 双策略（Unified）：当所有 sidecar 都满足 hit_policy == PoolHitPolicy.ALL_PAGES 且 indices_from_pool == PoolName.KV（DSA / MiniMax indexer 场景）时，可用前缀取 Full KV 完成数与各 sidecar 命中页数的最小值并对齐 page_size，部分预取仍可使用；SWA / Mamba / DeepSeekV4 混合栈则要求每个池都完整覆盖目标前缀，任一短缺即整体丢弃并释放 host 内存、扣减 prefetch_tokens_occupied、清空 ongoing_prefetch。这样既保住 DSA 场景的 L3 命中收益，又避免混合栈 KV 错位。
3. 修正 sidecar 命中统计口径：python/sglang/srt/mem_cache/hicache_storage.py 的 PoolTransferResult.update_extra_pool_hit_pages 由 sum(rs) 改为 rs.index(False)（无

False 时取 len(rs))，只统计前导连续成功段，防止 fetch 中间断档时把后续成功页误算进安全前缀——这是乱码根因的直接修复点。

4. 预取终止守卫：python/sglang/srt/mem_cache/hybrid_cache/hybrid_cache_controller.py 的 _page_transfer 在触发 extra pool IO 前增加 not operation.is_terminated() 条件，终止后的预取不再执行 batch_get_v2，避免与 helper 的丢弃路径产生状态竞争。
5. 测试与验证配套：未新增自动化测试文件；CI 通过 /rerun-group hicache radix_cache/unified_radix_tree 回归，hicache 存储与 unified radix 组全部通过，8-gpu-h200 夜间组有偶发失败。社区验证：stepinto 在 GLM-5.2 SWE-bench 上测得 500 题从 393 提升到 399，DeepSeek-V4-Flash 解出 358/500；junliu-mde 在 0.5.15.post1 上连续运行数天无乱码复现。

关键文件：

- python/sglang/srt/mem_cache/unified_radix_cache.py（模块 前缀缓存；类别 source；类型 core-logic；符号 _sync_and_check_hybrid_prefetch_result, check_prefetch_progress）：核心变更文件：新增 _sync_and_check_hybrid_prefetch_result，把跨 ATTN group 同步与可用前缀判定从 check_prefetch_progress 抽出，实现 DSA 风格 clamp 与 all-or-nothing 双策略，是本 PR 正确性修复的主体。
- python/sglang/srt/mem_cache/hiradix_cache.py（模块 前缀缓存；类别 source；类型 core-logic；符号 _sync_and_clamp_prefetch_result, check_prefetch_progress）：HiRadix 缓存树（旧路径）的对应收尾实现 _sync_and_clamp_prefetch_result，保持两条缓存路径行为一致，并在 DSA 场景下同样支持前缀截断。
- python/sglang/srt/mem_cache/hicache_storage.py（模块 存储层；类别 source；类型 core-logic；符号 PoolTransferResult.update_extra_pool_hit_pages）：修复 PoolTransferResult.update_extra_pool_hit_pages 的统计口径：由 sum 改为前导连续成功段（rs.index(False)），是乱码根因的直接修复点。
- python/sglang/srt/mem_cache/hybrid_cache/hybrid_cache_controller.py（模块 缓存控制；类别 source；类型 entrypoint；符号 _page_transfer）：在 extra pool 传输入口增加 operation.is_terminated() 守卫，避免终止后的预取仍触发 sidecar IO，与 helper 的丢弃收尾逻辑配合。

关键符号：_sync_and_check_hybrid_prefetch_result, _sync_and_clamp_prefetch_result, check_prefetch_progress, PoolTransferResult.update_extra_pool_hit_pages, HybridCacheController._page_transfer

关键源码片段

python/sglang/srt/mem_cache/unified_radix_cache.py

核心变更文件：新增 _sync_and_check_hybrid_prefetch_result，把跨 ATTN group 同步与可用前缀判定从 check_prefetch_progress 抽出，实现 DSA 风格 clamp 与 all-or-nothing 双策略，是本 PR 正确性修复的主体。

```
def _sync_and_check_hybrid_prefetch_result(
    self,
    req_id: str,
```

```
    operation: PrefetchOperation,
    completed_tokens: int,
    hash_value: list[str],
    host_indices: torch.Tensor,
    last_host_node: UnifiedTreeNode,
    anchor_lock_params: DecLockRefParams,
    prefetch_key: RadixKey,
) -> Optional[int]:
    """跨 ATTN group 同步预取结果，并决定可用的前缀长度。

    按混合缓存布局分两种策略：DSA 风格（Full attention 加 KV 派生的
    ALL_PAGES sidecar，如 DSA / MiniMax indexer）会截断到 Full KV 池与
    每个 sidecar 共同取得的最小前缀，因为 sidecar 与 KV 页对齐且每页都
    需要它，部分前缀仍可用；其它情况（SWA / Mamba 组件、混合 DeepSeekV4
    栈）采用 all-or-nothing，这些池只覆盖窗口或尾部，无法按页截断，任何
    短缺都会丢弃整个预取结果。

    返回同步后的可用 token 数（可能被截断，可能为 0）；当 all-or-nothing
    预取被丢弃时返回 None，调用方应把预取视为已结束。
    """
    # 把完成 token 数与各池命中页数打包成 tensor，一次 all_reduce 取最小，
    # 保证所有 rank 对同一可用前缀长度达成一致。
    pool_transfers = operation.pool_transfers or []
    hit_pages = (
        operation.pool_storage_result.extra_pool_hit_pages
        if pool_transfers
        else {}
    )
    pool_hit_pages = [hit_pages.get(t.name, 0) for t in pool_transfers]
    packed = torch.tensor([completed_tokens, *pool_hit_pages], dtype=torch.int)
    self._all_reduce_attn_groups(packed, torch.distributed.ReduceOp.MIN)
    min_completed_tokens = int(packed[0].item())
    pool_hit_pages = list(map(int, packed[1:].tolist()))
    for transfer, count in zip(pool_transfers, pool_hit_pages):
        hit_pages[transfer.name] = count

    # DSA 风格截断：每个 sidecar 都是 KV 派生的 ALL_PAGES 池，对完整前缀
    # 都必需，所以可用长度就是 Full KV 完成数与各 sidecar 命中的共同最小值。
    clampable = bool(pool_transfers) and all(
        t.hit_policy == PoolHitPolicy.ALL_PAGES
        and t.indices_from_pool == PoolName.KV
        for t in pool_transfers
    )
    if clampable:
        usable_pages = min(min_completed_tokens // self.page_size, *pool_hit_pages)
        return usable_pages * self.page_size

    # 混合缓存状态是 all-or-nothing：每个额外池（SWA / Mamba 等）都必须
    # 覆盖相同的预取前缀；任一池短缺则整个预取结果不可用，丢弃并释放资源。
```

```
    expected_tokens = len(hash_value) * self.page_size
    all_succeeded = min_completed_tokens == expected_tokens and all(
        transfer.keys is not None and count == len(transfer.keys)
        for transfer, count in zip(pool_transfers, pool_hit_pages)
    )
    if pool_transfers and not all_succeeded:
        # 控制器的预取 IO 线程已释放未传输尾部（host_indices[completed_tokens:]），
        # 这里只释放已传输部分并清理记账状态，然后上报丢弃日志。
        self.cache_controller.append_host_mem_release(
            host_indices=host_indices[:completed_tokens],
            extra_pools=pool_transfers,
        )
        self.dec_host_lock_ref(last_host_node, anchor_lock_params)
        del self.ongoing_prefetch[req_id]
        self.cache_controller.prefetch_tokens_occupied -= len(prefetch_key)
        self.prefetch_loaded_tokens_by_reqid[req_id] = 0
        logger.warning(
            "HiCache hybrid prefetch discarded req=%s completed=%d requested=%d",
            req_id,
            completed_tokens,
            expected_tokens,
        )
        return None
    return min_completed_tokens
```

评论区精华

- 单卡同步开销（gemini-code-assist，medium）：helper 内无条件构造 tensor 并调用 _all_reduce_attn_groups，建议在 tp_world_size > 1 或 attn 并行组有效时才执行同步。合并版本未采纳该建议；单 rank 的 all_reduce 语义正确，但相比 base 版本少了 tp_world_size > 1 保护，每完成一次预取仍遗留少量额外开销。
- 跨 rank 分支一致性（ConcentrativeMan）：询问 if pool_transfers and not all_succeeded: 是否保证所有 rank 一致进入或跳过该分支。hzh0425 答复：all_succeeded 来自同步（all_reduce MIN）后的结果，pool_transfers 在 hybrid 场景恒为非空列表，因此所有 rank 必然一致。
- DSA indexer 预取时机（chenhao-stick-to / stepinto）：_page_transfer 仅在 kv_completed_pages == len(hash_value) 时同步 trailing keys，DSA indexer 的预取会一直停在 0，已完成的 KV 预取对 indexer 无效。stepinto 确认这是真实优化空间，由 #31668 与 #27010 后续修复；本 PR 保持最小修复范围。
  - 单卡 / 非并行配置下的无谓分布式同步 (performance): 合并版本未采纳该建议；单 rank 上 all_reduce 语义正确，但相比 base 版本去掉了 tp_world_size > 1 保护，每完成一次预取仍留有少量额外开销，属于遗留性能优化项。
  - 所有 rank 是否保证走同一丢弃分支 (correctness): 作者已澄清机制；正确性依赖 pool_transfers 在各 rank 的一致性构造，未引入额外防护。

- DSA indexer 也应参与 prefetch 的优化空间 (design): 确认为真实优化空间，由后续 PR 修复；本 PR 保持最小修复范围，不展开。

风险与影响

- 风险：
  1. 单卡同步开销回归：unified_radix_cache.py 的 _sync_and_check_hybrid_prefetch_result 无条件执行 _all_reduce_attn_groups，base 版本有 if self.tp_world_size > 1 保护；gemini-code-assist 已指出但合并版本未修复。单 rank 语义正确，但每次预取收尾多一次 tensor 分配与集合通信调用。
  2. 属性契约不一致：hiradix_cache.py 用 getattr(operation, "pool_transfers", None) 防御，unified_radix_cache.py 直接访问 operation.pool_transfers；若未来有代码路径构造不完整的 PrefetchOperation，两条路径的报错行为不一致。
  3. clampable 判定依赖隐式标注：clamp 策略只对 PoolHitPolicy.ALL_PAGES + indices_from_pool == PoolName.KV 的 sidecar 生效；新增 sidecar 类型若未正确标注，会误入 all-or-nothing（保守丢前缀，性能损失）或误判 clampable（前缀不一致，正确性风险）。
  4. 缺少自动化测试覆盖：本 PR 没有新增测试文件，正确性主要依赖人工 SWE-bench 验证与 CI 回归组；all-or-nothing 丢弃路径中 append_host_mem_release、prefetch_tokens_occupied 扣减等资源释放逻辑需要防止与正常路径重复执行。 - 影响：影响范围集中在启用 HiCache（分层缓存 / L3）且使用混合模型的部署，典型场景为 GLM-5.2、DeepSeek-V4、MiniMax-H3（DSA indexer）配合 EAGLE/MTP 推测解码与 Mooncake 或文件存储后端。对用户而言，乱码输出问题消除（junliu-mde 在 0.5.15.post1 上运行数天无新报告）；DSA 风格场景从整体丢弃变为截断使用，L3 有效命中率提升，GLM-5.2 SWE-bench 500 题从 393 提升到 399。对团队而言，本 PR 确立了 hybrid 预取收尾的两策略模型（clamp / all-or-nothing），为后续 #31668、#27010 的 DSA indexer prefetch 演进铺路；同时遗留了单卡同步开销优化与单测补充两个待办。 - 风险标记：核心路径变更，缺少测试覆盖，单卡同步开销未优化，策略判定依赖隐式标注

关联脉络

- PR #34823 skip oow slot freeing under eagle: 同改 python/sglang/srt/mem_cache/unified_radix_cache.py，同属 HiCache / radix 缓存生命周期正确性修复线，可对照阅读。
- PR #31348 （上下文中未提供标题）: PR body 中引用：stepinto 用于复现 issue #30321 的 PR，并提供了 GLM-5.2 SWE-bench 复现命令。
- PR #31668 （上下文中未提供标题）: 评论区确认的后续修复：让 DSA indexer 也参与 prefetch，避免 indexer 命中恒为 0；与 #27010 一起推进本 PR 遗留的优化空间。