PR #37091 完整报告

sgl-project/sglang

[Unified Cache][1/N]: Support cache contract for external linker

合并时间：2026-08-30 22:24

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/37091

执行摘要

- 一句话：为外部缓存链接器定义传输与加载契约，并跟踪插入采用的 KV 范围
- 推荐动作：建议精读。该 PR 展示了如何在不污染既有路径的前提下，通过阶段枚举和组件钩子为跨节点 KV 缓存定义清晰契约，值得学习其“先契约后实现”的拆分策略。关注重点：record_adopted_range 的区间合并语义、external_cache_stored 与 HiCache 备份的互斥设计，以及 SWA 的 TRAILING_PAGES 窗口对齐方式。

功能与动机

PR body 明确说明这是 external linker 系列的第一个 PR，从原先的完整实现 #35687 中拆分出来，目的是引入组件级传输和加载生命周期契约，并在 UnifiedTree 插入时记录被 Full/SWA 采用的范围，以便后续去重远程加载。由于只动缓存侧契约，linker 接线与后端实现留到后续 PR，这样可以独立评审缓存侧的正确性。

实现拆解

实现按“契约定义、组件实现、树状态集成、测试配套”四步展开：

1. 定义外部链接器阶段契约：在 tree_component.py 中新增 LinkerTransferPhase（LOOKUP / LOAD / OFFLOAD）和 ExternalLinkerLoadPhase（PREPARE / COMMIT / ABORT）两个枚举，并在 TreeComponent 基类增加默认钩子 build_external_linker_transfer 和 update_external_linker_load，默认分别返回 None 与原样透传，保证未接入链路时行为不变。
2. 按组件实现传输构造：
   - full_component.py 新增 _full_allocator() 统一选择 full-attention 池；build_external_linker_transfer 在 OFFLOAD 时导出整段设备 KV 索引与逐页哈希，LOOKUP 只带 keys，LOAD 时先按页数计算不足并驱逐，再分配设备槽位填入 PoolTransfer.device_indices；update_external_linker_load 在 ABORT 阶段释放预留槽位。
   - swa_component.py 更复杂：OFFLOAD 只导出滑动窗口尾部若干页并标记 PoolHitPolicy.TRAILING_PAGES；LOAD 分配 swa_attn_allocator 的槽位；PREPARE 阶段将 full 与 SWA 槽位建立 set_full_to_swa_mapping，并按 prefix_len 推算 swa_evicted_seqlen 写回 req.kv；COMMIT 阶段用最终落定的 canonical full 索引重新映射。

- mamba_component.py 显式 raise AssertionError，声明暂不支持外部链接器模式，避免静默错误。

3. 为 UnifiedTree 增加外部缓存状态与采用范围记录：unified_tree_core.py 中 UnifiedTreeNode 新增 external_cache_stored 标志，UnifiedTreeCore 新增 enable_external_cache_linker 开关（默认 False）；_inc_hit_count_and_check 在开启外部链接器时改用 external_cache_stored 判断是否触发 write-back，_add_new_node 在开启时也计算 hash_value，_split_node 传播该标志；_build_backup_kv_action 将 external_cache_stored 的祖先视为无需再备份。同时在 InsertResult 中新增 adopted_ranges 与 record_adopted_range()，在 insert walk 的 unevict、overlap、commit 三个位置按组件记录实际吸收 KV 的区间，供后续去重远端加载。
4. 配套调整与测试：hicache_storage.py 的 PoolTransferResult 新增 restorable_prefix_pages 元数据，用于多 rank 场景下由调用方对可恢复前缀页集合求交；components/__init__.py 导出新枚举；测试文件 test_unified_radix_cache_unittest.py 在 test_prev_prefix_len 中开启 track_adopted_ranges 并断言 adopted_ranges[ComponentType.FULL] 的区间正确性。

关键文件：

- python/sglang/srt/mem_cache/unified_cache/components/swa_component.py（模块 缓存组件；类别 source；类型 core-logic；符号 build_external_linker_transfer, update_external_linker_load）：SWA 组件首次实现外部链接器钩子，包含窗口页对齐、TRAILING_PAGES 策略、full→SWA 映射与 req.kv 回写，是契约落地最复杂的组件。
- python/sglang/srt/mem_cache/unified_cache/components/full_component.py（模块 缓存组件；类别 source；类型 core-logic；符号 _full_allocator, build_external_linker_transfer, update_external_linker_load）：Full 组件为整个外部链接器提供 KV 主数据的 OFFLOAD/LOOKUP/LOAD 实现，并抽象出 _full_allocator 处理 SWA 开启时的池选择。
- python/sglang/srt/mem_cache/unified_cache/components/tree_component.py（模块 组件契约；类别 source；类型 core-logic；符号 LRURefreshPhase, LinkerTransferPhase, ExternalLinkerLoadPhase, build_external_linker_transfer）：定义外部链接器的阶段枚举与组件基类钩子，是整个契约的骨架，所有组件实现都依赖这里的抽象。
- python/sglang/srt/mem_cache/unified_cache/unified_tree_core.py（模块 树核心；类别 source；类型 core-logic）：把 external_cache_stored 状态、adopted_ranges 记录和外部链接器开关接入树核心的插入 / 拆分裂 / 备份路径，是状态机集成的关键文件。
- python/sglang/srt/mem_cache/base_prefix_cache.py（模块 前缀缓存；类别 source；类型 core-logic；符号 record_adopted_range）：为 InsertResult 增加 adopted_ranges 数据模型和 record_adopted_range 方法，并新增 InsertParams.track_adopted_ranges 开关，是所有组件记录采用范围的数据基础。
- python/sglang/srt/mem_cache/unified_cache/components/mamba_component.py（模块 缓存组件；类别 source；类型 core-logic；符号 build_external_linker_transfer）：Mamba 组件通过显式 AssertionError 声明暂不支持外部链接器，防止未来误用。
- python/sglang/srt/mem_cache/hicache_storage.py（模块 缓存存储；类别 source；类型 core-logic）：为 PoolTransferResult 增加 restorable_prefix_pages 字段，描述

TRAILING_PAGES 池在节点边界上的可恢复前缀页集合，供多 rank 调用方求交。

- python/sglang/srt/mem_cache/unified_cache/components/__init__.py（模块 组件导出；类别 source；类型 core-logic）：导出两个新阶段枚举，保证组件模块的对外 API 一致。
- test/registered/unit/mem_cache/test_unified_radix_cache_unittest.py（模块 缓存测试；类别 test；类型 test-coverage）：验证 track_adopted_ranges 开启后 adopted_ranges 中 Full 组件区间记录正确，是唯一覆盖新契约语义的测试。

关键符号：build_external_linker_transfer, update_external_linker_load, record_adopted_range, _full_allocator, update_component_on_insert_overlap, recover_after_unevict, commit_insert_component_data

关键源码片段

python/sglang/srt/mem_cache/unified_cache/components/full_component.py

Full 组件为整个外部链接器提供 KV 主数据的 OFFLOAD/LOOKUP/LOAD 实现，并抽象出 _full_allocator 处理 SWA 开启时的池选择。

```
def _full_allocator(self):
    """返回独占 full-attention 池的分配器。"""
    # SWA 启用时 full 池与 SWA 池分离，需要单独选择
    allocator = self.cache.token_to_kv_pool_allocator
    return allocator.full_attn_allocator if self.cache.is_swa_enabled else allocator


def build_external_linker_transfer(
    self,
    phase: LinkerTransferPhase,
    node: Optional[UnifiedTreeNode],
    keys: Optional[Sequence[str]],
) -> Optional[PoolTransfer]:
    # OFFLOAD：把设备上的整段 KV 连同每页哈希交给外部存储
    if phase == LinkerTransferPhase.OFFLOAD:
        if node is None or not node.hash_value:
            return None
        value = node.component_data[self.component_type].value
        if value is None:
            return None
        return PoolTransfer(
            name=PoolName.KV,
            device_indices=value.to(torch.int64), # 设备页索引，供远端读取
            keys=list(node.hash_value), # 每页哈希，用于远端去重
        )

    if not keys:
        return None

    # LOOKUP：只查询远端是否存在这些页
```

```
    if phase == LinkerTransferPhase.LOOKUP:
        return PoolTransfer(name=PoolName.KV, keys=list(keys))

    # LOAD：先按页数预留设备 KV 槽位，缺页走本地驱逐腾位
    if phase == LinkerTransferPhase.LOAD:
        allocator = self._full_allocator()
        num_tokens = len(keys) * self.cache.page_size
        shortfall = max(0, num_tokens - allocator.available_size())
        if shortfall:
            self.cache.evict(EvictParams(num_tokens=shortfall))
        slots = allocator.alloc(num_tokens)
        if slots is None:
            return None
        return PoolTransfer(
            name=PoolName.KV,
            device_indices=slots.to(torch.int64), # 远端写入的落点
            keys=list(keys),
        )


def update_external_linker_load(
    self,
    phase: ExternalLinkerLoadPhase,
    req: Req,
    full_transfer: PoolTransfer,
    transfer: PoolTransfer,
    prefix_len: int,
    *,
    insert_result: Optional[InsertResult] = None,
    canonical_full: Optional[torch.Tensor] = None,
) -> Optional[PoolTransfer]:
    # ABORT：远端加载失败时释放本地预留的槽位
    if phase == ExternalLinkerLoadPhase.ABORT:
        self._full_allocator().free(transfer.device_indices)
        return None
    return transfer
```

python/sglang/srt/mem_cache/unified_cache/components/tree_component.py

定义外部链接器的阶段枚举与组件基类钩子，是整个契约的骨架，所有组件实现都依赖这里的抽象。

```
class LinkerTransferPhase(str, Enum):
    LOOKUP = "lookup" # 仅查询远端是否已有对应 KV 页
    LOAD = "load" # 向远端请求数据并写入本地池
    OFFLOAD = "offload" # 将本地设备页导出到远端存储


class ExternalLinkerLoadPhase(str, Enum):
```

```
    PREPARE = "prepare" # 远端数据到达前预留本地槽位 / 映射
    COMMIT = "commit" # 数据落盘后提交映射与请求状态
    ABORT = "abort" # 加载失败，释放 PREPARE 预留的资源


class TreeComponent(ABC):
    # ---- External Cache Linker Hooks ----

    def build_external_linker_transfer(
        self,
        phase: LinkerTransferPhase,
        node: Optional[UnifiedTreeNode],
        keys: Optional[Sequence[str]],
    ) -> Optional[PoolTransfer]:
        """构建本组件与外部缓存之间的一次直接传输。

        ``node`` 在 OFFLOAD 时携带设备页数据，其他阶段为 None；
        ``keys`` 是设备未缓存尾部的每页哈希（page 0 为第一个未缓存页），
        在 LOOKUP / LOAD 时提供，OFFLOAD 时取 ``node.hash_value``。
        默认返回 None，表示该组件不参与外部链接。
        """
        return None

    def update_external_linker_load(
        self,
        phase: ExternalLinkerLoadPhase,
        req: Req,
        full_transfer: PoolTransfer,
        transfer: PoolTransfer,
        prefix_len: int,
        *,
        insert_result: Optional[InsertResult] = None,
        canonical_full: Optional[torch.Tensor] = None,
    ) -> Optional[PoolTransfer]:
        """准备、提交或中止本组件的一次直接加载。"""
        return transfer
```

评论区精华

该 PR 没有行内 review 评论，仅有一条 huangtingwei9988 的 APPROVED。Issue 评论主要是作者触发 CI 重跑（/rerun-test 与 /rerun-group）以及机器人回报测试通过，AMD ROCm 网格失败但未在评论中展开。从实现可以推断设计者有意将 external_cache_stored 与 HiCache 的 backuped 状态分离，使外部链接器存储过的祖先节点在 _build_backup_kv_action 中跳过重复备份；Mamba 组件则用显式断言标明能力边界，避免后续误用。

- 外部链接器契约的设计权衡 (question): 未记录正式讨论；契约边界和 Mamba 不支持以代码形式固化为后续约束。

风险与影响

- 风险：核心风险集中在默认路径的回归与边界状态的一致性上：
  - unified_tree_core.py 的插入 walk 在所有组件方法签名中新增 result: InsertResult，任何遗漏更新该签名的第三方组件或子类都会导致 TypeError；但仓库内组件均已同步。
  - record_adopted_range 目前只在 SWA 与 unevict 路径上被调用，Full 的普通插入通过 _insert_commit_step 和 _insert_walk_step 显式记录，若后续组件漏记会导致远端加载去重不完整。
  - external_cache_stored 在 _split_node 中传播但在 _add_new_node 中初始化为 False，新节点若由已存储节点派生，可能存在需要手工恢复标志的遗漏点。
  - swa_component.update_external_linker_load 中 req.kv 的 swa_evicted_seqlen 计算依赖 prefix_len 与窗口页数，若外部 linker 返回的 keys 顺序或页面边界与本地窗口不对齐，会引入静默错误。
  - 新开关 enable_external_cache_linker 默认 False，所有新增逻辑在未启用时不生效，回归面被控制在契约接入后。
  - 影响：对现有用户与系统：无行为影响，因为外部链接器尚未接线且开关默认关闭。对开发团队：这是一次核心缓存模块的接口扩展，后续外部链接器 PR（如 #37151）将直接依赖这些钩子与元数据；同时 InsertResult/InsertParams 的字段扩展要求所有调用 cache.insert 的下游保持兼容。对测试体系：新增的 adopted_ranges 断言覆盖了插入去重语义，但尚缺少对 OFFLOAD/LOAD/ABORT 全生命周期的单元测试。
  - 风险标记：新增契约默认关闭，组件方法签名变更，测试覆盖有限，跨模块状态依赖

关联脉络

- PR #35687 原始外部链接器完整实现（本 PR 的拆分来源）：本 PR 从 #35687 拆分而来，后续链接器接线与后端实现将继续沿用本 PR 定义的契约。
- PR #37151 [Unified Cache Linker][3/N]: Add backend-independent linker core: 同一外部链接器系列的后续 PR，在该契约之上增加后端无关的链接器核心，是直接依赖本契约的下一步实现。
- PR #35245 refactor(unified-memory): translate the KV write location once, at ForwardBatch construction: 同属 Unified Cache / 内存池演进方向，重构了 KV 写入位置的翻译流程，与本 PR 在 unified_tree_core 与 KvIndexTranslator 等路径上有潜在交互。