

PR #37914 完整报告

sgl-project/sglang

[Unified Cache][7/N] Support MTP, EAGLE, and DSpark draft KV caches in the external linker

合并时间: 2026-09-13 18:22

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/37914>

执行摘要

- 一句话: 外部链接器支持草稿 KV 打包并新增严格校验
- 推荐动作: 值得精读: 这是外部链接器从「只存目标 KV」走向「目标 + 草稿统一打包」的关键一步, layer_mapping 的 `int | Sequence[int]` 放宽与 `_with_packed_draft_mapping` 的层号复用技巧是整条 Unified Cache 功能线里最值得借鉴的设计点。阅读时建议重点关注三处: (1) `_with_packed_draft_mapping` 的 `(layer_mapping[depth], target_device_layer_num + depth)` 约定如何与 components 的拼接顺序耦合, 任何一侧顺序变化都会静默错配; (2) `_build_hicache_draft_plan` 中新增的 fail-fast 是否是最终形态——Leoyzen 在评论里提出的「草稿部分失败只丢 draft 还是整条 entry」还没有答案, 若采纳 fail-soft, 这里需要配套改动; (3) DSV4 与 DSA 两条策略的校验强度不一致 (page size、KV/indexer 数量), 建议补齐 DSV4 侧校验并补一个 `is_unified_kv=True` 且带草稿池的负向用例。另外建议在合并后回看 CI: GPU linker 用例在 rerun 时两条 lane 都失败, 需要确认是 flaky 还是真实回归。

功能与动机

PR body 明确本 PR 是从 #35687 拆出的系列 PR 之一 (标题标 [7/N], body 称第 5 个), 前序拆分包括 #37091、#37098、#37151、#37205、#37578、#37381。要解决的问题是: 外部链接器此前只处理目标 KV——开启 `enable_unified_cache_external_linker` 时, `_build_hicache_draft_plan` 的早退条件里并不包含链接器开关, 草稿池既不会被写入 `mtp_draft_device_pools`, 也不会进入链接器的设备池视图, 导致 MTP / EAGLE / DSpark 的草稿 KV 在 L3 命中回放时缺失。作者在评论中给出的对比数据正是围绕「cold → L3-hit replay」这条路径做的精度验证, 说明动机是让草稿状态也能随统一缓存一起跨层回放。

实现拆解

1. 打通草稿池的登记入口 (python/sglang/srt/speculative/base_spec_worker.py):
BaseSpecWorker._build_hicache_draft_plan 的早退条件新增 `get_memory().enable_unified_cache_external_linker`, 使链接器开启时也会走完整的草稿规划流程; packed 分支把 `draft_pools` 写回 `target_model_runner.mtp_draft_device_pools`。若 `_can_pack_hicache_mtp()` 判定不可打包且链接器开启, 则直接 `raise NotImplementedError("The external linker only supports packed draft KV caches.")`, 取代原来的 SIDECAR 兜底——因为 sidecar 形态无法被链接器表达, 静默漏存比启动报错更危险。

- 2. 放宽设备池的层映射结构 (python/sglang/srt/mem_cache/hybrid_cache/linker_pool_assembler.py) : DevicePoolEntry.layer_mapping 的类型从 dict[int, int] 变为 dict[int, int | Sequence[int]]; get_prepared_layer_range_meta 先把取值归一化成 buffer_indices 列表, 再对每个 component 展开成多个 (base_ptr, row_stride, size, offset) 项, 使一次 transfer 能把目标层与草稿层的物理 buffer 一起搬运, 而 packed 模式下的行级并行数组结构保持不变。
- 3. 新增打包映射工具 _with_packed_draft_mapping: 约定「草稿 depth N 复用目标第 N 个 transfer layer」, 因此映射值为 (layer_mapping[depth], target_device_layer_num + depth), 第二项即拼接后组件列表里草稿 buffer 的下标; 当 draft_layer_num > len(layer_mapping) 时抛 ValueError, 避免草稿层数超过目标层数导致错位。该工具是 DSV4 SWA 与 DSA 两条策略共用的核心。
- 4. 在两条策略里应用打包 (同上文件): _build_deepseek_v4_device_pool_group(kvcache, page_size, mtp_draft_device_pools=()) 在非 unified KV 分支里展平 pool.swa_kv_pool.kv_buffer 得到 draft SWA buffer, 用 target_device_layer_num=len(kvcache.swa_kv_pool.kv_buffer) 生成 SWA 层映射, 并把组件改为 [*kvcache.swa_kv_pool.kv_buffer, *draft_swa_buffers]; _build_dsa_device_pool_group(kvcache, page_size, mtp_draft_device_pools=()) 新增草稿池 page size 校验与 «KV / indexer 草稿层数必须一致» 校验, 然后以相同的 layer_mapping 同时改造 PoolName.KV (rows_are_pages=False) 与 PoolName.INDEXER (rows_are_pages=True) 两个 entry。
- 5. 参数透传与测试配套: hybrid_pool_assembler.py 中 _DeepSeekV4Strategy.build_direct_linker_pool_group 与 _DsaStrategy.build_direct_linker_pool_group 改为把 params.mtp_draft_device_pools 传给 assembler; test/registered/unit/mem_cache/test_linker_pool_assembler.py 同步扩展——DSV4 用例注入假 draft SWA buffer 并断言 get_prepared_layer_range_meta 返回打包后的 sizes / offsets, DSA 用例抽出 dsa_pool(kv_width, index_width) 工厂后追加两个 draft 池, 新增 test_linker_requires_packed_draft 用 4 组 (linker_enabled, nextn_layers) 组合校验启动期行为。

文件	关键符号	变更性质
linker_pool_assembler.py	_with_packed_draft_mapping、 _build_dsa_device_pool_group、 get_prepared_layer_range_meta	核心机制 + 两条策略
hybrid_pool_assembler.py	_DeepSeekV4Strategy / _DsaStrategy 的 build_direct_linker_pool_group	参数透传
base_spec_worker.py	BaseSpecWorker._build_hicache_draft_plan	入口条件 + fail-fast
test_linker_pool_assembler.py	test_linker_requires_packed_draft、dsa_pool	单测覆盖

关键文件:

- python/sglang/srt/mem_cache/hybrid_cache/linker_pool_assembler.py (模块 链接池装配; 类别 source; 类型 core-logic; 符号 DevicePoolEntry.init, DevicePoolEntry.get_prepared_layer_range_meta, _with_packed_draft_mapping, _build_deepseek_v4_device_pool_group) : 本 PR 的核心机制文件: 放宽 DevicePoolEntry 层映射、新增 _with_packed_draft_mapping, 并在 DSV4 SWA 与 DSA (KV + INDEXER) 两条装配路径上实现草稿 buffer 打包与校验。
- python/sglang/srt/speculative/base_spec_worker.py (模块 草稿规划; 类别 source; 类型 core-logic; 符号 BaseSpecWorker._build_hicache_draft_plan) : 草稿规划的入口文件 : 把链接器开关纳入 _build_hicache_draft_plan 的触发条件, 并对不可打包的草稿改为启动期 fail-fast, 直接决定存量的链接器部署能否启动。
- test/registered/unit/mem_cache/test_linker_pool_assembler.py (模块 装配测试; 类别 test; 类型 test-coverage; 符号 test_dsa_uses_hybrid_assembler_strategy, dsa_pool, test_linker_requires_packed_draft) : 唯一测试文件: 既验证了草稿 buffer 打包后的 sizes / offsets 元数据, 也用新增用例固化了链接器 + 非 packed 草稿必须启动报错的契约。
- python/sglang/srt/mem_cache/hybrid_cache/hybrid_pool_assembler.py (模块 混合池策略; 类别 source; 类型 core-logic; 符号 _DeepSeekV4Strategy.build_direct_linker_pool_group, _DsaStrategy.build_direct_linker_pool_group) : 策略层的参数透传点: _DsaStrategy 与 _DeepSeekV4Strategy 的 build_direct_linker_pool_group 从 params 读取 mtp_draft_device_pools 并转交 assembler, 是草稿池从模型执行层进入链接器装配层的唯一通路。

关键符号: _with_packed_draft_mapping, _build_dsa_device_pool_group, _build_deepseek_v4_device_pool_group, DevicePoolEntry.get_prepared_layer_range_meta, BaseSpecWorker._build_hicache_draft_plan, _DsaStrategy.build_direct_linker_pool_group, _DeepSeekV4Strategy.build_direct_linker_pool_group, test_linker_requires_packed_draft, test_dsa_uses_hybrid_assembler_strategy

关键源码片段

python/sglang/srt/mem_cache/hybrid_cache/linker_pool_assembler.py

本 PR 的核心机制文件: 放宽 DevicePoolEntry 层映射、新增 _with_packed_draft_mapping, 并在 DSV4 SWA 与 DSA (KV + INDEXER) 两条装配路径上实现草稿 buffer 打包与校验。

```
# 打包映射的 " 层号复用 " 约定: 草稿 depth N 不新增 transfer layer,
# 而是挂在目标第 N 层上; 第二项 target_device_layer_num + depth
# 正是拼接后组件列表里草稿 buffer 的物理下标。
def _with_packed_draft_mapping(
    layer_mapping: dict[int, int],
    *,
    target_device_layer_num: int,
    draft_layer_num: int,
) -> dict[int, int | tuple[int, ...]]:
    """Attach draft depth N to the same transfer layer as target layer N."""
    # 草稿层数一旦超过目标 transfer layer 数, 第 N 层草稿就找不到可复用的目标层,
```

```

# 这里选择直接失败，而不是让草稿 KV 被静默漏存。
if draft_layer_num > len(layer_mapping):
    raise ValueError(
        "Packed draft layers exceed the target transfer layer count: "
        f"{draft_layer_num} > {len(layer_mapping)})."
    )
result: dict[int, int | tuple[int, ...]] = dict(layer_mapping)
for depth in range(draft_layer_num):
    # 第一个元素沿用原目标层下标，第二个元素指向草稿 buffer，
    # 于是 get_prepared_layer_range_meta 会对同一层返回两组元数据。
    result[depth] = (layer_mapping[depth], target_device_layer_num + depth)
return result

def _build_dsa_device_pool_group(
    kvcache: Any,
    page_size: int,
    mtp_draft_device_pools: tuple[Any, ...] = (),
) -> DevicePoolGroup:
    # 目标池与草稿池的 page size 必须一致，否则指针 / size 计算会整体错位。
    if kvcache.page_size != page_size:
        raise ValueError(
            "DSA KV page size must match the tree page size: "
            f"{kvcache.page_size} != {page_size}."
        )
    num_layers = kvcache.layer_num
    if any(pool.page_size != page_size for pool in mtp_draft_device_pools):
        raise ValueError("DSA MTP page size must match the tree page size.")

    # 按 " 池顺序 -> 层顺序 " 展平草稿 buffer；KV 与 indexer 必须一一对应，
    # 否则打包映射里的第二个下标会串到错误的 buffer 上。
    draft_kv_buffers = [
        buffer for pool in mtp_draft_device_pools for buffer in pool.kv_buffer
    ]
    draft_indexer_buffers = [
        buffer
        for pool in mtp_draft_device_pools
        for buffer in pool.index_k_with_scale_buffer
    ]
    if len(draft_kv_buffers) != len(draft_indexer_buffers):
        raise ValueError("DSA MTP KV and indexer draft layer counts must match.")

    # 目标层保持 0..num_layers-1 的恒等映射，草稿层从 num_layers 开始排布，
    # 与下面 components 的 " 目标在前、草稿在后 " 拼接顺序严格对应。
    layer_mapping = _with_packed_draft_mapping(
        {layer: layer for layer in range(num_layers)},
        target_device_layer_num=num_layers,
        draft_layer_num=len(draft_kv_buffers),
    )

```

```

entries = [
    DevicePoolEntry(
        name=PoolName.KV,
        indices_from_pool=PoolName.KV,
        device_pool=kvcache,
        # 同一 entry 内把草稿 KV 追加在目标 KV 之后，沿用同一份 layer_mapping。
        components=[[*kvcache.kv_buffer, *draft_kv_buffers]],
        layer_mapping=layer_mapping,
        page_size=page_size,
        rows_are_pages=False,
    ),
    DevicePoolEntry(
        name=PoolName.INDEXER,
        indices_from_pool=PoolName.KV,
        device_pool=kvcache,
        # indexer 侧同样追加草稿 buffer，保证 KV 与 indexer 的层语义同步。
        components=[*kvcache.index_k_with_scale_buffer, *draft_indexer_buffers]],
        layer_mapping=layer_mapping,
        page_size=page_size,
        rows_are_pages=True,
    ),
]
# transfer_layer_num 仍是 num_layers，草稿没有引入新的逻辑层。
return DevicePoolGroup(entries, num_layers, page_size, rank_replicated=True)

```

python/sglang/srt/speculative/base_spec_worker.py

草稿规划的入口文件：把链接器开关纳入 _build_hicache_draft_plan 的触发条件，并对不可打包的草稿改为启动期 fail-fast，直接决定存量的链接器部署能否启动。

```

def _build_hicache_draft_plan(self) -> HiCacheDraftPlan:
    target_model_runner = self.target_worker.model_runner
    # 先清空，任何失败路径都不会留下半成品状态。
    target_model_runner.mtp_draft_device_pools = ()
    spec_algorithm = target_model_runner.spec_algorithm
    # 新增条件：external linker 打开时也要走草稿规划流程，
    # 否则草稿池不会被登记进 mtp_draft_device_pools，链接器也就看不到它。
    if not (
        get_memory().enable_hierarchical_cache
        or get_memory().enable_unified_cache_external_linker
        or get_disagg().disaggregation_decode_retraction_backup == "host_pool"
    ):
        return HiCacheDraftPlan()

    draft_runners = self._draft_model_runners()
    if not draft_runners:
        return HiCacheDraftPlan()
    draft_pools = tuple(runner.token_to_kv_pool for runner in draft_runners)
    # (Inkling MTP 草稿状态在此前已被显式拒绝，逻辑未变。)

```

```

if _can_pack_hicache_mtp(spec_algorithm, draft_runners):
    # 只有 packed 形态才能与目标层共用同一个 transfer layer,
    # 因此这是唯一会把 draft pools 暴露给链接器的分支。
    target_model_runner.mtp_draft_device_pools = draft_pools
    return HiCacheDraftPlan(
        mode=HiCacheDraftMode.PACKED,
        device_pools=draft_pools,
    )

# 新增的 fail-fast: 链接器无法表达 sidecar 形态的草稿缓存,
# 与其静默漏存, 不如在启动期就报错。
if get_memory().enable_unified_cache_external_linker:
    raise NotImplementedError(
        "The external linker only supports packed draft KV caches."
    )

return HiCacheDraftPlan(
    mode=HiCacheDraftMode.SIDECAR,
    # Preserve the legacy non-packed HiCache behavior: multi-layer
    # EAGLE registers only the first draft runner as the sidecar.
    device_pools=draft_pools[:1],
)

```

test/registered/unit/mem_cache/test_linker_pool_assembler.py

唯一测试文件：既验证了草稿 buffer 打包后的 sizes / offsets 元数据，也用新增用例固化了链接器 + 非 packed 草稿必须启动报错的契约。

```

def test_linker_requires_packed_draft(self):
    """Do not accept draft state that the linker would omit from storage."""
    from sglang.srt.speculative import base_spec_worker as spec
    from sglang.srt.speculative.spec_info import SpeculativeAlgorithm

    # 构造最小可用的 draft runner / target model runner,
    # num_nextn_predict_layers 决定草稿池能否 packed。
    draft = SimpleNamespace(
        token_to_kv_pool=object(),
        model_config=SimpleNamespace(
            num_nextn_predict_layers=0,
            hf_config=SimpleNamespace(architectures=["LlamaForCausalLM"]),
        ),
    )

    target = SimpleNamespace(spec_algorithm=SpeculativeAlgorithm.EAGLE)
    worker = SimpleNamespace(
        target_worker=SimpleNamespace(model_runner=target),
        _draft_model_runners=lambda: (draft,),
    )

    # 四种组合：链接器开 / 关 × NextN 0 / 1 层；
    # 只有 "开链接器 + 0 层" 这一格必须抛错。
    for linker_enabled, nextn_layers in (

```

```

(False, 0),
(True, 0),
(False, 1),
(True, 1),
):
draft.model_config.num_nextn_predict_layers = nextn_layers
with (
    self.subTest(linker=linker_enabled, nextn=nextn_layers),
    patch.object(
        spec,
        "get_memory",
        return_value=SimpleNamespace(
            enable_hierarchical_cache=not linker_enabled,
            enable_unified_cache_external_linker=linker_enabled,
        ),
    ),
):
    if linker_enabled and not nextn_layers:
        with self.assertRaisesRegex(
            NotImplementedError, "only supports packed"
        ):
            spec.BaseSpecWorker._build_hicache_draft_plan(worker)
        # 失败路径必须保持 mtp_draft_device_pools 为空,
        # 否则链接器可能读到未完成登记的草稿池。
        self.assertEqual(target.mtp_draft_device_pools, ())
    else:
        plan = spec.BaseSpecWorker._build_hicache_draft_plan(worker)
        self.assertEqual(
            plan.mode,
            spec.HiCacheDraftMode.PACKED
            if nextn_layers
            else spec.HiCacheDraftMode.SIDECAR,
        )
        self.assertEqual(plan.device_pools, (draft.token_to_kv_pool,))
        # 只有 packed 形态才允许把草稿池暴露给外部链接器。
        self.assertEqual(
            target.mtp_draft_device_pools,
            plan.device_pools if nextn_layers else (),
        )

```

评论区精华

本 PR 没有行内 review 评论，最终由 hzh0425、Qiaolin-Yu、alphabetc1 三人 approve；真正的交锋来自 issue 评论区。

1. 生产环境 linker 加载路径的稳定性 (Leoyzen)：他在 DeepSeek-V4-Flash、TP4、Mooncake L3、约 2100 万 key / 1.4TB master 的生产配置下报告了两类故障：一是 master 驱逐与在途加载竞争，21k key 的 BatchEvict 之后 batch_get_session_start 对 32 个 key 中的 2 个返回 -702 (INVALID_REPLICA)，而 ranged session 协议只接受

memory replica, 「this is fatal by construction」; 二是部分批次失败会拖垮整个引擎, 而不是丢弃失败 key 重新 prefill。他提交的 #38347 让加载 fail-soft (提交 device slot 前 重新校验、失败 key 退化为 cache miss + re-prefill, 与 vLLM mooncake connector 同策略), 并提示该改动会碰 mooncake_direct_linker.py 的 load_layer_wise / load, 与本 PR 有少量重叠, 需要 rebase。

2. 草稿部分失败的降级粒度 (Leoyzen 的提问, 本 PR 未答复): 他直接问「with draft caches (MTP/EAGLE/DSPark) added to the load path, is a partial failure supposed to drop just the draft component or the whole entry?」, 并补充他们的 draft KV 是按 rank 复制的, 草稿单独 miss 似乎可以同样降级。这与本 PR 的打包方案直接相关。
 3. 精度与接受长度验证 (作者): 贴出 cold → L3-hit replay 的对比表——DSV4 SWE tool-call validity 98.50% → 98.00%、平均接受长度 4.283 → 4.321; DSV4 GSM8K 20-shot 99.50% → 99.50%、4.243 → 4.235; GLM-5.2 SWE 100.00% → 99.50%、3.548 → 3.376; GLM-5.2 GSM8K 98.50% → 98.50%、3.626 → 3.648。
 4. CI 排查 (hzh0425): 先后尝试 `/rerun-test radix_cache/unified_radix_tree` (bot 回复 File not found) 与 `/rerun-group radix_cache/unified_radix_tree`, group 重跑在 4-gpu-h100 (5 个用例) 与 8-gpu-h200 (4 个用例, 含 linker/test_unified_cache_linker_kl_glm52.py) 两条 lane 均未通过; PR 的 CI 状态里 AMD ROCm 10 也是 🟡。
- 链接器加载路径 fail-hard 与 fail-soft 的取舍 (design): 本 PR 内未做任何改动或回应; 该议题被留给 #38347 处理, 两 PR 的合并顺序需要协调。
 - 草稿部分失败时应丢 draft 还是整条 entry (question): 评论区内没有答复。从本 PR 的实现看, 草稿 buffer 与目标 buffer 被打包进同一个 transfer layer (layer_mapping 的元组取值), 因此「只丢 draft」在当前打包方案下并不自然成立, 需要链接器侧单独设计降级语义。
 - 非 packed 草稿在链接器下必须 fail-fast (correctness): 已实现并由单测固化, 代价是存量非 packed 草稿 + 链接器的配置会在启动期失败。
 - 草稿 KV 回放的精度与接受长度验证 (performance): 给出了正负两侧的数据, 没有设定阈值或进一步解释 GLM-5.2 SWE 接受长度下降约 5% 的原因。
 - linker GPU 用例的 rerun 失败 (testing): 评论区内没有达成结论, PR 在没有全绿记录的情况下被合并, 失败原因是 flaky 还是真实回归仍待确认。

风险与影响

- 风险:
 1. 启动期行为变更 (base_spec_worker.py): 新增的 NotImplementedError("The external linker only supports packed draft KV caches.") 会在 init_hicache_draft_plan 阶段抛出。原先链接器开启但未开 hicache 时该函数会早退并返回空 plan (草稿 KV 静默漏存、但不报错), 现在会进入完整流程并可能直接启动失败。对已在用链接器 + 非 packed 草稿 (例如多层 EAGLE 且 num_nextn_predict_layers 为 0) 的存量部署, 这是一次需要显式适配的行为变更。
 1. 层序对齐依赖隐式约定 (linker_pool_assembler.py): `_with_packed_draft_mapping` 把「草稿 depth N」绑定到「目标第 N 个 transfer layer」, 只校验 `draft_layer_num >`

`len(layer_mapping)`，既不检查 `layer_mapping` 的 key 是否覆盖 `0..draft_layer_num-1`（稀疏映射会抛 `KeyError` 而非语义化报错），也不校验草稿池的顺序是否与推理时 `depth` 顺序一致。若调用方传入顺序错乱的 `mtp_draft_device_pools`，草稿 KV 会被写到错误的 `transfer layer` 上，属于静默数据错配。

- DSV4 路径校验弱于 DSA 路径：`_build_dsa_device_pool_group` 新增了草稿池 `page size` 校验与 `len(draft_kv_buffers) != len(draft_indexer_buffers)` 校验，但 `_build_deepseek_v4_device_pool_group` 只校验目标 `kvcache.swa_page_size`，没有对 `mtp_draft_device_pools` 的 `swa_page_size` 做同样检查，`page size` 不一致时会得到错误的 `size / offset` 而不是报错。
- unified KV 分支下草稿被静默忽略：DSV4 的 SWA entry（草稿 buffer 唯一挂载点）位于 `if not is_unified_kv:` 分支内，`is_unified_kv=True` 时 `mtp_draft_device_pools` 不会被消费，也没有任何告警。需要确认该组合是否实际可达。另外草稿池属性（`swa_kv_pool`、`index_k_with_scale_buffer`）都是直接访问，缺少 `getattr` 兜底，异常模型组合会以 `AttributeError` 形式失败。
- 打包后的层内异构元数据：只有前 `draft_layer_num` 个层携带第二个 buffer，`get_prepared_layer_range_meta` 返回的每层 `items` 数量因此不一致。宏观看协议侧（Mooncake 直连）需要容忍「同一 `transfer layer` 内 buffer 数随层变化」，这是本 PR 引入的新协议形态，若下游存储后端假设每层 buffer 数恒定会出错。
- 端到端验证证据不足：新增用例是 CPU 级单测（`register_cpu_ci`，`est_time=11`），真正跑 linker 的 GPU 用例（`test/registered/radix_cache/unified_radix_tree/linker/test_unified_cache_linker_kl_dsv4.py`、`..._kl_glm52.py` 等）在 `/rerun-group` 中两条 lane 均失败，AMD ROCm 10 lane 也是 Ⓣ，合并时缺少全绿记录。- 影响：对用户：使用 unified cache 外部链接器（Mooncake 直连）且开启投机解码的部署，首次可以把 MTP / EAGLE / DSpark 的草稿 KV 落盘并在 L3 命中时回放，少一次草稿状态的重新 `prefill`；作者的对照表显示精度基本持平（DSV4 GSM8K 99.50% → 99.50%），但 GLM-5.2 SWE 的接受长度从 3.548 降到 3.376、得分 100% → 99.5%，属于需要写进发布说明的可见波动。对系统：改动只发生在链接器的设备池装配与启动规划层，不改内核与前向计算，但把「草稿缓存是否参与存储」从隐式变为显式契约——非 packed 草稿会直接启动失败。对团队：本 PR 是与 #38347（加载 fail-soft）同一段代码的并行改动，二者涉及 `mooncake_direct_linker.py` 的 `load_layer_wise / load`，合并顺序与 `rebase` 需要协调；同时它把一个此前只在 HiCache 内成立的假设（草稿以 packed 形态存在）扩展成链接器的硬约束，后续做 `sidecar / 多层 EAGLE 草稿持久化` 的同学需要先解除这个约束。- 风险标记：启动期行为变更，草稿层序隐式约定，DSV4 缺少草稿池校验，unified KV 下草稿静默忽略，GPU 用例未全绿，与 #38347 代码重叠

关联脉络

- PR #35687 Unified Cache 主 PR（标题未在上下文中提供，来源为本 PR body）：本 PR 与 #37091、#37098、#37151、#37205、#37578、#37381 一样，都是从 #35687 拆分出来的子 PR；理解拆分的整体设计需要回看该 PR。
- PR #38347 让 Mooncake 直链接器加载 fail-soft（标题由评论内容推断）：Leoyzen 在评论中说明该 PR 让加载在提交 `device slot` 前重新校验、失败 key 退化为 `cache miss + re-prefill`，并提示它会改动 `mooncake_direct_linker.py` 的 `load_layer_wise / load`，与本

PR 有重叠、需要 rebase；两者共同决定草稿 KV 进入加载路径后的失败语义。

- PR #38486 [HiCache] Publish a host store event for storage-prefetch refills: 同属 HiCache / 统一缓存与外部链接器功能线，围绕 L2 索引与存储事件一致性做修复，与本 PR 的设备池装配、transfer layer 语义处于同一子系统上下文。