# PR #30762 完整报告

sgl-project/sglang

fix(hicache/umbp): support DeepSeek-V4 hybrid HostPoolGroup (multi-po…

合并时间：2026-08-13 15:40

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/30762

## 执行摘要

- 一句话：UMBP 支持 DSV4 混合 HostPoolGroup v2 多池接口
- 推荐动作：值得精读：这是 UMBP 从单池走向多池 v2 接口的关键实现，register_mem_host_pool_v2 / batch_exists_v2 的收窄查询与 fail-closed 语义设计值得借鉴；审查 E2E 测试的断言方式（storage cached_tokens >= PAGE_SIZE 且 storage_backend == UMBPStore）。关注点：nightly-only 验证缺口、零拷贝静默回退、logical anchor v1 no-op 语义。

## 功能与动机

UMBPStore 原先假设 mem_pool_host 是单一 KV 持有池，而 DeepSeek-V4 HiCache 栈中 mem_pool_host 是 HostPoolGroup，其 KV anchor 是只持有页索引、无物理 KV tensor 的 LogicalHostPool（get_page_buffer_meta() 按设计返回 None）。这导致两个问题：构造期解包 None 直接崩溃杀死 prefill worker；UMBPStore 未实现 v2 多池 API，side pool 无法存储 / 加载。PR body 明确说明这是本 PR 的目标场景：1 Prefill 节点 + 1 Decode 节点的 PD 分离下，UMBP 作为 L3 后端驱动 SWA / compressed KV / indexer / state 各 side pool。

## 实现拆解

按 4 步拆解：

1) 构造期探测加固：umbp_store.py __init__中将 get_page_buffer_meta() 返回值先存为 meta 再解包，逻辑 anchor 返回 None 时保留 dram_page_size=0，交给 mori master 默认值 + PoolClient partial-tail 兜底； 2) 逻辑 anchor 识别与注册重构：register_mem_pool_host 中一次性计算 _kv_anchor_is_logical（kv_buffer is None），跳过逻辑 anchor 的 RDMA 注册与 v1 I/O（batch_get_v1 / batch_set_v1 对逻辑 anchor 直接返回 [True]，对齐 MooncakeStore），并把 RDMA 注册逻辑抽取为 _register_host_buffer_for_zero_copy() helper 供单池 / 多池复用； 3) v2 多池接口实现：register_mem_host_pool_v2 跳过 PoolName.KV、注册各 side pool 并做零拷贝注册；_get_hybrid_page_component_keys 按 page_first 单对象每页布局生成带池名后缀的存储 key；batch_exists_v2 实现 ALL_PAGES / TRAILING_PAGES 命中策略并按 final_pages 逐步收窄查询；_batch_io_v2 统一驱动 batch_get_v2 / batch_set_v2 的逐池 zero-copy get/put 与 put 去重； 4) 配套改动：memory_pool_host.py 为 LogicalHostPool 增加无操作 destroy() 满足 HostPoolGroup 销毁契约；新增 test/registered/hicache/test_hicache_storage_umbp_backend.py 的 AMD 8-GPU E2E（round trip 验证 storage 层命中）；

test/registered/unit/mem_cache/test_umbp_store.py 增加防御性单元测试（逻辑 anchor 构造、短结果 fail-closed、跨池查询收窄）；test/run_suite.py 注册 nightly-amd-8-gpu-mi35x-deepseek-v4-flash nightly 套件。

关键文件：

- python/sglang/srt/mem_cache/storage/umbp/umbp_store.py（模块 存储后端；类别 source；类型 core-logic；符号 _register_host_buffer_for_zero_copy, register_mem_host_pool_v2, _get_hybrid_page_component_keys, batch_exists_v2）：核心实现文件，修复逻辑 anchor 探测崩溃并实现 v2 多池接口，是 PR 的主变更。
- test/registered/hicache/test_hicache_storage_umbp_backend.py（模块 端到端测试；类别 test；类型 test-coverage；符号 TestHiCacheStorageUMBPBackend, setUpClass, tearDownClass, _launch_server）：新增 AMD 8-GPU E2E 测试，验证 DSV4 混合 HostPoolGroup 经 UMBP 的 round trip，是唯一覆盖 v2 路径的端到端测试。
- test/registered/unit/mem_cache/test_umbp_store.py（模块 单元测试；类别 test；类型 test-coverage；符号 MockLogicalHostPool, MockHybridSidePool, get_page_buffer_meta, import_umbp_store_module）：单元测试覆盖逻辑 anchor 构造、v2 查询收窄、fail-closed 语义，是 PR CI 中唯一能跑的 UMBP 测试。
- python/sglang/srt/mem_cache/memory_pool_host.py（模块 内存池；类别 source；类型 core-logic；符号 destroy）：为 LogicalHostPool 补充无操作 destroy()，满足 HostPoolGroup 销毁契约，避免调度器关闭时隐藏子进程异常。
- test/registered/unit/mem_cache/test_hicache_staged_write_back_dispatch.py（模块 回写测试；类别 test；类型 test-coverage；符号 test_host_pool_group_destroys_logical_anchor）：补充 HostPoolGroup 销毁逻辑 anchor 的单元测试，守护 destroy() 契约。
- test/run_suite.py（模块 测试套件；类别 test；类型 test-coverage）：注册 nightly-amd-8-gpu-mi35x-deepseek-v4-flash 套件，使新 E2E 测试可被 nightly 调度。

关键符号：register_mem_pool_host, _register_host_buffer_for_zero_copy, register_mem_host_pool_v2, _get_hybrid_page_component_keys, batch_exists_v2, _batch_io_v2, batch_get_v2, batch_set_v2, destroy

## 关键源码片段

### python/sglang/srt/mem_cache/storage/umbp/umbp_store.py

核心实现文件，修复逻辑 anchor 探测崩溃并实现 v2 多池接口，是 PR 的主变更。

```
# python/sglang/srt/mem_cache/storage/umbp/umbp_store.py
# 逻辑 anchor 识别：DeepSeek-V4 混合 HostPoolGroup 的 KV anchor 是 LogicalHostPool，
# 只持有页索引、无物理 KV tensor。这里一次性计算标志位，供 v1 no-op 与 v2 注册复用，
# 避免在多个调用点重复 getattr 防御式判断。
def register_mem_pool_host(self, mem_pool_host: HostKVCache):
    super().register_mem_pool_host(mem_pool_host)
    # 仅支持 page_first 系列布局，这是 UMBP 零拷贝 / key 映射的前提
    assert self.mem_pool_host.layout in [
        "page_first", "page_first_direct", "page_head",
    ], "UMBP store only supports page_first, page_first_direct, or page_head layout"
```

```
    # 逻辑 anchor 没有 kv_buffer，无需也不能做 RDMA 注册；
    # 真实的 side pool 缓冲由 register_mem_host_pool_v2() 逐个注册
    self._kv_anchor_is_logical = self.mem_pool_host.kv_buffer is None
    self._zero_copy_registered = False
    if self._kv_anchor_is_logical:
        return

    # 单池路径：整块 KV 缓冲预注册到 RDMA IOEngine，使 PoolClient 走 zero-copy
    if self._register_host_buffer_for_zero_copy(mem_pool_host):
        self._zero_copy_registered = True


def _register_host_buffer_for_zero_copy(self, host_pool: HostKVCache) -> bool:
    """注册 host pool 的 KV 缓冲到 RDMA IOEngine，供零拷贝传输使用。

    单池路径（register_mem_pool_host）与多池路径（register_mem_host_pool_v2）共用；
    返回 True 表示注册成功，False 表示跳过 / 失败（调用方透明回退 staging buffer）。

    注意：当前只处理 page_first 布局下单个连续 kv_buffer 的情形；若后续支持
    layer_first 布局，或 side pool 通过 get_hybrid_pool_buffer() 暴露多缓冲
    （例如 DSAIndexerPoolHost 的 index_k_with_scale_buffer），必须扩展本分支
    逐个注册，否则这些池会静默跳过零拷贝、走较慢的 staging buffer 路径。
    """
    if self.client is None:
        return False
    try:
        is_distributed = bool(self.client.is_distributed())
    except Exception:
        is_distributed = False
    if not is_distributed:
        return False
    if not hasattr(self.client, "register_memory"):
        return False
    if getattr(self, "_disable_zero_copy_register", False):
        return False

    kv_buffer = getattr(host_pool, "kv_buffer", None)
    if kv_buffer is None:
        return False
    try:
        host_ptr = int(kv_buffer.data_ptr())
        host_size = int(kv_buffer.numel() * kv_buffer.element_size())
        # 大页场景下 mmap 区域会向上取整到巨页边界，RDMA ibv_reg_mr
        # 要求注册完整巨页，因此优先使用 mapped_size
        allocator = getattr(host_pool, "allocator", None)
        mapped_size_fn = getattr(allocator, "mapped_size_for", None)
        mapped_size = mapped_size_fn(host_ptr) if mapped_size_fn is not None else
        getattr(allocator, "mapped_size", 0)
        if mapped_size > host_size:
```

```
            host_size = mapped_size
        ok = bool(self.client.register_memory(host_ptr, host_size))
    except Exception as exc:
        logger.warning("UMBPStore: register_memory failed (%s); falling back to staging buffer
        path.", exc)
        return False
    if ok:
        logger.info("UMBPStore: registered host KV buffer for RDMA zero-copy (ptr=0x%x, size=
        %d MB)", host_ptr, host_size // (1024 * 1024))
        return True
    logger.warning("UMBPStore: register_memory returned false; staying on staging buffer
    fallback path.")
    return False
```

## test/registered/hicache/test_hicache_storage_umbp_backend.py

新增 AMD 8-GPU E2E 测试，验证 DSV4 混合 HostPoolGroup 经 UMBP 的 round trip，是唯一覆盖 v2 路径的端到端测试。

```
# test/registered/hicache/test_hicache_storage_umbp_backend.py
# DeepSeek-V4 混合 HostPoolGroup 侧池经 UMBP L3 的 round trip 验证。
# 注册为 AMD nightly 专用套件，需要 ROCm + unified_kv_triton 的 DSV4 分支。
register_amd_ci(
    est_time=3600,
    suite="nightly-amd-8-gpu-mi35x-deepseek-v4-flash",
    nightly=True,
)

@unittest.skipUnless(is_hip(), "UMBP HiCache requires ROCm.")
class TestHiCacheStorageUMBPBackend(CustomTestCase):
    input_ids = list(range(4000, 5024))

    def test_hybrid_host_pool_round_trip_from_umbp(self):
        # 先清空设备与宿主缓存，确保第一次请求完全冷启动
        self._flush_device_and_host_cache()
        first = self._generate()
        self.assertEqual(first["meta_info"]["cached_tokens"], 0)

        # UMBP 写入在请求路径之下是异步的，先等待其落盘，再清缓存
        time.sleep(15)
        self._flush_device_and_host_cache()

        # 第二次请求必须从 UMBP 恢复 side-pool KV，并上报 storage 层命中
        second = self._generate()
        cached_details = second["meta_info"].get("cached_tokens_details") or {}
        storage_cached_tokens = int(cached_details.get("storage", 0))
        self.assertGreaterEqual(
            storage_cached_tokens,
            PAGE_SIZE,
            f"Expected DeepSeek-V4 side-pool KV to load from UMBP storage, got {cached_details=}
```

```
            ",
        )
        self.assertEqual(cached_details.get("storage_backend"), "UMBPStore")
```

## 评论区精华

核心讨论有三条：

1) gemini-code-assist 指出 _register_host_buffer_for_zero_copy 只处理 kv_buffer 单缓冲，DSAIndexerPoolHost 等无 kv_buffer 的 side pool 会静默退回 staging buffer 路径，作者以代码注释记录 layer_first / 多缓冲扩展点，maning00 认为当前不阻塞； 2) gemini-code-assist 建议 batch_exists_v2 只查询至 final_pages 减少 ZMQ roundtrip，作者在 86aa1e574 中采纳并补回归测试； 3) TianDi101 提出 hasattr 防御式写法违反 no-getattr-defensive 规则且 4 处重复，作者重构为 __init__一次性初始化、单点计算 _kv_anchor_is_logical；另有 batch_set_v2 去重冗余的讨论（UMBP 本身去重，已删除）、extra_info 未传给 _batch_io_v2 导致 eviction depth 缺失（maning00 确认当前调用方传 None，不阻塞）。hzh0425 要求补 E2E 测试，作者按要求新增 AMD nightly-only 测试。

- 零拷贝注册仅支持 kv_buffer 单缓冲 (design): 作者在代码中补充 NOTE(layer_first) 注释记录扩展点，maning00 认为当前不阻塞；保留问题。
- batch_exists_v2 应只查询至 final_pages 减少网络往返 (performance): 作者在 86aa1e574 中采纳并补 4→2→1 收窄回归测试。
- registered_pools 与逻辑 anchor 判断应单点初始化 (design): 作者重构为 __init__初始化 registered_pools / _kv_anchor_is_logical，register_mem_pool_host 单点计算。
- batch_set_v2 未传 extra_info 导致 eviction depth 缺失 (performance): maning00 确认当前 hybrid_cache_controller 调用 batch_set_v2 时 extra_info 为 None，不阻塞；作者未实现 depth 传递。
- E2E 测试注册方式与 PR CI 覆盖缺口 (testing): 作者通过 AMD 专用 job（如 dsv4-umbp-hicache-amd-rocm720）人工触发验证，测试通过；PR CI 覆盖缺口保留为已知限制。
- batch_set_v2 中的重复 put 去重是否冗余 (design): 作者删除冗余去重，改为直接 put。

## 风险与影响

- 风险： 1) 测试覆盖风险：E2E 测试注册为 nightly=True 且经 register_amd_ci 限定 nightly-amd-8-gpu-mi35x-deepseek-v4-flash 套件，PR CI 从不运行该路径（amd-bot 两次明确警告 "The changed DSv4 multi-pool code is not exercised by any PR-CI test"），且单元测试用 mocked HostKVCache； 2) 回归风险：batch_get_v1 / batch_set_v1 对逻辑 anchor 直接返回 [True]，若未来逻辑 anchor 携带物理数据会静默丢数据，当前依赖 kv_buffer is None 判断； 3) 零拷贝静默回退：DSAIndexerPoolHost 等无 kv_buffer 的 side pool 绕过 RDMA 注册走 staging buffer，性能劣化但无报错； 4) dram_page_size 留 0 依赖 mori master 默认 2 MiB 与 PoolClient partial-tail 兜底，size 错配场景依赖外部层； 5) memory_pool_host.py 新增 destroy() 为无操作，若未来逻辑池持有注册资源需同步扩展。

- 影响：影响范围：AMD（ROCm）HiCache 存储后端链路，使 UMBP（MoRI 分布式内存）成为 DeepSeek-V4 混合 HostPoolGroup 可用的 L3 后端；用户侧收益：PD 分离场景下 prefill 节点可将 SWA / compressed KV / indexer / state side pool 的 KV offload 到远端 DRAM / SSD，benchmark 显示 1P1D TP8 下 3,779 请求 0 错误、prefix cache hit 97.5%。对团队影响：新增 nightly AMD 套件与 CI 资源占用（MI35x 8-GPU）；v2 多池 API 成为 UMBP 后续支持其他混合池模型（layer_first 等）的基座。
- 风险标记：核心存储路径变更，nightly-only 验证，E2E 需 8 块 MI35X，零拷贝静默回退，PR CI 无法覆盖 DSV4 路径，兼容性依赖 mori 版本

## 关联脉络

- PR #34653 Enable unified cache out-of-window slot freeing by default: 同属统一缓存（unified cache / SWA 出窗）演进线，与 DSV4 混合 SWA 的缓存释放策略相关。
- PR #34644 [Fix] Snapshot req.prefix_indices when the prefix cache is disabled: 同为 kv-cache / scheduling 修复线，涉及 swa_radix_cache 与 unified_radix_cache。
- PR #34667 Drop the mmlu case from the unified radix cache kit: 同为 hicache / 统一缓存测试基础设施调整，涉及 unified_radix_cache_kit。
- PR #33894 refactor error responses into shared utils::response helpers: 与 PR 的 UMBPStore 重构同属错误处理 / helper 抽取模式，但为 Rust 服务端，关联度较弱。