PR #35306 完整报告

sgl-project/sglang

[mem_cache][9/N] refactor: move DSAIndexerPoolHost to pool_host.dsa

合并时间：2026-08-21 15:59

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35306

执行摘要

- 一句话：DSAIndexerPoolHost 迁入 pool_host.dsa，纯机械重构
- 推荐动作：值得快速浏览而非精读。如果你在参与 mem_cache 重构，可以把它当作“大段机械移动”的样板：提取 → 逐字节对比 → 重定向调用方 → 同步测试 patch 目标。值得关注的决策是“不加 shim”——短期有 ImportError 风险，长期保证结构不腐化。普通使用者只需记住导入路径已变化。

功能与动机

issue #25371 给出的 roadmap 把 memory_pool_host.py 拆为 pool_host/ 包，目标是消除 catch-all 文件。前序 PR #30616（7/N）、#31180（8/N）已移走 MLA 与 Mamba 宿主池，本 PR 延续同样的机械模式处理 DSA 索引宿主池。PR body 强调“byte-for-byte identical”，并刻意不添加 re-export shim，保持系列一贯策略。

实现拆解

1. 新建 pool_host/dsa.py（+482 行）：将 DSAIndexerPoolHost 类整体迁入，随迁其专属依赖——DSATokenToKVPool、TYPE_CHECKING 下的 MLATokenToKVPoolHost、以及平台门控的 sgl_kernel.kvcacheio 传输 kernel；共享基类 HostKVCache、host_memory_budget_bytes 和分配器辅助改从 pool_host.base / pool_host.common 引入。
2. memory_pool_host.py 纯删除（-440 行）：移除被搬走的类及 DSATokenToKVPool 导入、TYPE_CHECKING 中的 MLATokenToKVPoolHost；剩余类均不引用 DSAIndexerPoolHost，故不需要 back-import。
3. 调用方导入重定向（3 处）：运行时 hybrid_cache/hybrid_pool_assembler.py 与测试 test_dsa_pool_host_unit.py、test_hicache_staged_write_back_dispatch.py 的 import 从 memory_pool_host 改为 pool_host.dsa。
4. 测试 mock 目标同步：test_hicache_staged_write_back_dispatch.py 新增 DSA_POOL_HOST_MODULE 常量，将 staged/fallback/load 三个 transfer 函数的 patch 目标切到新模块，确保 backup_from_device_all_layer / load_to_device_per_layer 的分支仍被真实覆盖。
5. 机械移动验证：类体逐字节 diff 为空、git diff --check、Python 编译与 lint 全通过；本 PR 无运行时、速度或模型输出变化，不涉及文档。

关键文件：

- python/sglang/srt/mem_cache/pool_host/dsa.py（模块 宿主池；类别 source；类型 dependency-wiring；符号 DSAIndexerPoolHost, init, get_size_per_token, get_ksize_per_token）：新模块，DSAIndexerPoolHost 的新归属地；类体与旧位置逐字节一致，随迁平台门控导入与共享基类引用，是本次重构的核心产物。
- python/sglang/srt/mem_cache/memory_pool_host.py（模块 宿主池；类别 source；类型 dependency-wiring；符号 DSAIndexerPoolHost）：纯删除 440 行被搬走的类与 DSA 专属导入；剩余类均不依赖 DSAIndexerPoolHost，因此无需 back-import，是验证机械移动的关键目标。
- python/sglang/srt/mem_cache/hybrid_cache/hybrid_pool_assembler.py（模块 混合缓存；类别 source；类型 dependency-wiring）：运行时唯一调用方，import 重定向到 pool_host.dsa，保证装配逻辑仍能构造 DSA 宿主池。
- test/registered/unit/mem_cache/test_hicache_staged_write_back_dispatch.py（模块 测试；类别 test；类型 test-coverage）：新增 DSA_POOL_HOST_MODULE 并把 transfer 函数 patch 目标切到新模块，保证 staged write-back dispatch 测试真正命中迁移后的路径。
- test/registered/unit/mem_cache/test_dsa_pool_host_unit.py（模块 测试；类别 test；类型 test-coverage）：直接 import 类的单元测试，同步调整导入路径，保持测试覆盖有效。

关键符号：DSAIndexerPoolHost.init, DSAIndexerPoolHost.get_size_per_token, DSAIndexerPoolHost.get_ksize_per_token, DSAIndexerPoolHost.init_kv_buffer, DSAIndexerPoolHost._init_write_back_staging_buffers, DSAIndexerPoolHost.get_hybrid_pool_buffer, DSAIndexerPoolHost._get_indexer_page_indices

关键源码片段

python/sglang/srt/mem_cache/pool_host/dsa.py

新模块，DSAIndexerPoolHost 的新归属地；类体与旧位置逐字节一致，随迁平台门控导入与共享基类引用，是本次重构的核心产物。

```
class DSAIndexerPoolHost(HostKVCache):
    """Host-side DSA index buffers only. Slot layout matches the anchor MLA host pool."""

    device_pool: DSATokenToKVPool

    def __init__(
        self,
        device_pool: DSATokenToKVPool,
        anchor_host: MLATokenToKVPoolHost,
        layout: str,
        pin_memory: bool = True,
        device: str = "cpu",
        allocator_type: str = "default",
    ):
```

```
self.device_pool = device_pool
# 锚点 host pool 提供 page 级元数据，DSA 索引池只镜像同一套槽位布局
self.page_size = anchor_host.page_size
self.layout = layout
self.pin_memory = pin_memory
self.device = device
self.allocator = get_allocator_from_storage(allocator_type)
self.dtype = device_pool.store_dtype
self.start_layer = device_pool.start_layer
self.end_layer = device_pool.end_layer
self.target_layer_num = self._effective_host_layer_num()
self.mtp_draft_device_pools = anchor_host.mtp_draft_device_pools
self.layer_num = self.target_layer_num + len(self.mtp_draft_device_pools)

# 索引按 token 记账：index_head_dim 一份 + 量化 block 对应的 4 字节 scale
self.index_head_dim = device_pool.index_head_dim
self.indexer_quant_block_size = device_pool.quant_block_size
self.indexer_dtype = DSATokenToKVPool.index_k_with_scale_buffer_dtype
self.indexer_size_per_token = (
    self.index_head_dim
    + self.index_head_dim // self.indexer_quant_block_size * 4
)
self.size = anchor_host.size
self.page_num = anchor_host.page_num

# page 维优先布局：每层 stride = 单 token 大小 × page_size × dtype 宽度
self.indexer_page_stride_size = (
    self.indexer_size_per_token * self.page_size * self.indexer_dtype.itemsize
)
self.indexer_layout_dim = self.indexer_page_stride_size * self.layer_num
self.indexer_page_num = (self.size + self.page_size + 1) // self.page_size
self.size_per_token = (
    self.indexer_size_per_token * self.layer_num * self.indexer_dtype.itemsize
)

# 分配前先对照 host 内存预算检查，超预算直接报错而不是等 OOM
buf_elem_size = self.page_num * self.layer_num * self.indexer_page_stride_size
requested_bytes = buf_elem_size * self.indexer_dtype.itemsize
available_bytes = host_memory_budget_bytes()
if requested_bytes > available_bytes:
    raise ValueError(
        f"Not enough host memory for DSA indexer hierarchical cache. "
        f"Requesting {requested_bytes / 1e9:.2f} GB but only have "
        f"{available_bytes / 1e9:.2f} GB free."
    )
draft_layer_num = self.layer_num - self.target_layer_num
if draft_layer_num > 0:
    logger.info(
        "Allocating %.2f GB host memory for DSA indexer (layout=%s), "
```

```
                "packed MTP layers: "
                "target_layers=%d, draft_layers=%d, total_layers=%d.",
                requested_bytes / 1e9,
                layout,
                self.target_layer_num,
                draft_layer_num,
                self.layer_num,
            )
        else:
            logger.info(
                "Allocating %.2f GB host memory for DSA indexer (layout=%s).",
                requested_bytes / 1e9,
                layout,
            )
        self.init_kv_buffer()
        # 迁移后仍按运行时能力探测 JIT / write-back JIT 可用性
        self.can_use_jit = False
        self.can_use_write_back_jit = False
        self._init_write_back_staging_buffers()
        self.lock = threading.RLock()
        self.clear()
```

评论区精华

本 PR 没有实质的 review 交锋：唯一 review 来自 ispobock，状态为 APPROVED 且 body 为空。作者在 issue 评论中指出挂起的 base-c-test-8-gpu-b300 CI 任务已因 PR #35627 被禁用，并附上工作流链接，因此该挂起状态不构成合入障碍。PR body 中可提取的决策点：延续 7/N、8/N 的做法，不提供 re-export shim，强制所有调用方同步迁移；作者同时给出可复现的机械移动验证方法（git blame -C -C -C、合并基对比逐字节 diff），供后续 10/N 参考。

- Pending CI（base-c-test-8-gpu-b300）被禁用 (other): 该说明成立：对应 B300 CI 任务已在 #35627 中临时禁用，不影响本 PR 的合入判断。

风险与影响

- 风险：
  - 兼容性（中）：不提供 re-export shim，任何仍从 memory_pool_host 导入 DSAIndexerPoolHost 的代码（仓库外扩展、未同步分支）会直接 ImportError；仓库内引用已全部检查，但外部不可控。
  - 测试有效性（中）：如果 mock patch 目标停留在旧模块，测试不会真正命中迁移后的代码路径，表现为“假通过”；本 PR 已同步修改，但后续新增 DSA 相关测试必须使用 pool_host.dsa 作为 patch 目标。
  - 合并冲突（低 - 中）：memory_pool_host.py 的大段删除与 main 频繁冲突（提交历史含 5 次 merge main），该文件仍是重构热点，后续 pool/ 阶段建议继续小步提交。
  - 运行 / 性能 / 安全：无影响；模块级平台门控导入与旧位置完全等价。

- 影响：对用户和推理行为零影响（类体逐字节一致，编译、lint、CI 均通过）。结构上，memory_pool_host.py 进一步瘦身，pool_host 包新增 dsa.py，与 base/common/mha/mla/mamba 并列，向 issue #25371 的目标布局收敛。对开发者的实际影响：新增或修改 DSA 相关代码时应从 pool_host.dsa 导入；测试 mock 目标需更新。对团队协作的影响：该 PR 展示了大文件小步拆分、逐步回归验证的路径，降低了后续大重构的心理负担。
- 风险标记：无后向兼容 shim，导入路径变更影响外部引用，mock patch 目标需同步，大段删除易与 main 冲突

关联脉络

- PR #31180 [mem_cache][8/N] refactor: move MambaPoolHost to pool_host.mamba: 同一 pool_host 阶段重构系列的前一步，同样改动 memory_pool_host.py 与 staged write-back 测试的 import/patch 路径，本 PR 完全延续其模式。
- PR #35627 [CI] Temporarily disable B300 jobs: issue 评论中作者用它解释本 PR 挂起的 base-c-test-8-gpu-b300 CI 状态。