PR #30249 完整报告

sgl-project/sglang

[mem_cache][6/N] refactor: move MHA host-pool into pool_host/mha.py

合并时间：2026-07-08 20:12

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/30249

执行摘要

- 一句话：MHA host pool 类机械迁移至独立模块
- 推荐动作：值得精读。PR 展示了如何通过可复现的机械转换脚本进行一次性大规模重构，保持 git blame 追溯，是大型项目的良好实践。关注其模块拆分策略、调用点扫描方法以及循环导入的预防措施。

功能与动机

作为 mem_cache 重构（#25371）的一部分，继续将 memory_pool_host.py 按池族拆分为独立模块。上一 PR（#27273）已抽取出基础层，本 PR 将第一个具体池族（MHA）搬出，减少巨型文件、改善模块边界，并为后续 MLA、Mamba 等池族的拆分铺平道路。

实现拆解

1. 创建新文件 pool_host/mha.py，将 memory_pool_host.py 中的 MHATokenToKVPoolHost、AsymmetricMHATokenToKVPoolHost、MHATokenToKOnlyPoolHost、get_mha_host_pool_cls 及其关联的导入（sgl_kernel、jit_kernel 等）整体移入，代码体保持不变。
2. 将共享常量 _WRITE_BACK_STAGING_PAGE_CHUNK（值为 64）从 memory_pool_host.py 迁移至 pool_host/base.py，使所有 host pool 模块（MHA、MLA、Mamba、DeepSeekV4、DSA）均可从基础模块导入，避免反向依赖。
3. 删除 memory_pool_host.py 中 MHA 特有的导入（如 MHATokenToKVPool、jit_transfer_hicache_all_layer 等），保留其他池族代码。
4. 更新 12 个调用点的导入路径：5 个运行时文件（kv_cache_builder.py、hiradix_cache.py、hybrid_pool_assembler.py、decode_kvcache_offload_manager.py、aibrix_kvcache/unit_test.py）和 7 个测试 / 基准文件。其中测试文件还需调整 mock.patch 目标模块名。
5. 通过可重现的机械转换脚本（transform_move_mha_host_pool.py）确保 diff 等价性，并利用 git blame -C -C -C 保留代码溯源（新文件 99.5% 行可追溯）。

关键文件：

- python/sglang/srt/mem_cache/pool_host/mha.py（模块 MHA 池；类别 source；类型 dependency-wiring；符号 MHATokenToKVPoolHost, AsymmetricMHATokenToKVPoolHost, MHATokenToKOnlyPoolHost,

get_mha_host_pool_cls）：新模块，承载 MHA 族 host pool 所有类及工厂函数，是本次重构的核心产出。

- python/sglang/srt/mem_cache/memory_pool_host.py（模块 主机池；类别 source；类型 dependency-wiring；符号 MHATokenToKVPoolHost, MHATokenToKOnlyPoolHost, AsymmetricMHATokenToKVPoolHost, get_mha_host_pool_cls）：原宿主文件，删除了 MHA 相关代码和对应导入，精简后保留其他池族。
- python/sglang/srt/mem_cache/pool_host/base.py（模块 基类；类别 source；类型 core-logic；符号 _WRITE_BACK_STAGING_PAGE_CHUNK）：添加了共享常量 _WRITE_BACK_STAGING_PAGE_CHUNK，避免跨模块循环导入。
- python/sglang/srt/disaggregation/decode_kvcache_offload_manager.py（模块 分离部署；类别 source；类型 dependency-wiring）：调用点之一，导入从 memory_pool_host 改为 pool_host.mha。
- python/sglang/srt/mem_cache/hiradix_cache.py（模块 缓存层；类别 source；类型 dependency-wiring）：调用点之一，导入从 memory_pool_host 改为 pool_host.mha。
- python/sglang/srt/mem_cache/kv_cache_builder.py（模块 缓存构建；类别 source；类型 dependency-wiring）：调用点之一，导入从 memory_pool_host 改为 pool_host.mha。
- python/sglang/srt/mem_cache/hybrid_cache/hybrid_pool_assembler.py（模块 混合池；类别 source；类型 dependency-wiring）：调用点之一，导入从 memory_pool_host 改为 pool_host.mha。
- test/registered/unit/mem_cache/test_hicache_staged_write_back_dispatch.py（模块 测试；类别 test；类型 test-coverage）：测试文件，导入路径更新并调整 mock.patch 目标模块。
- test/registered/unit/mem_cache/test_asymmetric_mha_pool_host_unit.py（模块 测试；类别 test；类型 test-coverage）：测试文件，导入路径更新并调整 mock.patch 目标模块。

关键符号：MHATokenToKVPoolHost.init, MHATokenToKVPoolHost.get_size_per_token, MHATokenToKVPoolHost.get_ksize_per_token, MHATokenToKVPoolHost.init_kv_buffer, MHATokenToKVPoolHost._init_write_back_staging_buffers, get_mha_host_pool_cls, AsymmetricMHATokenToKVPoolHost.init

关键源码片段

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

新模块，承载 MHA 族 host pool 所有类及工厂函数，是本次重构的核心产出。

```
from __future__ import annotations

import logging
import threading

import psutil
import torch

from sglang.jit_kernel.hicache import (
```

```
    can_use_hicache_jit_kernel,
    can_use_write_back_jit_kernel,
    transfer_hicache_all_layer as jit_transfer_hicache_all_layer,
    # ... 其他 JIT 导入保持原样 ...
)
from sglang.srt.mem_cache.pool_host.base import (
    _WRITE_BACK_STAGING_PAGE_CHUNK,
    HICACHE_HOST_MEMORY_RESERVE_BYTES,
    HostKVCache,
)
from sglang.srt.mem_cache.pool_host.common import (
    ALLOC_MEMORY_FUNCS,
    get_allocator_from_storage,
)

logger = logging.getLogger(__name__)


class MHATokenToKVPoolHost(HostKVCache):
    """MHA 族主机端 KV 缓存池。

    本模块是机械重构的产物，代码体与原 memory_pool_host.py 中的实现完全一致，
    仅导入路径和模块前缀做了调整。原代码位于 memory_pool_host.py 第 89-376 行。
    """

    device_pool: MHATokenToKVPool

    def __init__(
        self,
        device_pool: MHATokenToKVPool,
        host_to_device_ratio: float,
        host_size: int,
        page_size: int,
        layout: str,
        pin_memory: bool = True,
        device: str = "cpu",
        allocator_type: str = "default",
    ):
        # 调用基类 HostKVCache 完成通用初始化
        super().__init__(
            device_pool,
            host_to_device_ratio,
            host_size,
            page_size,
            layout,
            pin_memory,
            device,
            allocator_type,
        )
```

```
        # 计算单个 token 的元素维度，用于 JIT kernel 决策
        self.element_dim = self.device_pool.head_num * self.device_pool.head_dim
        # 检查是否可以使用 HiCache JIT kernel
        self.can_use_jit = _is_cuda and can_use_hicache_jit_kernel(
            element_size=self.element_dim * self.dtype.itemsize
        )

        if self.layout == "page_first":
            # 转置 K/V 缓冲区以获得按层连续的数据引用
            k_transposed = self.k_buffer.transpose(0, 1)
            v_transposed = self.v_buffer.transpose(0, 1)
            self.k_data_refs = [k_transposed[i] for i in range(self.layer_num)]
            self.v_data_refs = [v_transposed[i] for i in range(self.layer_num)]
        else:
            self.k_data_refs = [self.k_buffer[i] for i in range(self.layer_num)]
            self.v_data_refs = [self.v_buffer[i] for i in range(self.layer_num)]
        # 收集每层数据指针，用于 kernel 调用
        self.k_data_ptrs = torch.tensor(
            [x.data_ptr() for x in self.k_data_refs],
            dtype=torch.uint64,
            device=self.device_pool.device,
        )
        self.v_data_ptrs = torch.tensor(
            [x.data_ptr() for x in self.v_data_refs],
            dtype=torch.uint64,
            device=self.device_pool.device,
        )
        # 初始化 write-back staging 缓冲区
        self._init_write_back_staging_buffers()
```

评论区精华

自动审核机器人 gemini-code-assist[bot] 指出 memory_pool_host.py 中新增导入 _WRITE_BACK_STAGING_PAGE_CHUNK 可能不再需要（因为 MHA 类已移出）。但该常量由剩余池族（MLA、Mamba、DeepSeekV4、DSA）共享，保留导入是正确设计。维护者 hzh0425 未采纳该建议并批准 PR。

- 关于 _WRITE_BACK_STAGING_PAGE_CHUNK 导入的必要性 (correctness): PR 维护者保留了该导入，因为其他池族（MLA、Mamba、DeepSeekV4、DSA）仍在使用这个常量。评论未采纳，PR 被批准合并。

风险与影响

- 风险：纯机械重构，代码体未改动，无功能、性能、安全风险。唯一的风险是调用点导入路径遗漏更新，但已通过全仓库符号扫描确认无遗漏，且可重现脚本保证等价性。测试均通过（B200 上预存在的 JIT 测试失败与此 PR 无关）。
- 影响：影响范围：5 个运行时文件和 7 个测试 / 基准文件，涉及内存池构建、缓存路由、分离部署等功能。影响程度：对用户与系统无感知，仅为代码组织优化，为后续重构奠定基础。

无 API 变化，无需用户配合更改。

- 风险标记：机械重构，导入变更

关联脉络

- PR #27273 [mem_cache][5/N] refactor: extract host KV cache base layer into pool_host package: 本 PR 的前置步骤，提取了公共基类到 pool_host 包，为 MHA 模块的独立提供了基础。