PR #25278 完整报告

sgl-project/sglang

[HiCache] fix: Mooncake Dummy Client mode for hybrid Mamba models

合并时间：2026-05-27 10:29

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25278

执行摘要

- 一句话：修复 Mooncake Dummy Client 模式未包含 Mamba 混合缓冲区
- 推荐动作：建议阅读：对于使用 Mooncake 分布式缓存或关注 HiCache 功能的开发者，该 PR 展示了如何处理跨进程共享内存边界计算。特别是 _standalone_required_bytes 的设计体现了对多种内存池结构的兼容。新增的测试模式（伪造模块注入）值得借鉴。

功能与动机

This PR fixes Mooncake standalone / Dummy Client mode for hybrid Mamba models. Previously, setup_dummy() only sized the dummy-client shared mapping based on the KV host pool. However, hybrid Mamba models also register additional host-side buffers later through register_mem_host_pool_v2(PoolName.MAMBA), including temporal and convolution state buffers. In Dummy Client mode, the real Mooncake client runs in a separate process, so those buffers must be included in the shared memory range during setup_dummy().

实现拆解

1. 在 MooncakeStore 中新增 _standalone_required_bytes() 静态方法，用于计算 standalone 模式下需要共享的总字节数。该方法内部定义 _add_tensor() 辅助函数，对每个 tensor 通过 data_ptr 去重后累加其字节数。
2. 计算逻辑：首先通过 kv_buffer 属性加入 KV 锚点缓冲区；然后遍历 mem_pool.entries 中的每个 entry，对每个 entry 的 host_pool 获取其 kv_buffer 和通过 get_hybrid_pool_buffer() 获取的混合池缓冲（如 Mamba 的 temporal 和 conv 缓冲）；若无 entries，则直接尝试调用 get_hybrid_pool_buffer() 方法获取混合池缓冲。
3. 在 __init__() 的 standalone 分支中，将原来的 mem_pool.size * mem_pool.size_per_token 替换为调用 self._standalone_required_bytes(mem_pool) 的结果，保证所有主机缓冲区都被计入共享映射。
4. 新增单元测试文件 test_mooncake_standalone_dummy_mamba.py，通过伪造 Mooncake 模块和分布式存储类，验证在 standalone 模式下 setup_dummy 接收的 required_bytes 正确包含了 KV 和 Mamba 缓冲区的总大小，并且不会调用 setup() 路径。

关键文件：

- python/sglang/srt/mem_cache/storage/mooncake_store/mooncake_store.py（模块 缓存层；类别 source；类型 core-logic；符号 _standalone_required_bytes, _add_tensor）：核心源码变更，新增 _standalone_required_bytes 方法并修改 __init__调用。
- test/registered/unit/mem_cache/test_mooncake_standalone_dummy_mamba.py（模块 单元测试；类别 test；类型 test-coverage；符号 _fake_mooncake_modules, TestMooncakeStandaloneDummyMamba, test_setup_dummy_includes_hybrid_buffers, FakeMooncakeDistributedStore）：新增测试文件，验证 standalone dummy 模式包含混合缓冲。

关键符号：_standalone_required_bytes, _add_tensor, test_setup_dummy_includes_hybrid_buffers, _fake_mooncake_modules

关键源码片段

python/sglang/srt/mem_cache/storage/mooncake_store/mooncake_store.py

核心源码变更，新增 _standalone_required_bytes 方法并修改 __init__调用。

```
@staticmethod
def _standalone_required_bytes(mem_pool: Any) -> int:
    """计算 standalone 模式下需要共享的总字节数。

    在 standalone (dummy client) 模式下，真正的 mooncake_client 进程
    需要映射我们后续通过 register_buffer() 传入指针的所有主机缓冲区。
    对于 hybrid 模型，这包括 KV 和 sidecar 池（如 Mamba temporal/conv）。
    """
    total = 0
    seen_ptrs: set[int] = set()

    def _add_tensor(t: Optional[torch.Tensor]):
        """去重累加 tensor 字节数。"""
        nonlocal total
        if t is None:
            return
        try:
            ptr = int(t.data_ptr())
        except Exception:
            return
        if ptr in seen_ptrs:
            return
        seen_ptrs.add(ptr)
        total += int(t.numel() * t.element_size())

    # 总是包含锚点 KV 缓冲（如果存在）
    _add_tensor(getattr(mem_pool, "kv_buffer", None))

    # HostPoolGroup：遍历每个 entry，获取 hybrid 缓冲
    entries = getattr(mem_pool, "entries", None)
    if entries:
```

```
            for entry in entries:
                host_pool = getattr(entry, "host_pool", None)
                if host_pool is None:
                    continue
                # KV 池锚点内存已包含，但重复添加无害
                _add_tensor(getattr(host_pool, "kv_buffer", None))
                for buf in getattr(host_pool, "get_hybrid_pool_buffer", lambda: [])():
                    _add_tensor(buf)
            return total

        # 单个 HostKVCache-like 池：添加 sidecar 缓冲
        for buf in getattr(mem_pool, "get_hybrid_pool_buffer", lambda: [])():
            _add_tensor(buf)
        return total
```

test/registered/unit/mem_cache/test_mooncake_standalone_dummy_mamba.py

新增测试文件，验证 standalone dummy 模式包含混合缓冲。

```
class TestMooncakeStandaloneDummyMamba(CustomTestCase):
    def test_setup_dummy_includes_hybrid_buffers(self):
        """验证 standalone(dummy) 模式必须为 KV + Mamba 缓冲正确分配共享映射大小。"""
        import torch
        captured = {}

        # 伪造分布式存储，记录 setup_dummy 参数
        class FakeMooncakeDistributedStore:
            def setup_dummy(self, required_bytes, local_buffer_bytes, addr):
                captured["required_bytes"] = int(required_bytes)
                captured["local_buffer_bytes"] = int(local_buffer_bytes)
                captured["addr"] = addr
                return 0

            def setup(self, *args, **kwargs):
                raise AssertionError("在 standalone 模式下不应调用 setup()")

            def register_buffer(self, ptr, size):
                return 0

            def put(self, *args, **kwargs):
                return 0

            def is_exist(self, *args, **kwargs):
                return 1

            def get(self, *args, **kwargs):
                return bytes(4 * 1024)

        # 用伪造模块替换 moonlight，注入 FakeMooncakeDistributedStore
```

```
with patch.dict(
    "sys.modules",
    _fake_mooncake_modules(FakeMooncakeDistributedStore),
):
    from sglang.srt.mem_cache.hicache_storage import HiCacheStorageConfig, PoolName
    from sglang.srt.mem_cache.storage.mooncake_store import mooncake_store as mc_
    mod
    from sglang.srt.mem_cache.storage.mooncake_store.mooncake_store import
    MooncakeStore

    # 伪造内存池：包含 KV 和 Mamba 缓冲
    class FakeAllocator:
        pass

    class FakeKVPool:
        def __init__(self):
            self.kv_buffer = torch.empty((128,), dtype=torch.uint8)
            self.size = 128
            self.size_per_token = 1
            self.allocator = FakeAllocator()

    class FakeMambaPool:
        def __init__(self):
            self.temporal_buffer = torch.empty((64,), dtype=torch.uint8)
            self.conv_buffer = [torch.empty((32,), dtype=torch.uint8)]

        def get_hybrid_pool_buffer(self):
            return [self.temporal_buffer, *self.conv_buffer]

    class FakeHostPoolGroup:
        def __init__(self):
            self.kv = FakeKVPool()
            self.mamba = FakeMambaPool()
            self.entries = [
                FakeEntry(PoolName.KV, self.kv),
                FakeEntry(PoolName.MAMBA, self.mamba),
            ]

        @property
        def kv_buffer(self):
            return self.kv.kv_buffer

        @property
        def size(self):
            return self.kv.size

        @property
        def size_per_token(self):
            return self.kv.size_per_token
```

```
mem_pool = FakeHostPoolGroup()
cfg = HiCacheStorageConfig(
    tp_rank=0, tp_size=1, pp_rank=0, pp_size=1,
    attn_cp_rank=0, attn_cp_size=1, is_mla_model=False,
    enable_storage_metrics=False, is_page_first_layout=True,
    model_name="test",
    extra_config={"standalone_storage": True, "client_server_address": "127.0.0.1:
    50052"},
)

# 实例化 MooncakeStore，触发 setup_dummy
with patch.object(mc_mod, "MooncakeHostTensorAllocator", FakeAllocator):
    MooncakeStore(cfg, mem_pool)

# 验证 required_bytes 包含 KV + Mamba 缓冲
expected = (
    mem_pool.kv.kv_buffer.numel() * mem_pool.kv.kv_buffer.element_size() +
    mem_pool.mamba.temporal_buffer.numel() * mem_pool.mamba.temporal_buffer.
    element_size() +
    mem_pool.mamba.conv_buffer[0].numel() * mem_pool.mamba.conv_buffer[0].
    element_size()
)
self.assertEqual(captured["required_bytes"], expected)
```

评论区精华

Review 中 gemini-code-assist[bot] 指出 _standalone_required_bytes 缺少 mem_pool is None 的防御性检查，可能导致 AttributeError。该评论未被 resolve（合并时未采纳），但考虑到调用方始终传入非空 mem_pool，风险较低。两位 reviewer（huangtingwei9988, hzh0425）均批准了该 PR。

- 缺失 mem_pool None 检查 (correctness): 未被采纳（PR 已合并但未添加该检查），但调用方始终传入非空 mem_pool，风险可接受。

风险与影响

- 风险：
  1. 变更范围对已有路径有影响：所有使用 standalone_storage=True 的场景都会走新的 _standalone_required_bytes 计算逻辑，如果 mem_pool 结构不符合预期（缺少 entries、host_pool 或 get_hybrid_pool_buffer），且未触发 AttributeError，可能导致计算出错，但代码通过 getattr 和默认 lambda 尽可能容错。
  2. 未使用 MooncakeHostTensorAllocator 时原有路径不受影响。
  3. 缺少 mem_pool is None 检查：原版未检查，但调用方保证非空，低风险。
  4. 测试覆盖：新增了单元测试，但仅覆盖了标准场景，未测试边界情况（如空 entries、无混合缓冲等）。 - 影响：影响范围：仅影响启用 Mooncake standalone 模式并使用 hybrid 模型（如 Mamba）的用户。其他模式和模型不受影响。影响程度：修复了关键功能缺陷，确保 Dummy Client 模式下缓冲区正确注册，是正确性修复。对于非

standalone 模式无影响。 - 风险标记：缺少 nil 检查，mem_pool 结构依赖，测试未覆盖边界

关联脉络

- PR #26062 [UnifiedRadixTree]: Support L3 HiStorage framework: 引入了 get_hybrid_pool_buffer 等接口，本 PR 依赖于这些接口来定位 Mamba 混合缓冲。
- PR #26346 Add mooncake_tcp transfer backend (mooncake over TCP): 同属 Mooncake 基础设施变更，可能与本 PR 有间接关联（共享存储逻辑）。