# PR #29310 完整报告

sgl-project/sglang

[HiCache] Detect for double-free in HostKVCache

合并时间：2026-06-29 09:02

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29310

## 执行摘要

- 一句话：为 HostKVCache 添加双重释放检测
- 推荐动作：该 PR 是典型的防御性编程实践，值得相关模块开发者精读。建议关注：如何使用布尔张量实现分配状态追踪、如何通过测试模拟各种异常路径、以及如何用 benchmark 验证性能无退化。对于正在维护 KV 缓存层或内存池的工程师尤其有参考价值。

## 功能与动机

host_indices 在 HiRadixCache 和 HiCacheController 中的生命周期灵活，容易产生 double-free 问题。该 PR 添加检测机制，使 double-free 时触发断言失败。

## 实现拆解

1. 在 clear() 中初始化 slot_used 追踪数组 - 文件：python/sglang/srt/mem_cache/pool_host/base.py - 在 clear() 末尾添加 self.slot_used = torch.zeros(self.size, dtype=torch.bool)，用于记录每个 slot 的分配状态。
2. 在 alloc() 中添加 double-alloc 检测 - 在返回 select_index 之前，检查 self.slot_used[select_index] 是否有任何 True，若有则触发 AssertionError 并列出重复分配的 slot。通过后将对应 slot 标记为已使用。
3. 在 free() 中添加 double-free 检测 - 先将 indices 转移到 CPU，断言所有待释放 slot 均为已分配状态，否则触发 AssertionError 并列出未分配的 slot。断言通过后清除 slot_used 标记，再将 slots 追加回 free_slots。
4. 新增单元测试文件 - 文件：test/registered/unit/mem_cache/test_mem_pool_host.py - 包含四个测试用例：test_double_alloc（模拟 bookkeeping 损坏）、test_double_free（二次释放）、test_free_unallocated（释放未分配 slot）、test_free_after_clear（clear 后释放旧索引），均验证触发正确的断言消息。
5. 配置与集成 - 测试通过 register_cpu_ci 注册到 CI 套件 base-a-test-cpu，确保在 CPU CI 中自动运行。

关键文件：

- python/sglang/srt/mem_cache/pool_host/base.py（模块 内存池；类别 source；类型 core-logic；符号 clear, alloc, free）：核心逻辑变更，添加了 slot_used 布尔张量并在 alloc/free 中添加断言检测

- test/registered/unit/mem_cache/test_mem_pool_host.py（模块 内存池测试；类别 test；类型 test-coverage；符号 TestHostKVCache, setUp, test_double_alloc, test_double_free）：新增的单元测试文件，覆盖 double-alloc、double-free 等异常路径

关键符号：clear, alloc, free, setUp, test_double_alloc, test_double_free, test_free_unallocated, test_free_after_clear

## 关键源码片段

### python/sglang/srt/mem_cache/pool_host/base.py

核心逻辑变更，添加了 slot_used 布尔张量并在 alloc/free 中添加断言检测

```
# 省略导包和类定义，仅展示修改后的核心方法

@synchronized
def clear(self):
    # 初始化内存状态和追踪结构
    self.mem_state = torch.zeros(
        (self.size,), dtype=torch.uint8, device=self.device
    )
    self.free_slots = torch.arange(self.size, dtype=torch.int64)
    # 新增：per-slot 分配标志，True 表示已分配
    self.slot_used = torch.zeros(self.size, dtype=torch.bool)

@synchronized
def alloc(self, need_size: int) -> Optional[torch.Tensor]:
    assert need_size % self.page_size == 0, "请求大小应为页大小的倍数"
    if need_size > self.available_size():
        return None

    select_index = self.free_slots[:need_size]
    self.free_slots = self.free_slots[need_size:]

    # 断言：选中的 slot 必须全部未使用
    assert not self.slot_used[select_index].any(), (
        f"Double-alloc detected: slots already allocated: "
        f"{select_index[self.slot_used[select_index]].tolist()}."
    )
    self.slot_used[select_index] = True

    return select_index

@synchronized
def free(self, indices: torch.Tensor) -> int:
    indices_cpu = indices.cpu()
    # 断言：待释放的 slot 必须全部处于已分配状态
    assert self.slot_used[indices_cpu].all(), (
        f"Double-free detected: slots not currently allocated: "
        f"{indices_cpu[~self.slot_used[indices_cpu]].tolist()}."
```

```
        )
        self.slot_used[indices_cpu] = False
        self.free_slots = torch.cat([self.free_slots, indices_cpu])
        return len(indices)
```

## test/registered/unit/mem_cache/test_mem_pool_host.py

新增的单元测试文件，覆盖 double-alloc、double-free 等异常路径

```
import unittest
import torch
from sglang.srt.mem_cache.memory_pool import MHATokenToKVPool
from sglang.srt.mem_cache.memory_pool_host import MHATokenToKVPoolHost
from sglang.test.ci.ci_register import register_cpu_ci
from sglang.test.test_utils import CustomTestCase

register_cpu_ci(est_time=2, suite="base-a-test-cpu")

class TestHostKVCache(CustomTestCase):
    def setUp(self):
        self.page_size = 2
        # 构造一个小的 device pool 用于初始化 host pool
        self.device_pool = MHATokenToKVPool(
            size=self.page_size * 2,
            page_size=self.page_size,
            dtype=torch.float16,
            head_num=2,
            head_dim=4,
            layer_num=2,
            device="cpu",
            enable_memory_saver=False,
        )
        self.host_pool = MHATokenToKVPoolHost(
            device_pool=self.device_pool,
            host_to_device_ratio=2.0,
            host_size=0,
            page_size=self.page_size,
            layout="layer_first",
            pin_memory=False,
            device="cpu",
            allocator_type="default",
        )

    def test_double_alloc(self):
        indices = self.host_pool.alloc(4)
        # 模拟 bookkeeping 损坏：将已分配的 slot 插回 free_slots 头部
        leak = torch.tensor([int(indices[0])])
        self.host_pool.free_slots = torch.cat([leak, self.host_pool.free_slots])
        with self.assertRaises(AssertionError) as ctx:
            self.host_pool.alloc(4)
```

```
        msg = str(ctx.exception)
        self.assertIn("Double-alloc", msg)
        self.assertIn(f"[{int(leak[0])}]", msg)

    def test_double_free(self):
        indices = self.host_pool.alloc(4)
        self.host_pool.free(indices[:2])
        # indices[1] 被二次释放
        with self.assertRaises(AssertionError) as ctx:
            self.host_pool.free(indices[1:])
        msg = str(ctx.exception)
        self.assertIn("Double-free", msg)
        self.assertIn(f"[{int(indices[1])}]", msg)

    def test_free_unallocated(self):
        indices = torch.tensor([1])
        with self.assertRaises(AssertionError) as ctx:
            self.host_pool.free(indices)
        msg = str(ctx.exception)
        self.assertIn("Double-free", msg)
        self.assertIn(f"[{int(indices[0])}]", msg)

    def test_free_after_clear(self):
        indices = self.host_pool.alloc(4)
        self.host_pool.clear()
        with self.assertRaises(AssertionError) as ctx:
            self.host_pool.free(indices)
        msg = str(ctx.exception)
        self.assertIn("Double-free", msg)
        self.assertIn(str(indices.tolist()), msg)

if __name__ == "__main__":
    unittest.main()
```

## 评论区精华

Reviewer hzh0425 在审批时建议补充 benchmark 对比数据和长期稳定性测试，以验证改动不会引入异常崩溃。作者在 PR body 中提供了 benchmark 结果（吞吐量无影响），审查者认可后合并。

- 要求补充性能基准和稳定性测试 (testing): 作者在 PR body 中提供了 benchmark 截图（吞吐量无影响），审查者认可后合并。

## 风险与影响

- 风险：该 PR 仅增加断言和布尔标记，风险较低。潜在风险： 1) 新增的 slot_used 布尔张量占用少量额外内存（与 pool size 成正比），但 HostKVCache 规模通常有限，可忽略； 2) 断言检查增加了 alloc/free 路径的计算开销，但 benchmark 显示无显著影响； 3) 如果

断言逻辑存在 bug（例如在并发下 @synchronized 是否能保证正确性），但该装饰器已确保序列化访问，正确性较高。

- 影响：对用户：在调试阶段或生产环境中，若出现 double-free 会立即崩溃并给出明确错误信息，避免静默数据损坏，提升了系统可靠性和可调试性。对系统：增加可忽略的内存和计算开销。对团队：易于定位 HostKVCache 相关的 bug，降低排查成本。
- 风险标记：低性能风险，微增内存，测试覆盖边界场景，断言防御

# 关联脉络

- 暂无明显关联 PR