PR #24950 完整报告

sgl-project/sglang

fix: SGLANG_RADIX_FORCE_MISS chunk-cache passthrough

合并时间：2026-05-11 15:07

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24950

执行摘要

- 一句话：修复 chunk cache 在 FORCE_MISS 标志下崩溃
- 推荐动作：建议快速合并。这是一个明确的小范围 bugfix，修复了特定配置下的崩溃，测试覆盖到位，代码简洁。

功能与动机

DeepSeek-V4 分离式部署中，prefill worker 使用 SWAChunkCache，启用 SGLANG_RADIX_FORCE_MISS=1 后调度器首次调用 match_prefix 即崩溃。ChunkCache 没有前缀匹配树，match_prefix 已返回空结果，FORCE_MISS 标志无需额外处理。

实现拆解

1. 修改 zero_match_result 函数(python/sglang/srt/mem_cache/base_prefix_cache.py)：原逻辑通过 getattr(tree_cache, "root_node", None) 获取根节点，若为 None 则抛出 RuntimeError。新逻辑先调用 tree_cache.is_chunk_cache() 判断，若是 chunk cache 则直接原样返回 match_result，否则直接访问 tree_cache.root_node 进行置零操作。
2. 更新测试用例(test/registered/unit/mem_cache/test_radix_force_miss.py)：移除 test_no_root_node_raises 测试（验证旧异常行为），新增 test_chunk_cache_is_passthrough 测试，使用实现 is_chunk_cache 方法的 _StubChunkCache 模拟 chunk cache，验证 zero_match_result 原样返回 MatchResult。

关键文件：

- python/sglang/srt/mem_cache/base_prefix_cache.py（模块 缓存层；类别 source；类型 core-logic；符号 zero_match_result）：核心修复文件，修改 zero_match_result 函数逻辑，增加 chunk cache 短路分支。
- test/registered/unit/mem_cache/test_radix_force_miss.py（模块 缓存层；类别 test；类型 test-coverage；符号 test_no_root_node_raises, _NoRoot, test_chunk_cache_is_passthrough, _StubChunkCache）：更新测试以覆盖新行为，移除旧异常测试，新增 chunk cache 透传测试。

关键符号：zero_match_result

关键源码片段

python/sglang/srt/mem_cache/base_prefix_cache.py

核心修复文件，修改 zero_match_result 函数逻辑，增加 chunk cache 短路分支。

```
# python/sglang/srt/mem_cache/base_prefix_cache.py (head)

def zero_match_result(tree_cache, match_result: "MatchResult") -> "MatchResult":
    # Chunk caches (ChunkCache, SWAChunkCache) 没有前缀匹配树，
    # match_prefix 已经返回空结果，因此直接透传，不做任何修改。
    if tree_cache.is_chunk_cache():
        return match_result
    # Tree cache 有 root_node，将 match_result 的字段置零，
    # 强制后续请求缓存未命中。
    root = tree_cache.root_node
    return match_result._replace(
        # [:0] 保留原始 tensor 的 dtype 和 device，无需重新分配空 tensor。
        device_indices=match_result.device_indices[:0],
        last_device_node=root,
        last_host_node=root,
        host_hit_length=0,
    )
```

test/registered/unit/mem_cache/test_radix_force_miss.py

更新测试以覆盖新行为，移除旧异常测试，新增 chunk cache 透传测试。

```
# test/registered/unit/mem_cache/test_radix_force_miss.py (head)

class TestZeroMatchResult(unittest.TestCase):
    # 原有 test_no_root_node_raises 已被移除
    def test_chunk_cache_is_passthrough(self):
        # 模拟一个 ChunkCache 子类，仅实现 is_chunk_cache 返回 True
        class _StubChunkCache:
            def is_chunk_cache(self) -> bool:
                return True

        # 构造一个空的 MatchResult（ChunkCache 的 match_prefix 返回空结果）
        original = MatchResult(
            device_indices=torch.empty((0,), dtype=torch.int64),
            last_device_node=None,
            last_host_node=None,
            host_hit_length=0,
        )
        # 验证 zero_match_result 原样返回 original
        self.assertIs(zero_match_result(_StubChunkCache(), original), original)
```

评论区精华

无实质 review 讨论，仅有一条 gemini-code-assist 的自动评论确认无反馈。

- 暂无高价值评论线程

风险与影响

- 风险：低风险。变更局限于 zero_match_result 单函数，对于 tree cache（RadixCache，SwaRadixCache）行为不变；chunk cache 路径改为直接返回 MatchResult，而 match_prefix 在 chunk cache 上已返回空结果，因此语义正确。需要确保所有 chunk cache 子类均正确实现了 is_chunk_cache 方法。
- 影响：影响范围：仅影响使用 SGLANG_RADIX_FORCE_MISS=1 环境变量且缓存后端为 ChunkCache/SWAChunkCache 的用户。影响程度：修复崩溃 bug，对 tree cache 用户无影响。
- 风险标记：暂无

关联脉络

- 暂无明显关联 PR