PR #27913 完整报告

sgl-project/sglang

[Test] Add unit tests for srt/mem_cache/utils.py

合并时间：2026-06-13 11:38

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27913

执行摘要

- 一句话：为 mem_cache/utils.py 添加 46 个单元测试
- 推荐动作：此 PR 是单元测试编写的优秀范例，值得参考其如何为工具函数构建轻量级、不依赖环境的测试。团队可在其他核心模块中推广类似模式。重点关注其中 mock 和精确断言的使用。

功能与动机

此 PR 响应 Issue #20865（Improve Unit Test Coverage），旨在为 mem_cache/utils.py 核心逻辑添加纯 CPU 单元测试，从而在不启动服务器或加载模型的情况下验证关键工具函数，提高代码的回归防护和可测试性。

实现拆解

创建测试文件 test/registered/unit/mem_cache/test_mem_cache_utils.py，分步覆盖以下功能：

1. 驱逐策略工厂测试：通过 TestGetEvictionStrategy 验证所有 7 种策略（LRU、LFU、FIFO、MRU、FILO、Priority、SLRU）的正确实例化、大小写敏感性、未知策略报错以及每次调用返回新实例。
2. 哈希字符串生成测试：通过 TestGetHashStr 覆盖空列表、序列顺序敏感性、bigram 编码等价性、prior hash 链式传递、不同输入不同哈希以及 64 字符十六进制输出格式验证。
3. 哈希字符串转 int64 测试：通过 TestHashStrToInt64 覆盖无符号到有符号转换的溢出边界（2^63 处）、仅前 16 个字符有效、与 get_hash_str 的往返一致性。
4. 节点哈希值计算测试：通过 TestComputeNodeHashValues 使用 mock 创建模拟节点，验证页面粒度的哈希生成、父哈希链式传递、非对齐最终页面处理、空 key 或空 hash_value 等边界情况。
5. 哈希值分割测试：通过 TestSplitNodeHashValue 覆盖 page_size=1 和 >1 的分割、零 / 全分割边界、None 输入以及总长度不变性。
6. 自定义内存池初始化测试：通过 TestMaybeInitCustomMemPool 使用 mock 验证默认禁用路径和 Mooncake 启用路径。
7. CI 注册：使用 register_cpu_ci(est_time=8, suite='base-a-test-cpu') 将测试注册为 CPU CI 套件，确保在不依赖 GPU 的情况下运行。

关键文件：

- test/registered/unit/mem_cache/test_mem_cache_utils.py（模块 缓存工具；类别 test；类型 test-coverage；符号 TestGetEvictionStrategy, TestMaybeInitCustomMemPool, TestGetHashStr, TestHashStrToInt64）：唯一的变更文件，为 mem_cache/utils.py 添加了 46 个单元测试，覆盖所有核心工具函数，是此 PR 的核心交付物。

关键符号：get_eviction_strategy, get_hash_str, hash_str_to_int64, compute_node_hash_values, split_node_hash_value, maybe_init_custom_mem_pool

关键源码片段

test/registered/unit/mem_cache/test_mem_cache_utils.py

唯一的变更文件，为 mem_cache/utils.py 添加了 46 个单元测试，覆盖所有核心工具函数，是此 PR 的核心交付物。

```
class TestGetEvictionStrategy(CustomTestCase):
    """测试驱逐策略工厂函数 get_eviction_strategy"""

    def test_lru(self):
        # 验证 'lru' 返回 LRUStrategy 实例
        self.assertIsInstance(get_eviction_strategy('lru'), LRUStrategy)

    def test_lfu(self):
        self.assertIsInstance(get_eviction_strategy('lfu'), LFUStrategy)

    def test_fifo(self):
        self.assertIsInstance(get_eviction_strategy('fifo'), FIFOStrategy)

    def test_mru(self):
        self.assertIsInstance(get_eviction_strategy('mru'), MRUStrategy)

    def test_filo(self):
        self.assertIsInstance(get_eviction_strategy('filo'), FILOStrategy)

    def test_priority(self):
        self.assertIsInstance(get_eviction_strategy('priority'), PriorityStrategy)

    def test_slru(self):
        self.assertIsInstance(get_eviction_strategy('slru'), SLRUStrategy)

    def test_case_insensitive(self):
        # 验证策略名不区分大小写
        self.assertIsInstance(get_eviction_strategy('LRU'), LRUStrategy)
        self.assertIsInstance(get_eviction_strategy('Lru'), LRUStrategy)
        self.assertIsInstance(get_eviction_strategy('FIFO'), FIFOStrategy)

    def test_unknown_policy_raises_valueerror(self):
        # 验证未知策略名抛出 ValueError，并列出所有可用策略
```

```
        with self.assertRaises(ValueError) as ctx:
            get_eviction_strategy('nonexistent')
        msg = str(ctx.exception)
        self.assertIn('Unknown eviction policy', msg)
        # 确保错误消息包含所有支持的策略名
        for policy in ['lru', 'lfu', 'fifo', 'mru', 'filo', 'priority', 'slru']:
            self.assertIn(policy, msg)

    def test_each_call_creates_new_instance(self):
        # 验证每次调用都创建新的策略实例（非单例）
        s1 = get_eviction_strategy('lru')
        s2 = get_eviction_strategy('lru')
        self.assertIsNot(s1, s2)

class TestMaybeInitCustomMemPool(CustomTestCase):
    @patch('sglang.srt.mem_cache.utils.envs.SGLANG_MOONCAKE_CUSTOM_MEM_POOL.get')
    def test_disabled_by_default(self, mock_env_get):
        # 默认环境变量未设置时，应返回 (False, None, None)
        mock_env_get.return_value = None
        enabled, pool, pool_type = maybe_init_custom_mem_pool('cuda:0')
        self.assertFalse(enabled)
        self.assertIsNone(pool)
        self.assertIsNone(pool_type)

    @patch('sglang.srt.mem_cache.utils.envs.SGLANG_MOONCAKE_CUSTOM_MEM_POOL.get')
    @patch('sglang.srt.disaggregation.mooncake.utils.init_mooncake_custom_mem_pool')
    def test_enabled_via_env(self, mock_init, mock_env_get):
        # 环境变量设置为 'enabled' 时，应调用初始化函数并返回结果
        mock_env_get.return_value = 'enabled'
        mock_init.return_value = (True, 'mock_pool_instance', 'mooncake')
        enabled, pool, pool_type = maybe_init_custom_mem_pool('cuda:0')
        self.assertTrue(enabled)
        self.assertEqual(pool, 'mock_pool_instance')
        self.assertEqual(pool_type, 'mooncake')
        mock_init.assert_called_once_with('cuda:0')
```

评论区精华

审查评论主要来自 gemini-code-assist[bot]，提出了三点改进建议：

- 使用 self.assertRegex 替代手动长度和字符检查来验证十六进制格式。
- 对 compute_node_hash_values 的断言改用精确列表对比而非部分检查，以更严格地验证父哈希链式。
- 增加 parent.hash_value = None 的测试用例以覆盖 None 守卫。

作者采纳了这些建议，并在后续提交（如 trim tautological hash tests）中调整了测试代码。

- 测试断言改进建议 (testing): 作者接受了建议，在后续提交中调整了测试代码（如提交 'trim tautological hash tests; move ci register after imports'），最终版本已使用精准断言并添加了 None 用例。

风险与影响

- 风险：此 PR 仅新增测试文件，未修改任何源代码，因此回归风险极低。测试使用 mock 和纯 CPU 计算，不涉及 GPU 或模型加载。主要风险在于测试可能过于模拟或覆盖不全，但鉴于 46 个测试用例已覆盖所有主要函数和常见边界条件，该风险较小。
- 影响：对用户和服务无影响。对开发团队而言，增加了 mem_cache 模块的单元测试防护，便于未来重构时快速发现回归。测试注册为 CPU CI，运行时间短（预估 8 秒），不会显著增加 CI 负担。
- 风险标记：纯测试变更，无回归风险

关联脉络

- 暂无明显关联 PR