PR #27655 完整报告

sgl-project/sglang

[Unified Tree]fix compatibility bugs with eagle and unified l3

合并时间：2026-06-10 10:54

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27655

执行摘要

- 一句话：修复 Eagle 与统一 L3 缓存兼容性 bug
- 推荐动作：该 PR 变更小但关键，建议快速审核通过。重点关注确保 prefetch_key 传递方式变更不会破坏非 Eagle 场景的预取流程。合并后建议运行完整的 HiCache 测试套件（包括无 Eagle 配置）以验证无回归。新增的单元测试设计（FakeCacheController + 断言存储键类型）可复用作为类似缓存测试的示例。

功能与动机

在 Eagle 推测解码与 HiCache L3 统一缓存配合使用时，发现 prefetch_from_storage 传递的键不完整，导致预取失败。该 PR 修复了这一兼容性问题，并增加了对应的测试覆盖。

实现拆解

1. 核心逻辑修复：在 python/sglang/srt/mem_cache/unified_radix_cache.py 的 prefetch_from_storage 方法中，将 cache_controller.prefetch 调用的第三个参数从 prefetch_key.token_ids 改为 prefetch_key 本身。这样传递的是完整的 RadixKey 对象，包含 bigram 标志和原始 token 序列，确保 CacheController 能正确构造存储哈希键，而非仅使用 token ID 列表。
2. 测试配置扩展：在 test/registered/unit/mem_cache/test_unified_radix_cache_unittest.py 的 CacheConfig dataclass 中新增 is_eagle: bool = False 字段，并让 label 属性在 is_eagle=True 时附加 "eagle" 后缀。同时，build_fixture 函数将 is_eagle 参数传递给 CacheInitParams，使测试能够配置 Eagle 模式的缓存树。
3. 单元测试覆盖：新增 TestUnifiedRadixCacheEagleHiCacheStorageKey 测试类，使用 mock 对象模拟 CacheController，验证在 Eagle 配置下，执行 prefetch_from_storage 时传递的存储键是否为 RadixKey 类型且具有 is_bigram 属性，并检验哈希值与树节点一致。
4. 集成测试覆盖：在 test/registered/radix_cache/unified_radix_tree/test_unified_radix_cache_kl_dsv4.py 中新增 TestUnifiedDeepSeekV4FlashEagleHiCacheL3 类，启动带有 --speculative-algorithm EAGLE 等参数的 DeepSeek V4 Flash FP8 服务器，通过操作后检查文件存储页面数量和缓存命中率指标，确保 EAGLE + HiCache L3 实际工作。
5. CI 配置调整：将集成测试的预估运行时间从 768 秒增加到 1200 秒，以适应新增的 EAGLE L3 测试。

关键文件：

- python/sglang/srt/mem_cache/unified_radix_cache.py（模块 缓存层；类别 source；类型 core-logic；符号 prefetch_from_storage）：核心 bug 修复文件，修改了 prefetch_from_storage 方法，传递完整的 RadixKey 而非仅 token_ids。
- test/registered/unit/mem_cache/test_unified_radix_cache_unittest.py（模块 单元测试；类别 test；类型 test-coverage；符号 TestUnifiedRadixCacheEagleHiCacheStorageKey, test_l3_prefetch_uses_bigram_radix_key, FakeHostPool, alloc）：新增单元测试类 TestUnifiedRadixCacheEagleHiCacheStorageKey，验证 Eagle 模式下 L3 预取使用 bigram radix key。
- test/registered/radix_cache/unified_radix_tree/test_unified_radix_cache_kl_dsv4.py（模块 集成测试；类别 test；类型 test-coverage；符号 TestUnifiedDeepSeekV4FlashEagleHiCacheL3, setUpClass, tearDownClass, _count_file_storage_pages）：新增集成测试类 TestUnifiedDeepSeekV4FlashEagleHiCacheL3，使用真实服务器验证 Eagle+HiCache L3 文件存储缓存正确。

关键符号：prefetch_from_storage, test_eagle_l3_storage_cache_hit, test_l3_prefetch_uses_bigram_radix_key

关键源码片段

python/sglang/srt/mem_cache/unified_radix_cache.py

核心 bug 修复文件，修改了 prefetch_from_storage 方法，传递完整的 RadixKey 而非仅 token_ids。

```
# 在 UnifiedRadixCache.prefetch_from_storage 方法中
# 修复前：prefetch_key.token_ids （仅传递 token ID 列表）
# 修复后：prefetch_key （传递完整的 RadixKey 对象，包含 bigram 标记）
operation = self.cache_controller.prefetch(
    req_id,
    host_indices,
    prefetch_key, # 现在是 RadixKey 而非 list
    last_hash,
    prefix_keys,
    extra_pools=aux_xfers or None,
)
self.ongoing_prefetch[req_id] = (
    last_host_node,
    prefetch_key,
    host_indices,
    operation,
    anchor_lock_params,
    comp_xfers,
)
```

test/registered/unit/mem_cache/test_unified_radix_cache_unittest.py

新增单元测试类 TestUnifiedRadixCacheEagleHiCacheStorageKey，验证 Eagle 模式下 L3 预取使用 bigram radix key。

```
class TestUnifiedRadixCacheEagleHiCacheStorageKey(CustomTestCase):
    cfg = CacheConfig(page_size=4, components=(ComponentType.FULL,), is_eagle=True, kv_
    size=64, max_context_len=64)

    def test_l3_prefetch_uses_bigram_radix_key(self):
        # 建立缓存树并插入 token 序列
        tree, allocator, _ = build_fixture(self.cfg)
        tree.enable_storage = True
        tree.prefetch_threshold = 1
        tokens = array("q", [1, 2, 3, 4, 5, 6, 7, 8, 9])
        value = allocator.alloc(len(tokens) - 1)
        tree.insert(InsertParams(key=RadixKey(tokens), value=value))
        match = tree.match_prefix(MatchPrefixParams(key=RadixKey(tokens)))
        leaf = match.last_device_node
        self.assertTrue(leaf.key.is_bigram)
        self.assertEqual(len(leaf.hash_value), 2)

        # 替换为 mock CacheController 并触发预取
        controller = FakeCacheController()
        tree.cache_controller = controller
        tree.prefetch_from_storage("req", tree.root_node, tokens)

        # 断言存储键是完整的 RadixKey 且使用 bigram
        _, _, storage_key, _, _, _ = controller.prefetch_args
        self.assertIsInstance(storage_key, RadixKey)
        self.assertTrue(storage_key.is_bigram)
        self.assertEqual(len(storage_key), len(tokens) - 1)
        # 进一步验证哈希值与 leaf.hash_value 一致
        queried_hashes = []
        running_hash = None
        for start in range(0, len(storage_key), tree.page_size):
            running_hash = get_hash_str(storage_key[start:start + tree.page_size], running_hash)
            queried_hashes.append(running_hash)
        self.assertEqual(queried_hashes, leaf.hash_value)
```

test/registered/radix_cache/unified_radix_tree/test_unified_radix_cache_kl_dsv4.py

新增集成测试类 TestUnifiedDeepSeekV4FlashEagleHiCacheL3，使用真实服务器验证 Eagle+HiCache L3 文件存储缓存正确。

```
class TestUnifiedDeepSeekV4FlashEagleHiCacheL3(AccuracyTwoPassMixin, CustomTestCase):
    """DeepSeek V4 Flash EAGLE + HiCache L3 should load from storage."""

    page_size = 256
    input_ids = list(range(4000, 4300))
    storage_wait_timeout = 120

    @classmethod
    def setUpClass(cls):
```

```
        cls.model = DSV4_FLASH_MODEL
        cls.base_url = DEFAULT_URL_FOR_TEST
        cls.hicache_dir = tempfile.mkdtemp(prefix="hicache_l3_eagle_dsv4_")
        cls.process = popen_launch_server(
            cls.model,
            cls.base_url,
            timeout=DSV4_FLASH_LAUNCH_TIMEOUT,
            other_args=[
                "--trust-remote-code",
                "--tp-size", "4",
                "--enable-hierarchical-cache",
                "--hicache-storage-backend", "file",
                "--speculative-algorithm", "EAGLE",
                "--speculative-num-steps", "3",
                # ... 其他参数省略
            ],
            env={
                "SGLANG_ENABLE_UNIFIED_RADIX_TREE": "1",
                "SGLANG_HICACHE_FILE_BACKEND_STORAGE_DIR": cls.hicache_dir,
            },
        )

    @classmethod
    def _count_file_storage_pages(cls):
        # 统计 hicache 目录下 .bin 文件数量
        return sum(1 for f in os.listdir(cls.hicache_dir) if f.endswith(".bin"))

    @classmethod
    def _wait_for_file_storage_pages(cls, min_pages: int):
        deadline = time.monotonic() + cls.storage_wait_timeout
        while time.monotonic() < deadline:
            if cls._count_file_storage_pages() >= min_pages:
                return
            time.sleep(0.2)
        raise AssertionError(f"Timed out: {min_pages=}")

    def test_eagle_l3_storage_cache_hit(self):
        # 执行两轮 prompt 验证文件缓存
        cls._flush_cache()
        cls._generate(cls.input_ids)
        pages = cls._wait_for_file_storage_pages(1) # 等待至少 1 个文件页
        self.assertGreater(pages, 0)
```

评论区精华

无实质讨论。自动化 review（gemini-code-assist）无意见，由合著者 @hzh0425 直接批准。

- 暂无高价值评论线程

风险与影响

- 风险：风险较低。核心修改仅一行，但影响 prefetch_from_storage 路径。新增配置 is_eagle 仅在测试中使用，不影响运行时。测试覆盖了 Eagle 场景的预取键传递和哈希正确性。风险点：如果 Eagle 配置下 prefetch_key 不是 RadixKey 类型可能导致错误，但测试已验证该类型；集成测试的 CI 预估时间从 768 秒增加到 1200 秒，可能影响 CI 资源调度。
- 影响：用户：修复了 Eagle+HiCache L3 组合场景的预取功能，提高冷启动时的缓存命中率。系统：增加了一个集成测试和一个单元测试，CI 总时长增加约 7 分钟。团队：无直接影响。测试类为后续 Eagle 相关缓存测试提供了参考模板。
- 风险标记：核心缓存预取路径变更，CI 时间延长

关联脉络

- PR #27688 Fix flaky hicache l3 mmlu nightly test: 关联 HiCache L3 测试稳定性，本 PR 新增 Eagle+HiCache L3 集成测试类，继承相同的 AccuracyTwoPassMixin 基础。两个 PR 均围绕 HiCache L3 缓存测试，但本 PR 侧重于修复兼容性 bug。