PR #35130 完整报告

sgl-project/sglang

Fix NIXL cleaner grouping for hybrid cache keys

合并时间：2026-08-19 02:55

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35130

执行摘要

- 一句话：NIXL cleaner 修复 hybrid 键分组解析
- 推荐动作：值得精读。虽然 diff 很小，但包含三个可借鉴的设计点：（1）从 PoolName 枚举派生正则模式而非硬编码字符串，用“数据驱动规则”减小解析器与键生成之间的漂移；（2）按长度降序排列交替模式，优雅解决 swa 与 draft_swa 的前缀重叠；（3）测试用 subTest + 全形态枚举 + 原子删组断言，把“单测正确性”与“端到端驱逐语义”都锁死。建议同时阅读 issue #32702 中三个候选方案的权衡讨论——如果后续 NIXL cleaner 再做重构，可考虑向方案 1（构造时注入命名信息）演进，从根上消除解析器追逐键生成的问题。

功能与动机

issue #32702 明确指出 L3 cleaner 的设计保证是“按逻辑缓存条目原子驱逐”，即一个条目的所有物理分片（per-rank 文件、K/V 分片、hybrid 池组件）必须作为一个组一起删除，以维持 batch_exists 的全有或全无不变量。但 _parse_group_key() 只处理尾部裸 _k/_v 与尾部 _{tp_rank}_{tp_size} 两种形态，hybrid 组件位于 rank 后缀之后，锚定匹配失败，一个逻辑页会碎片化为多个组。后果正如 issue 所述：在 watermark 压力下 cleaner 按 mtime 独立删除碎片，batch_exists 报条目缺失而大部分字节仍占磁盘——“the worst of both: the space is not reclaimed proportionally and the entry is gone”。PR body 第一句也强调：“The NIXL FILE L3 cleaner must evict every physical shard of a logical cache entry atomically.”

实现拆解

实现分四步：

1. 构建hybrid组件后缀模式（python/sglang/srt/mem_cache/storage/nixl/nixl_cleaner.py）：新增 from sglang.srt.mem_cache.hicache_storage import PoolName 导入，从 PoolName 枚举值派生 _POOL_NAME_PATTERN（用 re.escape 并 按长度降序排序，保证 draft_swa 优先于 swa 参与匹配，避免前缀重叠误剥离），再构造编译一次的正则 _HYBRID_COMPONENT_SUFFIX_RE，匹配 _{pool}、_{pool}_k/v、_{pool}_temporal、_{pool}_conv_\d+、_{pool}_\d+ 等形态。
2. 调整 _parse_group_key 解析顺序：在既有的 K/V 后缀循环与 _RANK_SUFFIX_RE 剥离之前，先执行 _HYBRID_COMPONENT_SUFFIX_RE.sub("", stem)。因为 hybrid 物理名是 {hash}{config_suffix}{component} 且组件在 rank 之后，必须先剥组件才能让 rank 的锚定 $ 匹配重新生效。docstring 同步补充 HiCacheNixl._get_hybrid_component_keys 这一命名来源。

3. 测试重构与新增覆盖（test/registered/unit/mem_cache/test_hicache_nixl_cleaner.py）：提取 _run_single_group_cleanup() 辅助方法避免重复构造 cleaner；保留并强化原有 rank/K/V 形态测试；新增 test_parse_group_key_strips_hybrid_component_suffix（用 subTest 覆盖 kv_k、swa_k/v、mamba_temporal、mamba_conv_0、indexer_2、draft_swa、无 rank 的 deepseek_v4_c4_indexer_state_2 等 8 种形态，全部收敛到 page-a_model）与 test_tick_deletes_hybrid_components_atomically（构造 2 个 rank × 9 种物理后缀共 18 个旧键，验证一次 _tick() 全部删除且新组全部保留）。
4. 验证方式与 CI：单测在无 CUDA 依赖的 Python 3.10 harness 中运行（macOS checkout，9 passed / 8 subtests，cleaner 模块覆盖率 76%）；合并前由 ishandhanani 触发 /tag-and-rerun-ci 与 /rerun-group hicache，1-gpu-h100、2-gpu-h100、1-gpu-5090、4-gpu-h100 四组 hicache 测试全部通过。

关键文件：

- python/sglang/srt/mem_cache/storage/nixl/nixl_cleaner.py（模块 清理器；类别 source；类型 core-logic；符号 _parse_group_key, _HYBRID_COMPONENT_SUFFIX_RE, _POOL_NAME_PATTERN）：核心修复文件：_parse_group_key() 是 L3 cleaner 按逻辑条目分组驱逐的唯一入口，本次在此新增 hybrid 组件后缀剥离逻辑，并引入从 PoolName 派生的编译正则。
- test/registered/unit/mem_cache/test_hicache_nixl_cleaner.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_parse_group_key_strips_hybrid_component_suffix, test_tick_deletes_hybrid_components_atomically, test_parse_group_key_strips_rank_and_kv_suffix, test_tick_deletes_oldest_group_across_bucketed_dirs）：回归测试主力：新增 hybrid 全形态解析断言与跨桶 / 跨组件原子删组断言，并抽取 _run_single_group_cleanup() 辅助方法消除重复构造，是防本次缺陷回归的关键屏障。

关键符号：_parse_group_key, test_parse_group_key_strips_hybrid_component_suffix, test_tick_deletes_hybrid_components_atomically, _run_single_group_cleanup

关键源码片段

python/sglang/srt/mem_cache/storage/nixl/nixl_cleaner.py

核心修复文件：_parse_group_key() 是 L3 cleaner 按逻辑条目分组驱逐的唯一入口，本次在此新增 hybrid 组件后缀剥离逻辑，并引入从 PoolName 派生的编译正则。

```
# 从 PoolName 枚举派生池名交替模式，并按长度降序排列，
# 保证 draft_swa 在 swa 之前参与匹配，避免前缀重叠导致误剥离。
_POOL_NAME_PATTERN = "|".join(
    sorted((re.escape(pool.value) for pool in PoolName), key=len, reverse=True)
)


# hybrid 组件后缀包括 _swa_k / _mamba_temporal / _mamba_conv_0 /
# _indexer_2 / _draft_swa 等形态；该正则负责在 rank 与 K/V 解析
# 之前先剥离组件名，使 rank 后缀重新获得锚定匹配的机会。
_HYBRID_COMPONENT_SUFFIX_RE = re.compile(
    rf"_(?:{_POOL_NAME_PATTERN})(?:_(?:temporal|conv_\d+|[kv]|\d+))?$"
```

```
)


def _parse_group_key(name: str) -> str:
    """把 NIXL FILE 物理文件名归一到逻辑缓存键分组。

    物理名由 HiCacheNixl._get_suffixed_key（含 rank 与 K/V 后缀）以及
    HiCacheNixl._get_hybrid_component_keys（含 pool 组件后缀）生成。
    剥离顺序必须固定：先 hybrid 组件，再 K/V 后缀，最后 TP-rank 后缀。
    """
    stem = name
    stem = _HYBRID_COMPONENT_SUFFIX_RE.sub("", stem)
    for suffix in _KV_SUFFIXES:
        if stem.endswith(suffix):
            stem = stem[: -len(suffix)]
            break

    match = _RANK_SUFFIX_RE.search(stem)
    if match is not None:
        stem = stem[: match.start()]
    return stem
```

test/registered/unit/mem_cache/test_hicache_nixl_cleaner.py

回归测试主力：新增 hybrid 全形态解析断言与跨桶 / 跨组件原子删组断言，并抽取 _run_single_group_cleanup() 辅助方法消除重复构造，是防本次缺陷回归的关键屏障。

```
def test_parse_group_key_strips_hybrid_component_suffix(self):
    """所有 hybrid 组件形态都应归入同一逻辑页的清理组。"""
    names = [
        "page-a_model_7_8_kv_k", # K/V 池的非零拷贝分量
        "page-a_model_7_8_swa_k", # SWA 池 K 分量
        "page-a_model_7_8_swa_v", # SWA 池 V 分量
        "page-a_model_7_8_mamba_temporal", # Mamba 时域文件
        "page-a_model_7_8_mamba_conv_0", # Mamba 卷积片
        "page-a_model_7_8_indexer_2", # 多分量池的第 2 片
        "page-a_model_7_8_draft_swa", # 命名单分量池
        "page-a_model_deepseek_v4_c4_indexer_state_2", # 无 rank 的 DSV4 变体
    ]

    for name in names:
        with self.subTest(name=name):
            self.assertEqual(_parse_group_key(name), "page-a_model")


def test_tick_deletes_hybrid_components_atomically(self):
    """一次清理必须同时删除某个逻辑页的所有池分量与 TP rank。"""
    physical_suffixes = [
        "", # 基础文件
        "_k", "_v", # 零拷贝 K/V
```

```
        "_kv_k", "_kv_v", # K/V 池分量
        "_swa_k", "_swa_v", # SWA 池分量
        "_mamba_temporal", # Mamba 时域
        "_mamba_conv_0", # Mamba 卷积
    ]
    old_keys = [
        f"page-old_model_{rank}_2{suffix}"
        for rank in range(2)
        for suffix in physical_suffixes
    ]
    new_keys = [
        f"page-new_model_{rank}_2{suffix}"
        for rank in range(2)
        for suffix in physical_suffixes
    ]
    old_paths = [self._write_key(key, mtime=100.0) for key in old_keys]
    new_paths = [self._write_key(key, mtime=200.0) for key in new_keys]

    self._run_single_group_cleanup()
    # 旧组全部消失、新组全部保留，证明驱逐按逻辑条目原子进行
    self.assertFalse(any(os.path.exists(path) for path in old_paths))
    self.assertTrue(all(os.path.exists(path) for path in new_paths))
```

评论区精华

本 PR 的审查讨论不多但流程清晰：

- ishandhanani 审阅后直接 APPROVED（无 inline review comment），并主动触发 /tag-and-rerun-ci 与 /rerun-group hicache 验证。
- /rerun-group hicache 结果由 github-actions 机器人汇总：1-gpu-h100（2 tests）、2-gpu-h100（4 tests）、1-gpu-5090（1 test）、4-gpu-h100（2 tests）全部 ✅，覆盖 test_hicache_spec_file_storage.py、test_hicache_variants.py、test_hicache_storage_* 等存储与规格测试。
- yangweigbh 在评论中提醒“ishandhanani already approved, still needs a mem_cache CODEOWNER approval”，ishandhanani 随后以“only nixl l3 code was touched”说明改动仅限 NIXL L3 cleaner，帮助收敛审批范围。
- 值得注意：issue #32702 作者（yangweigbh）在 issue 中提出三种候选方案，其中更偏好“方案 1：构造时把命名后缀集传给 cleaner”，而本 PR 实际采用“方案 2：扩展解析器”的小 diff 路线，这一取舍在代码 review 中没有被单独挑战。
- hicache CI 组回归验证 (testing): hicache 相关 CI 全绿，验证改动未破坏 NIXL 存储与规格加载路径。
- mem_cache CODEOWNER 审批范围 (other): 通过明确改动范围收敛审批范围，PR 最终合并。
- 修复方案选型（方案 1 与方案 2 的取舍） (design): PR 选择小 diff 的方案 2，但通过从 PoolName 枚举派生模式缓解了漂移风险；review 中未被进一步挑战。

风险与影响

- 风险：主要风险点如下：

1. 解析器与键生成仍可能漂移：issue 作者明确表示“方案 2 的解析器必须一直追逐键生成的变化”，本 PR 采用方案 2。虽然 _POOL_NAME_PATTERN 从 PoolName 枚举派生、已显著缩小漂移面，但未来新增组件形态（如新的池后缀组合）时仍需要同步更新 _HYBRID_COMPONENT_SUFFIX_RE。
2. 正则误匹配边界：_HYBRID_COMPONENT_SUFFIX_RE 要求名字以 _{poolname} 结尾，对 MHA/MLA/DSA 命名（无池名）不匹配、行为不变；但若未来某个 config_suffix 恰好以池名字面结尾，存在理论误剥离风险（当前测试未覆盖该形状，例如 xxx_swa 且 swa 是配置后缀而非组件的情况）。
3. 测试环境限制：PR body 说明单测只在无 CUDA 的 Python 3.10 CPU harness 中运行，CPU 覆盖率 76%；GPU CI 的 hicache 组跑的是存储后端集成测试，并未直接执行该 cleaner 单测，存在“CI 绿但单测未在正式环境跑过”的空档。
4. 性能影响可忽略：正则模块导入时编译一次，每个文件仅多一次匹配，位于 watermark 触发的后台扫描路径，不在推理热路径。 - 影响：影响范围：

- 用户 / 系统：仅影响使用 hybrid 模型（Mamba、SWA、DSA 等多组件命名）+ --hicache-storage-backend nixl 且启用 L3 cleaner 的场景。修复后，逻辑缓存条目按组原子驱逐，batch_exists 不再出现“条目缺失但磁盘未被按比例回收”的最坏状态。对 MHA/MLA/DSA 纯非 hybrid 场景，正则不匹配、行为完全不变。
- 性能：不在推理热路径，后台扫描每文件一次正则匹配，开销可忽略。
- 团队：改动量小（2 文件 +75/-18），依赖新增仅 PoolName 枚举导入；审查与审批流程规范，CI hicache 组全绿。后续维护者需要意识到 cleaner 解析与键生成之间的隐式契约，新增池类型时应同步检查本正则。
- 风险标记：解析器与键生成存在再次漂移风险，测试仅在 CPU harness 运行，正则误匹配边界未完全覆盖

关联脉络

- PR #35164 Refactor kv cache event mixin into a recorder: 同一 mem_cache 子系统的重构，NIXL cleaner 依赖的 PoolName 与 hicache_storage 同属该域，两者共同收敛缓存内部组件的边界与职责。
- PR #31180 [mem_cache][8/N] refactor: move MambaPoolHost to pool_host.mamba: hicache/Mamba 池重构系列，涉及 MambaPoolHost 与 hybrid 池（Mamba/SWA）命名体系，与本次修复的 hybrid 组件后缀同源。