PR #48758 完整报告

vllm-project/vllm

[PD][NixlPush][Bugfix] Fix prefix caching

合并时间：2026-08-07 22:21

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/48758

执行摘要

- 一句话：修复 NixlPush 前缀缓存裁剪方向，消除静默 KV 损坏
- 推荐动作：值得精读。这是一个教科书式的正确性修复：先定位静默损坏根因（front-trim 方向错误），再通过抽象统一（decode/prefill 双端视角、kernel 块粒度）让 pull/push 复用同一实现，最后用只 stub NIXL WRITE 的测试把裁剪行为钉死在真实路径上。重点学习 _apply_prefix_caching 的输入契约设计（显式传两侧比例、明确 kernel 粒度）以及 review 驱动测试改进的过程。

功能与动机

PR body 明确指出：在 push 模式下 P 需要无视本地前缀缓存、把完整序列发给 D，而 D 只注册本地未计算的块（get_unhashed_block_ids_all_groups），因此 P 必须 tail-trim 只发送最后 N 个未计算块。当前实现在 push_worker.py:634-640 用 min 长度做 head-trim，部分命中时会把 P 的 prefix 块写进 D 的 suffix 槽位——这正是 Issue 48633 中 C3 所描述的场景："on a partial prefix-cache hit, D registers only its uncached suffix but the WRITE aligns via a min-length front-trim, sending P's prefix blocks into D's suffix slots"，被标为 P0 级静默 KV 损坏。PR body 还强调不应处理 elif num_local < num_remote: 分支，因为 P 必须预填充至少与 D 请求等量的块。

实现拆解

1. 重构 _apply_prefix_caching 为 decode/prefill 双端视角（base_worker.py）：参数从 (local_block_ids, remote_block_ids, remote_physical_per_logical) 改为 (decode_block_ids, prefill_block_ids, decode_physical_per_logical, prefill_physical_per_logical)，明确输入是已按各自比例展开的 kernel（物理）块 ID，裁剪发生在 kernel 粒度。非 Mamba 分支行为等价（对 prefill 做 end-trim 到 decode 长度）；Mamba 分支不再读取 self._physical_blocks_per_logical_kv_block，改为显式入参，使函数不依赖 worker 自身状态，pull 与 push 可安全复用。
2. 前缀缓存处理下沉到 kernel 块粒度（push_worker.py）：采纳 review 意见，把裁剪逻辑从 _do_start_push_kv（逻辑块层面、在 _logical_to_kernel_block_ids 之前）移除，移入 _xfer_blocks 中 _map_block_ids_for_block_size_ratio 之后、_compute_desc_ids 之前的位置。调用时 D 注册的未计算块作为 decode_block_ids、P 的完整序列作为 prefill_block_ids，实现 end-trim，兼容异构 physical-per-logical。

3. 删除破坏性的 front-trim 并强化断言（push_worker.py）：删除原逐组 min 对齐循环（其中 num_local < num_remote 分支就是 C3 静默损坏的来源），替换为裁剪后的严格一致性断言：group 数必须相等、每组块数必须相等，不等即报错而非静默 WRITE 错位数据，与 PR body 中 "P 必须预填充至少与 D 请求等量的块 " 的结论一致。
4. 测试与配套改动：test_nixl_push_connector.py 新增 TestPushPrefixCaching（+92 行），按 review 要求驱动真实 _xfer_blocks_for_req 路径、只 stub NIXL WRITE，覆盖 partial hit end-trim（[13,14] -> [500,501]）与无命中不裁剪两个场景；test_nixl_connector_hma.py 的 4 处 _apply_prefix_caching 调用适配四参签名；pull_worker.py 同步更新调用；测试 stub 新增 _engine_ttl 字段。

关键文件：

- vllm/distributed/kv_transfer/kv_connector/v1/nixl/base_worker.py（模块 公共基类；类别 source；类型 core-logic；符号 _apply_prefix_caching）：核心共享函数 _apply_prefix_caching 从 local/remote 相对视角重构为 decode/prefill 双端视角，新增两侧 physical-per-logical 显式入参并明确 kernel 块粒度输入契约，是 pull/push 统一前缀缓存语义的基础。
- vllm/distributed/kv_transfer/kv_connector/v1/nixl/push_worker.py（模块 推送连接器；类别 source；类型 core-logic；符号 _xfer_blocks, _do_start_push_kv）：修复落点：_xfer_blocks 内用 _apply_prefix_caching 替换了导致 C3 静默损坏的 min 长度 front-trim 循环，并新增严格一致性断言；裁剪位置从逻辑块层面下沉到 kernel 块层面。
- tests/v1/kv_connector/unit/test_nixl_push_connector.py（模块 推送测试；类别 test；类型 test-coverage；符号 TestPushPrefixCaching, _worker_driving_xfer, _written_block_ids, test_partial_prefix_hit_end_trims_producer_blocks）：新增 TestPushPrefixCaching，按 review 要求驱动真实 _xfer_blocks_for_req 路径、只 stub NIXL WRITE，是验证 end-trim 修复正确性的关键测试配套。
- vllm/distributed/kv_transfer/kv_connector/v1/nixl/pull_worker.py（模块 拉取连接器；类别 source；类型 core-logic；符号 _read_blocks）：同步适配 _apply_prefix_caching 新签名，以 local 作为 decode、remote 作为 prefill 并显式传入两侧比例，保证 pull 模式行为不变。
- tests/v1/kv_connector/unit/test_nixl_connector_hma.py（模块 混合模型测试；类别 test；类型 test-coverage；符号 test_apply_prefix_caching_mamba_hybrid, test_apply_prefix_caching_ssm_prefix_cache_hit, test_apply_prefix_caching_ssm_unpairable_slots_rejected, test_mismatched_physical_per_logical_fails_with_prefix_caching）：Mamba hybrid 相关 4 处 _apply_prefix_caching 调用适配四参签名，保持 SSM/FA 组裁剪行为测试有效。

关键符号：_apply_prefix_caching, _xfer_blocks, _read_blocks, _do_start_push_kv, TestPushPrefixCaching

关键源码片段

vllm/distributed/kv_transfer/kv_connector/v1/nixl/base_worker.py

核心共享函数 _apply_prefix_caching 从 local/remote 相对视角重构为 decode/prefill 双端视角，新增两侧 physical-per-logical 显式入参并明确 kernel 块粒度输入契约，是 pull/push 统一前缀缓存语义的基础。

```
def _apply_prefix_caching(
    self,
    decode_block_ids: BlockIds,
    prefill_block_ids: BlockIds,
    decode_physical_per_logical: int,
    prefill_physical_per_logical: int,
) -> tuple[BlockIds, BlockIds]:
    """裁剪块 ID 列表，使传输只覆盖未计算的后缀（kernel 块粒度）。

    输入必须是已按各自 physical-per-logical 比例展开的 kernel（物理）
    块 ID；pull 与 push 都在展开之后调用本方法，因此裁剪发生在 kernel
    粒度。前缀命中总是在 decode（D）侧：D 只持有未计算的块，P 持有
    完整序列——pull 把本地块作为 decode 传入，push 把远端 D 的注册
    块作为 decode 传入，实现模式无关。
    """
    prefill_block_ids = list(prefill_block_ids)
    if not self._has_mamba:
        # 非 Mamba 模型：按 D 的块数对 P 做 end-trim，跳过已缓存的 prefix。
        for i, prefill_group in enumerate(prefill_block_ids):
            num_decode_blocks = len(decode_block_ids[i])
            assert num_decode_blocks <= len(prefill_group), (
                f"Group {i}: decode {num_decode_blocks} > prefill {len(prefill_group)}"
            )
            if num_decode_blocks < len(prefill_group):
                prefill_block_ids[i] = prefill_group[-num_decode_blocks:]
        return decode_block_ids, prefill_block_ids

    # Mamba hybrid：SSM 组按位置配对状态槽（trim 会破坏完整状态语义，
    # 只在数量差 1 的合法范围内取尾部），FA 组在两侧比例一致时
    # end-trim 到未计算后缀；比例不一致时只能按 min 对齐
    # （TODO：支持不同 block_size 下的前缀缓存）。
    decode_block_ids = list(decode_block_ids)
    for i, prefill_group in enumerate(prefill_block_ids):
        num_decode_blocks = len(decode_block_ids[i])
        num_prefill_blocks = len(prefill_group)
        if _is_ssm_spec(self._group_spec_types[i]):
            if num_decode_blocks == num_prefill_blocks:
                continue
            # 更长的 prefill 列表携带 D 已有的更早位置（前缀命中）取尾部；
            # 更长的 decode 列表只可能多出 D 自己重算的末位。
            assert num_decode_blocks - num_prefill_blocks <= 1, (
                f"Group {i}: unpairable SSM state slots, "
                f"decode={num_decode_blocks} prefill={num_prefill_blocks}"
            )
            num_blocks = min(num_decode_blocks, num_prefill_blocks)
```

```
            if num_decode_blocks < num_prefill_blocks:
                prefill_block_ids[i] = prefill_group[-num_blocks:]
            else:
                decode_block_ids[i] = decode_block_ids[i][:num_blocks]
        elif (
            decode_physical_per_logical == prefill_physical_per_logical
            and num_decode_blocks < num_prefill_blocks
        ):
            # FA 组部分前缀命中且两侧比例一致：end-trim。
            prefill_block_ids[i] = prefill_group[-num_decode_blocks:]
        else:
            # 分配取整会合法地留下 ppl - 1 个尾部死块，两侧数量差超过
            # 两个比例之和即视为列表不匹配，报错而非静默错位传输。
            max_padding = decode_physical_per_logical + prefill_physical_per_logical
            assert abs(num_decode_blocks - num_prefill_blocks) <= max_padding, (
                f"Group {i}: |{num_decode_blocks} - {num_prefill_blocks}| > {max_padding}"
            )
            num_blocks = min(num_decode_blocks, num_prefill_blocks)
            decode_block_ids[i] = decode_block_ids[i][:num_blocks]
            prefill_block_ids[i] = prefill_group[:num_blocks]
    return decode_block_ids, prefill_block_ids
```

vllm/distributed/kv_transfer/kv_connector/v1/nixl/push_worker.py

修复落点：_xfer_blocks 内用 _apply_prefix_caching 替换了导致 C3 静默损坏的 min 长度 front-trim 循环，并新增严格一致性断言；裁剪位置从逻辑块层面下沉到 kernel 块层面。

```
        # 前缀缓存：D 只注册了它在本地没有（未计算）的块，因此部分命中时
        # D 的块数少于 P 的完整序列。P 必须 end-trim（保留尾部未计算块），
        # 只把最后 N 块写进 D 的槽位；front-trim 会把 P 已缓存的
        # prefix 块写进 D 的 suffix 槽位，造成静默 KV 损坏（Issue 48633 C3）。
        # 两侧的块 ID 在此处已是按各自 physical-per-logical 比例展开后的
        # kernel（物理）块 ID，比例信息来自 transfer_topo。
        remote_block_ids, local_block_ids = self._apply_prefix_caching(
            decode_block_ids=remote_block_ids, # D 的注册列表 = 未计算后缀
            prefill_block_ids=local_block_ids, # P 的完整序列
            decode_physical_per_logical=remote_info.remote_physical_blocks_per_logical,
            prefill_physical_per_logical=self._physical_blocks_per_logical_kv_block,
        )

        local_block_ids = list(local_block_ids)
        remote_block_ids = list(remote_block_ids)
        # P 必须预填充至少与 D 请求等量的块（PR body 已说明不应处理
        # num_local < num_remote 的分支），因此裁剪后两侧必须严格对齐；
        # 任何不一致都直接失败，而不是悄悄 WRITE 错位数据。
        assert len(local_block_ids) == len(remote_block_ids), (
            f"push group-count mismatch for {request_id}: {len(local_block_ids)} "
            f"local vs {len(remote_block_ids)} remote groups"
        )
        for i in range(len(local_block_ids)):
```

```
assert len(local_block_ids[i]) == len(remote_block_ids[i]), (
    f"push block-count mismatch for {request_id} group {i}: "
    f"{len(local_block_ids[i])} local vs "
    f"{len(remote_block_ids[i])} remote blocks"
)
```

评论区精华

snadampal 在初次 review 中提出三个关键问题，均已在最终版本解决：

1. _apply_prefix_caching 假定输入是 kernel block ids，裁剪必须下沉到 _xfer_blocks（kernel 块粒度）而不是 _do_start_push_kv（逻辑块粒度），否则异构 physical-blocks-per-logical 下 push 模式会出错，且 _xfer_blocks 里的新断言会被触发。
2. 该函数期望 kernel 块 ID 且非 Mamba 分支不使用物理比例参数，应在 docstring 中明确输入契约。
3. 最初测试 stub 了 _xfer_blocks_for_req，导致 _xfer_blocks 内新增的断言实际未被测试覆盖，应只 stub 真正的 NIXL xfer 调用、驱动真实路径。 最终 snadampal 与 LucasWilkinson 均 approve，LucasWilkinson 表示 "LGTM; thanks for doing this!"。

- 前缀缓存裁剪必须下沉到 kernel 块粒度（_xfer_blocks）(correctness): 已采纳：裁剪移入 _xfer_blocks，在 _map_block_ids_for_block_size_ratio 之后调用，并显式传入两侧 physical-per-logical 比例。
- _apply_prefix_caching 输入契约需要明确 (documentation): 已解决：docstring 明确输入必须是已按各自比例展开的 kernel（物理）块 ID，且比例改为显式入参，补齐了契约。
- 测试应驱动真实 _xfer_blocks_for_req 路径 (testing): 已解决：重写为 _worker_driving_xfer，1:1 比例下 _compute_desc_ids 为恒等映射，直接从 make_prepped_xfer 捕获裁剪后的块 ID 进行断言。

风险与影响

- 风险：
  1. 共享函数签名变更风险：_apply_prefix_caching 从 3 参数变 4 参数并重命名语义，pull 模式与 hma 测试已同步更新，但任何未同步的调用点会直接抛 TypeError（fail fast，优于静默错误）。
  2. push 全命中边界未覆盖：当 D 全前缀命中（注册 0 块）时，_xfer_blocks 前段的 len(local_block_ids) == 0 提前返回检查发生在裁剪之前，裁剪后各 group 为空列表，将进入空 descs 的 WRITE 提交路径；pull 模式有显式的全命中 send_notif 分支，push 模式没有等价处理，且测试只覆盖了 partial hit 与 no hit，该边界行为需在真实 NIXL 上确认。
  3. Mamba hybrid 场景的断言强度：SSM 槽位配对断言（num_decode_blocks - num_prefill_blocks <= 1）与 max_padding 断言在异构 TP 取整下可能过紧，一旦误报会直接失败请求（fail-loud，优于静默损坏，但需要真实异构 TP 回归验证）。
  4. 影响面：所有启用 NixlPushMode 且开启前缀缓存的部署（llm-d / Dynamo 编排的 prefill-decode 分离）都会因该修复改变传输块集合，建议做一次精度回归。 - 影响：对用户：修复 NixlPushMode 下部分前缀命中时的静默 KV 张量损坏——这是 Issue

48633 标注的 P0/C3，直接影响 push 模式大规模部署的推理正确性。对系统：_apply_prefix_caching 成为 pull/push 共享的 kernel 粒度裁剪原语，消除了两种模式在前缀缓存语义上的分叉；新断言使块数不一致时快速失败而非产出错误 KV。对团队：为后续 roadmap（C1、C2、L1、S2 等剩余 P0/P1 项）建立了统一的前缀缓存处理基础，测试模式（stub 最小化、驱动真实路径）可供后续 kv-connector 修复参考。 - 风险标记：核心传输路径变更，P0 静默损坏修复，共享函数签名调整，全命中边界未覆盖测试，新增失败断言

关联脉络

- PR #50234 [PD][PushConnector] Record last activity of remotes to allow clean up of stale ones: 同属 PD/Nixl PushConnector 可靠性工作线，共享 push_worker.py / base_worker.py 文件；本 PR 测试 stub 新增 _engine_ttl 字段，与 50234 引入的远端引擎 TTL 机制衔接。
- PR #48534 [Bugfix][KV-transfer] MoRIIO: per-layer READ-completion barrier in wait_for_layer_load: 同为 kv-connector 传输正确性 P0 修复（READ 模式高并发精度退化），反映 kv-transfer 正确性问题在仓库中的高优先级处理脉络。
- PR #48633 NixlPushMode (WRITE) Roadmap - Reliability Issue Inventory: 本 PR 是该 Issue 中 C3（P0，静默 KV 损坏）的落地实现，Issue 中同步跟踪 C1、C2、L1、S2 等后续项。