

PR #56640 完整报告

vllm-project/vllm

[Bugfix][NIXL] Report a full prefix cache hit as finished

合并时间: 2026-09-15 11:15

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/56640>

执行摘要

- 一句话: 修复满前缀命中请求永不结束导致的 KV 泄漏
- 推荐动作: 值得精读。这是一个典型的“状态未闭环导致资源泄漏”bugfix, 其核心设计——用 `awaiting_kvs` 区分“调度器真正在等 KV”与“仅通知 producer”——非常精巧, 既修复了 `full-hit` 不释放的问题, 又避免误报导致引擎核心断言崩溃。测试参数化覆盖了通知成功 / 失败 × 等待 / 仅通知四个组合, 值得作为类似嵌套状态 bug 的测试范式参考。

功能与动机

PR body 明确指出: `_read_blocks` 在本地 block 列表为空时提前返回, 且不记录任何内容; `get_finished` 只从 `_recvng_transfers` 和 `_failed_recv_reqs` 构造 `done_recvng`, 导致请求既不在其中, 调度器也没有其他方式释放它 (`WAITING_FOR_REMOTE_KVS` 无超时)。该请求因此一直停留在 `skipped_waiting`, 同时占用分配的 blocks, 直到进程结束。实际部署中 (10 prefill / 3 decode), 重启 peer 导致 `send_notif` 半小时内失败 122 次, 每个失败请求都被弃置, 最终 3 个 decode worker 持有 73–85% KV 缓存却零运行零等待, 只能靠 liveness 探针杀死后重启恢复。

实现拆解

本 PR 通过引入 `awaiting_kvs` 标志, 在调度器与 worker 之间传递“该 recv 是否被调度器挂起等待”的语义, 从而区分两类空 recv 并分别处理。实现可拆分为以下 4 步:

1. 元数据扩展 (`vllm/distributed/kv_transfer/kv_connector/v1/nixl/metadata.py`): 在 `ReqMeta` 数据类中新增 `awaiting_kvs: bool = False` 字段, 并在 `_add_new_req`、`add_new_req_to_recv` 中透传该参数, 默认值为 `False`。这样所有 recv 元数据都能携带该标志, 且不破坏原有调用方。
2. 调度器侧语义判定 (`base_scheduler.py`、`pull_scheduler.py`、`push_scheduler.py`): `_reqs_need_recv` 的值从三元组扩展为四元组 (`request`, `block_ids`, `local_num_computed_blocks`, `awaiting_kvs`)。 `build_connector_meta` 在组装 `NixlConnectorMetadata` 时把该标志传给 `add_new_req_to_recv`。关键判定在 `pull_scheduler.update_state_after_alloc`: 只有 `num_external_tokens > 0` (确实有外部 KV 需要拉取) 时才把 `awaiting_kvs` 置为 `True`, 完整本地命中 (`num_external_tokens==0`) 保持 `False`。 `request_finished` 中用于“abort 后通知 producer 释放 blocks”的空 recv 也显式传 `False`。

3. worker 侧上报决策 (`pull_worker.py`) : `_read_blocks` 新增 `awaiting_kvs` 参数。在完整前缀缓存命中分支中, 无论 `send_notif` 是否失败, 只要 `awaiting_kvs` 为 `True`, 就把 `request_id` 以空列表形式写入 `_recving_transfers`; 这样 `get_finished / _pop_done_transfers` 会把无未完成句柄的请求视为完成并进入 `finished_recving`。反之, 仅通知类 `recv` (如 `abort` 清理、未挂起请求的 `readback`) 不写入, 避免触发 `_update_from_kv_xfer_finished` 中的断言。
4. 测试配套: `test_nixl_connector.py` 新增 `test_empty_recv_is_reported_only_when_awaited`, 参数化 `awaiting_kvs` (`True/False`) 与 `notification_fails` (`False/True`), 断言 `done_recving` 中出现请求与否严格等于 `awaiting_kvs`; `test_nixl_connector_hma.py` 新增 `test_full_local_hit_is_not_awaited_by_the_scheduler`, 验证 `num_external_tokens == 0` 时 `awaiting_kvs` 为 `False`。同时更新 `test_abort_immediately_remote_prefill_enqueue_s_empty_recv`, 断言 `abort` 路径生成的 `recv awaiting_kvs is False`。
5. CI 与合并: 由于 `kv-connector` 测试出现超时, `njhill` 曾两次增大超时后回退, 最终确认是环境波动; 分支因落后 `main` 多次被 CI 拒绝, 合并前最终 `rebase` 到最新 `main` 并通过 `Buildkite` CI。

关键文件:

- `vllm/distributed/kv_transfer/kv_connector/v1/nixl/pull_worker.py` (模块 连接器; 类别 `source`; 类型 `core-logic`; 符号 `_read_blocks`) : 核心修复点: 完整前缀缓存命中分支根据 `awaiting_kvs` 决定是否把请求写入 `_recving_transfers`, 完成上报闭环。
- `vllm/distributed/kv_transfer/kv_connector/v1/nixl/pull_scheduler.py` (模块 拉取调度; 类别 `source`; 类型 `core-logic`; 符号 `update_state_after_alloc, request_finished`) : 决定 `awaiting_kvs` 语义的关键位置: 仅当存在需要拉取的外部 `token` 时才挂起请求等待 KV, 完整本地命中保持 `RUNNING`。
- `vllm/distributed/kv_transfer/kv_connector/v1/nixl/metadata.py` (模块 元数据; 类别 `source`; 类型 `core-logic`; 符号 `ReqMeta, _add_new_req, add_new_req_to_recv`) : `ReqMeta` 新增 `awaiting_kvs` 字段并在 `add_new_req_to_recv` 中透传, 是调度器与 `worker` 之间传递该语义的载体。
- `vllm/distributed/kv_transfer/kv_connector/v1/nixl/base_scheduler.py` (模块 调度基类; 类别 `source`; 类型 `core-logic`; 符号 `build_connector_meta`) : `_reqs_need_recv` 元组扩展为四元素, `build_connector_meta` 把 `awaiting_kvs` 传递给 `metadata`。
- `vllm/distributed/kv_transfer/kv_connector/v1/nixl/push_scheduler.py` (模块 推送调度; 类别 `source`; 类型 `core-logic`; 符号 `update_state_after_alloc, request_finished`) : `push` 模式同样需要更新 `_reqs_need_recv` 元组, 所有写入点都显式传 `False` (`push` 侧不等待 `recv` 完成)。
- `tests/v1/kv_connector/unit/test_nixl_connector.py` (模块 连接器测试; 类别 `test`; 类型 `test-coverage`; 符号 `test_empty_recv_is_reported_only_when_awaited`) : 新增 `test_empty_recv_is_reported_only_when_awaited`, 参数化通知成功 / 失败与是否等待, 验证上报行为; 并强化 `abort` 路径断言。
- `tests/v1/kv_connector/unit/test_nixl_connector_hma.py` (模块 HMA 测试; 类别 `test`; 类型 `test-coverage`; 符号 `test_full_local_hit_is_not_awaited_by_the_scheduler`) : 新增 `test_full_local_hit_is_not_awaited_by_the_scheduler`, 验证 `num_external_tokens == 0`

时 awaiting_kvs 为 False, 防止误报。

关键符号: pull_worker._read_blocks, pull_scheduler.update_state_after_alloc,
pull_scheduler.request_finished, push_scheduler.update_state_after_alloc,
push_scheduler.request_finished, base_scheduler.build_connector_meta,
metadata.ReqMeta, metadata.add_new_req_to_recv,
test_empty_recv_is_reported_only_when_awaited,
test_full_local_hit_is_not_awaited_by_the_scheduler

关键源码片段

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

核心修复点: 完整前缀缓存命中分支根据 awaiting_kvs 决定是否把请求写入
_recvng_transfers, 完成上报闭环。

```
# vllm/distributed/kv_transfer/kv_connector/v1/nixl/pull_worker.py
# _read_blocks 的完整前缀缓存命中分支:
# 本地已持有全部 KV 块 (local_block_ids 各 group 均为空) ,
# 无需发起 READ 传输, 只需通知 producer (P worker) 释放其 blocks。
def _read_blocks(
    self,
    read_spec: ReadSpec,
    dst_engine_id: str,
    request_id: str,
    remote_request_id: str,
    local_xfer_side_handle: int,
    local_dram_handle: int | None,
    remote_xfer_side_handle: int,
    expected_consumers: int,
    awaiting_kvs: bool,
) -> bool:
    # ... 前置处理: block_size_ratio 映射、notif_id 构造等 ...
    notif_id = f"{remote_request_id}:{expected_consumers}".encode()

    # 完整前缀缓存命中: 本地各 group 的 local block 均为空。
    if not any(len(group) > 0 for group in local_block_ids):
        agent_name = self._remote_agents[dst_engine_id][(0, remote_rank)]
        try:
            self.nixl_wrapper.send_notif(agent_name, notif_msg=notif_id)
        except Exception as e:
            # 通知失败时也要继续, producer 会在超时后自行释放 blocks。
            self._log_failure(
                failure_type="notification_failed",
                msg="P worker blocks will be freed after timeout. "
                "This may indicate network issues.",
                req_id=request_id,
                error=e,
                dst_engine_id=dst_engine_id,
                remote_rank=remote_rank,
```

```

        remote_agent_name=agent_name,
    )
    self.xfer_stats.record_failed_notification()
    # 关键修复：仅当调度器把请求挂起在 WAITING_FOR_REMOTE_KVS
    # 并期待 finished_recving 时才上报完成。
    # 上报方式是把空列表写入 _recving_transfers,
    # 之后 get_finished 会把它当作无未完成句柄而纳入完成集合。
    # 仅通知类 recv（如 abort 清理）必须保持未上报,
    # 否则会触发调度器 _update_from_kv_xfer_finished 的断言。
    if awaiting_kvs:
        self._recving_transfers.setdefault(request_id, [])
    return True

```

... 非命中路径：继续构造 READ 传输并 post 到远端 ...

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

决定 awaiting_kvs 语义的关键位置：仅当存在需要拉取的外部 token 时才挂起请求等待 KV，完整本地命中保持 RUNNING。

```

# vllm/distributed/kv_transfer/kv_connector/v1/nixl/pull_scheduler.py
# update_state_after_alloc 中构造 _reqs_need_recv 四元组：
# (request, local_block_ids, local_num_computed_blocks, awaiting_kvs)

# 只有存在需要从拉取端得到的 token (num_external_tokens > 0) 时,
# 调度器才会把请求挂起在 WAITING_FOR_REMOTE_KVS 并等待 finished_recving。
# 若完整命中 (num_external_tokens == 0)，请求保持 RUNNING,
# worker 不能上报 finished，否则会触发调度器断言。
self._reqs_need_recv[request.request_id] = (
    request,
    local_block_ids,
    local_num_computed_blocks,
    num_external_tokens > 0,
)

```

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

ReqMeta 新增 awaiting_kvs 字段并在 add_new_req_to_recv 中透传，是调度器与 worker 之间传递该语义的载体。

```

# vllm/distributed/kv_transfer/kv_connector/v1/nixl/metadata.py
# ReqMeta 新增 awaiting_kvs 字段，默认 False。
# 只有调度器把请求挂起在 WAITING_FOR_REMOTE_KVS 时才为 True,
# 表示“worker 需要把该 recv 上报为 finished”；
# 仅通知类 recv 必须保持 False。
@dataclass
class ReqMeta:
    local_block_ids: BlockIds
    local_physical_block_ids: BlockIds
    tp_size: int
    dcp_size: int = 1

```

```

local_num_computed_blocks: tuple[int, ...] = ()
remote: RemoteMeta | None = None
remote_block_size: int | None = None
pp_size: int = 1
awaiting_kvs: bool = False

# add_new_req_to_recv 透传该标志到 ReqMeta,
# 使 worker 在 start_load_kv 时能读到调度器的意图。
def add_new_req_to_recv(
    self,
    request_id: ReqId,
    local_block_ids: BlockIds,
    kv_transfer_params: dict[str, Any],
    local_num_computed_blocks: tuple[int, ...] = (),
    awaiting_kvs: bool = False,
):
    req = self._add_new_req(
        local_block_ids,
        kv_transfer_params,
        local_num_computed_blocks,
        awaiting_kvs,
    )
    # ... 组装 RemoteMeta 并写入 self.reqs_to_recv[request_id] ...

```

评论区精华

review 中唯一实质讨论来自 njhill 关于测试超时的观察：

“the kv connector test seems to be timing out and it's likely due to the new tests added. I am bumping the timeout again but we may want to look at whether we can make the new tests more efficient w.r.t. how long they take to run.”

后续提交显示 njhill 两次增加测试超时，随后又回退（revert timeout changes），最终 CI 通过，说明超时并非新增测试稳定复现，更像是环境抖动。此外，由于 PR 分支多次落后 upstream main，CI 机器人拒绝了 `/ci run`，最后通过 `/ci run --allow-stale` 在陈旧分支上运行后，合并前 rebase 到最新 main 并成功。

- kv connector 测试超时问题 (testing): njhill 两次增加超时后又回退，后续 CI 重跑通过，确认超时并非新增测试稳定复现，更可能是环境波动；未引入额外的超时配置。
- CI 分支陈旧无法启动 (other): 合并前 rebase 到最新 main，最终触发 Buildkite CI #88982 并顺利通过。

风险与影响

- 风险：主要风险点集中在状态机一致性上：
 - `_reqs_need_recv` 从三元组变为四元组，所有写入点（`pull_scheduler.update_state_after_alloc`、`pull_scheduler.request_finished`、`push_scheduler.update_state_after_alloc`、`push_scheduler.request_finished`）都必须同步传第 4 个元素，漏改会在解包时直

接报错，本 PR 已全部更新并通过测试。

- awaiting_kvs=True 时即使在 send_notif 失败后也会上报完成，意味着 producer 可能未收到通知，但其本就有超时机制自行释放 blocks，因此不会泄漏，只是通知延迟。
- 若未来新增空 recv 的调用方未正确设置 awaiting_kvs，可能重新引入卡死或断言崩溃；当前测试覆盖了 abort 与 full-hit 两条路径，但语义仍需维护者在后续扩展时留意。
- 影响：
 - 用户：修复了 KV 传输部署中满前缀命中请求“卡死”导致的级联故障，decode worker 不再因缓存耗尽而无响应。
 - 系统：调度器能够及时通过 finished_recving 释放 WAITING_FOR_REMOTE_KVS 请求，避免 skipped_waiting 无限堆积，队列深度指标恢复真实反映。
 - 团队：NixlConnectorMetadata 新增字段与 _reqs_need_recv 元组结构变化影响所有 NIXL 调度相关代码，后续改动需遵循相同约定。
- 风险标记：核心路径变更，状态机一致性，通知失败仍上报，元组结构变更

关联脉络

- PR #54714 [Bugfix][NIXL] Report a full prefix cache hit as finished: 本 PR 是该 PR 的重开版本：原分支因 force-push 孤儿化被意外关闭，GitHub 不允许直接重开，故以同一个 commit 重新提交。
- PR #56104 [Bugfix][NIXL] Fix multiple handles xfer race: 同一 NIXL kv-connector 模块的并发 / 状态修复，与本 PR 共同完善 NIXL 传输完成与失败上报的可靠性。
- PR #56486 [Bugfix][KV Offload] Ignore pending chunks in invalid sliding windows: 同为 kv-connector 调度层缺陷修复，处理调度器对远端 KV 状态错误判断导致请求无法释放的问题，与本 PR 属于同一功能演进线。