

PR #53903 完整报告

vllm-project/vllm

[NIXL][PCP] Report replicated-PCP ranks > 0 as done sending instead of hiding them

合并时间: 2026-09-10 18:58

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/53903>

执行摘要

- 一句话: NIXL 复制 PCP 下改为全体 rank 上报发送完成, 修复预填充停滞
- 推荐动作: 值得精读, 尤其适合正在做解耦服务、PCP 或 KV connector 组合的工程师。建议重点关注三点: (1) 为什么 `get_finished_count()` 返回 `None` 就等于 `world_size`, 以及它与 `MultiConnector` “任一子连接器返回 `None` 则整体返回 `None`” 的耦合; (2) 「立即 ack 代替真实传输」这种补偿式实现如何在 AND 聚合下保持完成时刻不变; (3) 评审把状态从 connector 下沉到 worker 的取舍——connector 保持无角色分支, 便于 scheduler 与 worker 走同一套逻辑。同时建议在合入后补上 body 中承诺的 E2E 基准 (GLM-5.2-FP8 PCP8 + `MultiConnector`, 并发 16/32/64), 并确认 rank > 0 重复上报在聚合侧是否幂等。

功能与动机

PR body 描述的现象非常具体: `MultiConnector.get_finished_count()` 在任一子连接器返回 `None` 时就返回 `None`, 于是 `KVOutputAggregator` 期望每个请求收到 `world_size` 份完成上报; 同时 `MultiConnector` 为每个请求额外记一次异步保存, 只有在「该 worker 上所有子连接器都上报」后才算完成, 而 PCP rank > 0 上的 NIXL 永远不上报。结果是 “Requests therefore never finish sending, their blocks are never freed and the prefill engine stalls at Running: 0 reqs, Waiting: N reqs with the KV cache full”, 在 GLM-5.2-FP8 PCP8 预填充 + `NixlConnector(kv_producer)` + `OffloadingConnector(TieringOffloadingSpec)` 的组合下, 并发 16 时 14/35 个请求完成后就再无进展。修复目标是在不改变 rank 0 真实语义 (消费者读走块后才上报) 的前提下, 让 rank > 0 也参与完成度聚合。

实现拆解

1. 定位双入口: 问题横跨 worker 侧的 `NixlConnector.get_finished()` (连接器把 rank > 0 的 `done_sending` 清空) 与 scheduler 侧的 `NixlConnector.get_finished_count()` (producer + prefill_context_parallel_size > 1 时返回 `TP × PP`)。两者共同构成「隐藏 rank > 0」的策略, 而该策略与 `MultiConnector` 的 `world_size` 聚合假设冲突。
2. 放开聚合粒度: `connector.py` 中 `get_finished_count()` 直接 return `None`, 回到调度侧默认的 `world_size` 语义, 即 PCP rank > 0 也必须上报。这一步是修复的前置条件, 否则即使 rank > 0 上报也会被聚合层忽略。
3. 新增一次性上报状态: `base_worker.py` 的 worker `__init__` 增加 `self._replicated_pcp_done_sending: set[ReqId]`, 用于暂存「被要求发送但本 rank 不参与传输」的请求; `get_finished()` 在原有 `_get_new_notifs()` 结果上 update 该集合后立刻

clear(), 保证同一 req_id 只上报一次。

4. 在发送路径上填充状态: pull_worker.py 的 start_load_kv() 在 if self.pcp_rank > 0: 提前返回前, 把 metadata.reqs_to_send 全部写入该集合。rank 0 的路径完全不变——仍维护 _reqs_to_process / _reqs_to_send 与租约 TTL, 仍然只在 D 侧真正读完块后才上报, 因此聚合完成时刻与修复前一致。
5. 按 review 意见收敛职责: commit e71a2f78 (“Move replicated completion tracking to worker”) 把最初写在 connector.py 的 done_sending = set(worker._replicated_pcp_done_sending) + clear() 下沉到 worker, connector.get_finished() 退化为纯委托, 同时对应的注释也从 get_finished_count() 上移除 (commit 5725a126), 使调度侧与 worker 侧走同一套无分支逻辑。
6. 测试配套: tests/v1/kv_connector/unit/test_nixl_connector.py::TestNixlHandshake::test_pcp_producer_uses_canonical_replica 从「rank > 0 被隐藏」改为「所有 rank 都上报」。由于 connector.get_finished() 现在会真正调用 worker 的 get_finished(), 测试改为 mock worker.transfer_topo 与 worker._get_new_notifs (rank 0 返回 {"sent"}, rank > 0 返回空集), 断言 rank > 0 首次返回 {req_id}、第二次返回 (set(), set()), 直接验证一次性上报语义。无部署或配置 schema 变更。

关键文件:

- vllm/distributed/kv_transfer/kv_connector/v1/nixl/connector.py (模块 连接器调度; 类别 source; 类型 core-logic; 符号 NixlConnector.get_finished_count, NixlConnector.get_finished): 修复的调度侧与 worker 侧入口: get_finished_count() 删除 PCP 特判并回到默认 world_size 聚合, get_finished() 去掉按 kv_role + pcp_rank 清空 done_sending 的逻辑, 退化为对 worker 的纯委托 (+1/-16)。
- vllm/distributed/kv_transfer/kv_connector/v1/nixl/base_worker.py (模块 传输工作器; 类别 source; 类型 core-logic; 符号 NixlBaseConnectorWorker.init, NixlBaseConnectorWorker.get_finished, _replicated_pcp_done_sending): 新增 _replicated_pcp_done_sending 状态并在 get_finished() 中汇入 done_sending 后清空, 是「一次性上报」语义的实际承载点; 评审要求把该状态从 connector 搬到 worker 后, 本文件成为唯一的状态归属方。
- vllm/distributed/kv_transfer/kv_connector/v1/nixl/pull_worker.py (模块 预填充取数; 类别 source; 类型 core-logic; 符号 start_load_kv, _replicated_pcp_done_sending): 唯一填充 _replicated_pcp_done_sending 的地方: start_load_kv() 在 pcp_rank > 0 提前返回前把 metadata.reqs_to_send 全部登记为已发送, 是修复生效的关键动作 (+5/-0)。
- tests/v1/kv_connector/unit/test_nixl_connector.py (模块 单元测试; 类别 test; 类型 test-coverage; 符号 TestNixlHandshake.test_pcp_producer_uses_canonical_replica): 改写 TestNixlHandshake::test_pcp_producer_uses_canonical_replica: 文档字符串从 “Only PCP rank zero publishes and reports sending completion” 改为 “...every rank reports completion”, 并把 mock 从 worker.get_finished 换成 transfer_topo + _get_new_notifs, 断言 rank > 0 首次返回 {req_id}、第二次为空, 覆盖一次性上报语义。

关键符号: NixlConnector.get_finished_count, NixlConnector.get_finished, NixlBaseConnectorWorker.get_finished, NixlBaseConnectorWorker.init, start_load_kv

关键源码片段

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

修复的调度侧与 worker 侧入口：`get_finished_count()` 删除 PCP 特判并回到默认 `world_size` 聚合，`get_finished()` 去掉按 `kv_role + pcp_rank` 清空 `done_sending` 的逻辑，退化为对 worker 的纯委托 (+1/-16)。

```
def get_finished_count(self) -> int | None:
    # 返回 None 表示沿用调度侧默认的聚合粒度 (world_size)。
    # 旧实现在 kv_producer + PCP > 1 时返回 TP × PP，只等 PCP rank 0 上报；
    # 现在所有 rank 都会上报完成，因此不再需要收窄等待范围。
    return None

def get_finished(self, finished_req_ids: set[str]) -> tuple[set[str], set[str]]:
    """Get the finished recving and sending requests."""
    assert self.connector_worker is not None
    # connector 层保持“哑”逻辑：不再按 kv_role + pcp_rank 过滤 done_sending,
    # 复制的 PCP rank > 0 也照常上报，完成状态完全由 worker 内部维护。
    return self.connector_worker.get_finished()
```

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

唯一填充 `_replicated_pcp_done_sending` 的地方：`start_load_kv()` 在 `pcp_rank > 0` 提前返回前把 `metadata.reqs_to_send` 全部登记为已发送，是修复生效的关键动作 (+5/-0)。

```
def start_load_kv(self, metadata: NixlConnectorMetadata):
    # (略) 既有逻辑：为 metadata 中的请求复用 / 发起 handshake,
    # 并对握手已完成的请求调用 _read_blocks_for_req 发起异步读。
    while not self._ready_requests.empty():
        self._read_blocks_for_req(*self._ready_requests.get_nowait())

    if self.pcp_rank > 0:
        # 复制 KV 的 PCP 场景：只有 PCP rank 0 真正把 KV 提供给 D 侧，
        # 本 rank 没有任何数据要发送。但调度侧按 world_size 聚合完成度，
        # 且 MultiConnector 会为每个请求额外记一次“异步保存”，
        # 若这里静默返回，请求永远不会被判定发送完成，KV 块无法释放，
        # 预填充引擎就会卡在 Waiting。因此立即把本 batch 待发送的请求
        # 登记为已完成发送；rank 0 仍只在消费者读完块之后才上报。
        self._replicated_pcp_done_sending.update(metadata.reqs_to_send)
    return
```

评论区精华

讨论集中在「状态该放在哪一层」。NickLucche 先指出 `get_finished_count()` 上的新注释过于具体 (“nit: thats a bit specific comment to drop on get_finished_count now :)”)，随后给出更关键的架构意见：“I think it’s confusing to keep `_replicated_pcp_done_sending` split between connector and base_worker. Let’s keep connector very dumb here (so same logic for scheduler/worker) and just move this into base_worker.py”。作者 LucasWilkinson 以 commit [e71a2f78](#) 回应“resolved in .../e71a2f78”，把集合的读取与清空

全部搬进 worker, connector 只剩一句委托, 评审通过 (APPROVED)。NickLucche 也对整体语义保留了温和的质疑: “Agree semantics of `_expected_finished_count` is a bit weird here, but at least it's not particularly inefficient as we have to poll from the worker anyway due to offloading”——即接受当前实现, 但认为期望完成数的语义本身值得后续清理。

- `_replicated_pcp_done_sending` 状态在 connector 与 base_worker 之间分裂 (design): 作者以 commit e71a2f78 把逻辑全部下沉到 worker: `base_worker.get_finished()` 负责 update + clear, `pull_worker.start_load_kv()` 负责填充, `connector.get_finished()` 退化为单行委托; 评审随后 APPROVED。
- `get_finished_count()` 上的注释过于具体 (style): 接受建议, 最终 `get_finished_count()` 只剩 `return None`, 改动通过 commit 5725a126 (Co-authored-by: Nicolò Lucchesi) 落地。
- `_expected_finished_count` 语义是否合理 (design): 未在本次 PR 内解决, 作为遗留语义问题留待后续清理; 不影响本修复的正确性。

风险与影响

• 风险:

1. 提前确认的语义风险: `rank > 0` 把 `metadata.reqs_to_send` 全部标记为已发送, 但这些请求可能最终并未被 `rank 0` 发出 (例如请求被 abort; 代码中 `metadata.reqs_not_processed` 的存在说明 `reqs_to_send` 可能包含未真正处理的请求)。由于 `KVOutputAggregator` 对同一请求做跨 `rank` 的 AND 聚合, `rank 0` 仍是真正的完成门控, 因此不会提前释放块, 但 `rank > 0` 会为这些 id 持续产生冗余上报。
2. 重复上报与集合语义: `start_load_kv()` 每个 step 都会被调用, 同一个仍处于 `reqs_to_send` 的请求会在后续 step 被再次 update 并被 `get_finished()` 再次上报。集合本身幂等, 但若调度侧按「收到一次就计数一次」处理, 理论上可能造成计数偏差; PR 中未提供这方面的说明或测试, 属于需要在 CI/ 线上观察的点。
3. 故障模式由「静默漏报」变为「依赖全员上报」: `get_finished_count()` 返回 `None` 后, 聚合严格等待 `world_size` 份上报。如果 `rank > 0` 的 `start_load_kv()` 因异常未走到填充分支 (例如 `metadata` 为空、`rank` 判定异常), 请求将阻塞而非提前放行, 故障从数据正确性转向可用性。
4. 验证证据不足: PR body 的 Test Result 明确写着 “After: pending — will update with the benchmark run”, 即合并时只有单元测试覆盖, E2E 吞吐 / 无停滞结论尚未回填, 缺少 PCP8 + MultiConnector 的回归数据。
5. 兼容性面: 改动只涉及 NIXL 连接器内部完成度上报, 不影响 KV 布局 (仍为 LBHNC)、不涉及传输数据面, 对不使用 PCP 或不使用 MultiConnector 的部署无行为变化。- 影响: 用户侧: 修复了使用 PCP (prefill context parallel) + 复制 KV + MultiConnector (尤其 OffloadingConnector/TieringOffloadingSpec) 做解耦服务时的硬性挂起, 这类部署此前在并发上升后会出现“预填充引擎 Running: 0 reqs, Waiting: N reqs、KV cache 打满”的死锁式表现。系统侧: 改变的是完成度聚合契约——NIXL 在 producer + PCP 下不再自行降低期望完成数, 而是与其他连接器保持一致的 `world_size` 语义, 这让它能在 MultiConnector 中与其他「每 rank 保存」的连接器共存。团队侧: 这是一次典型的「连接器组合语义」修复, 后续任何新增连接器 (特别是只在部分 rank 上有实际动作的连接器) 都需要遵守同一套上报假设, 否则会重现同类挂起。- 风险标记: 核心发送

完成路径变更，PCP + MultiConnector 组合场景，缺少 E2E 基准结果（body 标注 pending），提前确认发送的语义依赖 AND 聚合兜底，重复批次上报的幂等性未验证

关联脉络

- PR #56090 [Frontend][EPD] Use JSON arrays for multimodal metadata: 同属 kv-connector / 解耦服务（disaggregated）传输链路，本 PR 修的是同一链路中 KV 连接器完成度上报的记账问题，建议对照理解解耦场景下元数据与完成度传递的约定。
- PR #53885 [CI][ROCm][Disagg] Add GLM-5.2-FP8 to MoRIIO model catalog: 本 PR 的 E2E 验证场景正是 GLM-5.2-FP8 预填充 + PCP8 + MultiConnector 的解耦部署，与 AMD 解耦 CI 目录维护的同一模型 / 场景线相关，可作为复现该挂起的环境参照。