PR #45880 完整报告

vllm-project/vllm

[KVConnector][NIXL] Support pipeline-parallel prefill in push mode

合并时间：2026-07-09 07:49

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/45880

执行摘要

- 一句话：NIXL push 连接器新增 PP prefill 支持
- 推荐动作：值得精读。重点关注三类设计决策：notif-only 握手如何避免 D 端 descriptor 注册、P 端按层切片如何保持 region 列表同构、MLA 场景下如何把复制语义落到 ReadSpec fan-out。同时建议阅读 review 中关于 writer 轮询必要性的讨论，它展示了'引擎保活机制 vs 额外轮询'的权衡取舍。HMA 支持请跟进 #47981。

功能与动机

PR body 明确指出，为了让 PP 分片的 prefiller 能把 KV 推送给非 PP 的 decoder，push 方案相比之前的 pull 方案（#43366）更简单：去掉了 prefill 与 decode 之间 PP-aware 握手，且 push 模式未来可支持逐层（stage by stage）KV 传输。验证场景为 non-MLA（Llama-3.1-70B）与 MLA（DeepSeek-V2-Lite）下 PP=2 → TP=2 的解耦推理，gsm8k 精度与直接 prefill 无差异。

实现拆解

1. 统一身份模型：vllm/distributed/kv_transfer/kv_connector/v1/nixl/base_worker.py 中 _remote_agents 的键从 tp_rank: int 改为 (pp_rank, tp_rank) 元组；base_scheduler.py 的 set_xfer_handshake_metadata 与 side-channel 线上格式从 (GET_META_MSG, rank) 扩展为 (GET_META_MSG, pp_rank, tp_rank)，使每个 PP stage 的 agent metadata 都能被查询；connector.py 新增 pp-aware 入口 set_xfer_handshake_metadata_pp_aware 做转发。
2. P 端区间切片：NixlBaseConnectorWorker.__init__ 记录 pp_size 并初始化 _remote_region_offset；在 add_remote_agent 路径上，当 pp_size > 1 时按 regions_per_layer * start_layer 计算本 stage 对应的远端 region 子区间，保持本地与远端 region 列表同构，既有 descriptor 构建与校验逻辑不用改。同时显式拒绝两种不支持组合：HMA + PP > 1 与 decode 侧 PP > 1，避免隐藏错误。
3. D 端 notif-only 握手：_nixl_handshake 增加 remote_pp_size 与 notif_agents_only 参数；push 模式下 D 不建立 descriptor / dst handle，只加载 P 各 stage 的 agent 以接收完成通知，并调用 transfer_topo.register_remote_engine 记录远端拓扑用于块计账。push_worker.py的_send_registration_to_p从reg_data读取remote_pp_size（默认1），并在大于 1 时启用 notif-only 握手；心跳握手同样置 _hb_handshake_notif_only。

4. 完成通知聚合与 MLA fan-out：pp_size 经 kv_transfer_params 由 P 传给 D，metadata.py 与 push_scheduler.py 将其写入 ReqMeta；D 端 _get_new_notifs 按 producer TP rank 计数，只有收到全部 pp_size 个通知（每个 stage 一个）才把 recv 标记完成，pp_size = 1 时退化为原行为。MLA 场景下 latent KV 在 D 的各 TP rank 间复制，_xfer_blocks_for_req 改为先按分支构造 ReadSpec 列表，再对 dst_xfer_side_handles[engine_id] 中的每个 D rank 各发起一次 WRITE。
5. 配套与验证：demo 代理 disagg_proxy_pushconnector_demo.py 新增 --prefill-pp-size 参数并写入 push metadata，同时修正 StreamingResponse 的 JSON media type；新增独立集成测试目录 tests/v1/kv_connector/nixl_push_integration/ 与 config_sweep_accuracy_test.sh（覆盖 non-MLA 与 MLA 的 PP=2 → TP=2 配置）；单元测试新增 TestPushPipelineParallel、TestPushWriterMlaReplication。gsm8k 全量（1319 题）与 MLA 100 题对比均与基线在噪声内一致，0 传输错误。

关键文件：

- vllm/distributed/kv_transfer/kv_connector/v1/nixl/base_worker.py（模块 工作器基类；类别 source；类型 core-logic；符号 _add_notif_only_remote_agent, _nixl_handshake, add_remote_agent）：核心改造文件：远程 agent 按 (pp_rank, tp_rank) 管理，新增 notif-only 握手分支与 PP 能力校验，是 PP push 支持的地基。
- vllm/distributed/kv_transfer/kv_connector/v1/nixl/push_worker.py（模块 推送工作器；类别 source；类型 core-logic；符号 _send_registration_to_p, _xfer_blocks_for_req, _get_new_notifs, _push_writer_loop）：推送侧核心逻辑：注册时传递 remote_pp_size 并启用 notif-only 握手，MLA 场景向 D 的全部 TP rank fan-out WRITE，按 pp_size 聚合完成通知。
- tests/v1/kv_connector/unit/test_nixl_push_connector.py（模块 单元测试；类别 test；类型 test-coverage；符号 TestPushPipelineParallel, test_completion_waits_for_one_notif_per_pp_stage, test_req_meta_reads_pp_size_from_kv_transfer_params, test_add_remote_agent_slices_remote_regions_to_local_pp_window）：新增 PP 推送模式单元测试，覆盖每 stage 一个完成通知的聚合、kv_transfer_params 中 pp_size 的读取与默认值、remote region 按窗口切片以及 MLA 异构 TP 复制。
- vllm/distributed/kv_transfer/kv_connector/v1/nixl/base_scheduler.py（模块 调度器基类；类别 source；类型 data-contract；符号 on_new_request, set_xfer_handshake_metadata, _nixl_handshake_listener）：心跳信息与握手元数据改为按 (pp_rank, tp_rank) 键控，side-channel 解码扩展为三元组。
- vllm/distributed/kv_transfer/kv_connector/v1/nixl/connector.py（模块 连接器入口；类别 source；类型 data-contract；符号 set_xfer_handshake_metadata_pp_aware, set_xfer_handshake_metadata）：新增 pp-aware 握手元数据入口，键从 tp_rank 扩展为 (pp_rank, tp_rank)。
- tests/v1/kv_connector/nixl_push_integration/config_sweep_accuracy_test.sh（模块 集成测试；类别 test；类型 test-coverage）：新增 PP push 模式 e2e 准确性测试脚本，覆盖 non-MLA 与 MLA 的 PP=2 → TP=2 配置。

- examples/disaggregated/disaggregated_serving/disagg_proxy_pushconnector_demo.py（模块 示例代理；类别 source；类型 configuration）：demo proxy 新增 --prefill-pp-size 参数并把 pp_size 写入 push metadata，同时修正 JSON media type。
- vllm/distributed/kv_transfer/kv_connector/v1/nixl/push_scheduler.py（模块 推送调度器；类别 source；类型 configuration）：推送侧调度器补充 pp_size 相关字段透传。
- vllm/distributed/kv_transfer/kv_connector/v1/nixl/metadata.py（模块 元数据；类别 source；类型 data-contract）：ReqMeta 增加 pp_size 字段，供 D 端完成通知计数使用。
- vllm/distributed/kv_transfer/kv_connector/v1/nixl/pull_worker.py（模块 拉取工作器；类别 source；类型 core-logic）：适配远程 agent 键类型变化的小改动。

关键符号：_add_notif_only_remote_agent, _nixl_handshake, add_remote_agent, _send_registration_to_p, _xfer_blocks_for_req, _get_new_notifs, _push_writer_loop, set_xfer_handshake_metadata_pp_aware, set_xfer_handshake_metadata, on_new_request

关键源码片段

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

核心改造文件：远程 agent 按 (pp_rank, tp_rank) 管理，新增 notif-only 握手分支与 PP 能力校验，是 PP push 支持的地基。

```
# vllm/distributed/kv_transfer/kv_connector/v1/nixl/base_worker.py
# 初始化中的 PP 能力边界：push 模式支持 PP 生产者，但要求每层 region 数
# 均匀（HMA 不满足）且消费者不做 PP，否则在启动时直接拒绝，避免后续在
# 切片与通知聚合逻辑中出现隐性错误。
self.pp_size = vllm_config.parallel_config.pipeline_parallel_size
self._remote_region_offset = 0
if self.pp_size > 1 and self._is_hma_required:
    raise NotImplementedError(
        "NixlPushConnector does not support pipeline_parallel_size > 1 "
        "with hybrid KV cache layouts (HMA) yet."
    )
if vllm_config.kv_transfer_config.kv_role == "kv_consumer" and self.pp_size > 1:
    raise NotImplementedError(
        "NixlPushConnector consumer (decode) does not support "
        "pipeline_parallel_size > 1."
    )
# 对 PP 生产者，心跳握手也必须走 notif-only，与 PUSH_REG 路径一致。
self._hb_handshake_notif_only = False
```

```
def _nixl_handshake(self, host, port, remote_tp_size, expected_engine_id,
                    remote_pp_size=1, notif_agents_only=False):
    """与远端实例建立 NIXL 握手，返回 {(pp_rank, tp_rank): agent_name}。"""
    assert self.transfer_topo is not None
    p_remote_ranks = self.transfer_topo.handshake_target_ranks(remote_tp_size)
    remote_rank_to_agent_name: dict[tuple[int, int], str] = {}
    with zmq_ctx(zmq.REQ, make_zmq_path("tcp", host, port)) as sock:
        # side-channel 协议扩展为 (GET_META_MSG, pp_rank, tp_rank),
```

```
        # 每个 PP stage 对应一个 agent 条目。
        for remote_pp_rank, remote_rank in itertools.product(
            range(remote_pp_size), p_remote_ranks
        ):
            # ... 发送请求并解码 metadata（省略具体收发细节） ...
            if notif_agents_only:
                # Push 模式下 D 端永远不会寻址 P 的内存，只需要加载
                # P 的 agent 以接收完成通知；同时记录远端引擎拓扑，
                # 供块计账与通知计数使用。注意这里跳过 descriptor 注册。
                self.transfer_topo.register_remote_engine(
                    expected_engine_id,
                    EngineTransferInfo(...), # 字段与既有注册保持一致
                )
                remote_agent_name = self._add_notif_only_remote_agent(
                    expected_engine_id, (remote_pp_rank, remote_rank), metadata
                )
            else:
                remote_agent_name = self.add_remote_agent(
                    metadata, remote_rank, remote_tp_size
                )
            remote_rank_to_agent_name[(remote_pp_rank, remote_rank)] = remote_agent_name
    return remote_rank_to_agent_name
```

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

推送侧核心逻辑：注册时传递 remote_pp_size 并启用 notif-only 握手，MLA 场景向 D 的全部 TP rank fan-out WRITE，按 pp_size 聚合完成通知。

```
# vllm/distributed/kv_transfer/kv_connector/v1/nixl/push_worker.py
# D 端发送 PUSH_REG 注册：若远端 prefill 是 PP 分片的，握手必须
# 只加载对方 agent（notif-only），因为 push 模式下 D 从不寻址 P 的内存。
def _send_registration_to_p(self, req_id, reg_data):
    remote_pp_size = reg_data.get("remote_pp_size", 1)
    fut = self._ensure_handshake(
        reg_data["remote_engine_id"],
        reg_data["remote_host"],
        reg_data["remote_port"],
        reg_data["remote_tp_size"],
        pp_size=remote_pp_size,
        notif_agents_only=remote_pp_size > 1,
    )
    if fut is None:
        self._do_send_reg_notif(req_id, reg_data)
        return
    # ... 握手完成后回队，由 writer 线程真正发送 ...

# MLA latent 在 D 的 TP rank 间是复制的：tp-mapping 在读路径上
# 折叠为单个 rank，但 push 必须写到 D 的每个 rank，否则解码
# 会读到陈旧 KV；这里直接按目标 handle 构造 fan-out 读规格。
if self.use_mla and tp_ratio < 0:
```

```
            assert len(plan.all_source_ranks) == 1
            mla_local_ids = [list(ids) for ids in local_block_ids]
            mla_remote_ids = [list(ids) for ids in remote_block_ids]
            read_specs = [
                ReadSpec(
                    remote_rank=rank,
                    local_block_ids=mla_local_ids,
                    remote_block_ids=mla_remote_ids,
                )
                for rank in self.dst_xfer_side_handles[engine_id]
            ]
        else:
            # 非 MLA 路径保持原有基于 source_ranks_per_group 的 ReadSpec 构造。
            read_specs = [
                ReadSpec(
                    remote_rank=rank,
                    local_block_ids=[
                        list(local_block_ids[g])
                        if rank in plan.source_ranks_per_group[g]
                        else []
                        for g in range(num_groups)
                    ],
                    remote_block_ids=[
                        list(remote_block_ids[g])
                        if rank in plan.source_ranks_per_group[g]
                        else []
                        for g in range(num_groups)
                    ],
                )
                for rank in plan.all_source_ranks
            ]
```

tests/v1/kv_connector/unit/test_nixl_push_connector.py

新增 PP 推送模式单元测试，覆盖每 stage 一个完成通知的聚合、kv_transfer_params 中 pp_size 的读取与默认值、remote region 按窗口切片以及 MLA 异构 TP 复制。

```
# tests/v1/kv_connector/unit/test_nixl_push_connector.py
class TestPushPipelineParallel:
    """PP 分片生产者：按 stage 的完成通知计数、pp_size 管线、远端 region 切片。"""

    def test_completion_waits_for_one_notif_per_pp_stage(self):
        """每个 PP stage 各自 WRITE 自己的层并发送一个通知；D 必须收集齐
        pp_size 个通知后才把 recv 标记为完成。"""
        w = _StubWriterWorker.fresh()
        w.transfer_topo = MagicMock()
        request_id = "req-pp-2"
        w._recving_metadata[request_id] = MagicMock(pp_size=2)
        notif = f"{request_id}:1".encode()
```

```
        # 第一个 stage：计数，但尚未完成。
        w._pending_completion_notifs.put(notif)
        assert w._get_new_notifs() == set()
        assert request_id not in w._recving_transfers
        assert w.consumer_notification_counts_by_req[request_id] == 1

        # 第二个（最后一个）stage：此时才上报 done。
        w._pending_completion_notifs.put(notif)
        assert w._get_new_notifs() == set()
        assert request_id in w._recving_transfers
        assert request_id not in w.consumer_notification_counts_by_req

    def test_req_meta_reads_pp_size_from_kv_transfer_params(self):
        """D 从 kv_transfer_params 获取生产者的 pp_size（由代理转发），
        缺失时默认为 1。"""
        metadata = NixlConnectorMetadata()
        params = {
            "remote_block_ids": ([0],),
            "remote_engine_id": "p-engine",
            "remote_request_id": "p-req",
            "remote_host": "localhost",
            "remote_port": 1234,
            "tp_size": 1,
            "pp_size": 2,
        }
        metadata.add_new_req_to_recv("req", ([0],), params)
        assert metadata.reqs_to_recv["req"].pp_size == 2

        params.pop("pp_size")
        metadata.add_new_req_to_recv("req-default", ([0],), params)
        assert metadata.reqs_to_recv["req-default"].pp_size == 1
```

评论区精华

评审中最重要的交锋围绕三点：一、NickLucche 质疑 writer 线程轮询 WRITE 完成的改动，认为其与 PP 无关且应先用 PushScheduler.has_pending_push_work 解决，作者后续实验证明原 hang 在最新代码上无法复现、大 batch 下轮询反而负收益，最终保持回退；二、snadampal 指出完成通知计数在 TP=1 之外不正确，且简单计数会掩盖重复通知问题，作者改为按 producer TP rank 计数并说明；三、snadampal 要求显式拒绝 HMA+PP，作者添加 NotImplementedError。此外 NickLucche 要求独立 push 集成测试目录，njhill 建议拆分 notif-only 注册逻辑，均被采纳。

- writer 线程轮询 WRITE 完成的必要性 (design): 作者进一步实验：原 hang 在最新代码上无法复现，大 batch 下轮询反而负收益，决定保持回退且不单独提交。
- PP 完成通知计数的正确性（pp*tp 语义与重复通知） (correctness): 作者改为按 producer TP rank 计数（commit 5164fc9），并说明本 PR 只支持 TCPP=1 场景；snadampal 最终认可。

- HMA / hybrid 模型与 PP 的组合边界 (design): 本 PR 显式拒绝 HMA+PP；HMA 支持由 #47981 以语义化 region 匹配方式跟进。
- push 集成测试的组织 (testing): 作者新增 tests/v1/kv_connector/nixl_push_integration/ 目录与 config_sweep_accuracy_test.sh。
- 远程 agent 键的表示（tuple vs int） (style): 统一使用 (pp_rank, tp_rank) 键，非 PP 用 pp_rank=0。
- notif-only handshake 逻辑拆分 (style): 作者采纳并完成拆分与格式化。
- MLA 读规格构造顺序 (style): 作者改为先分支再构造 read_specs。

风险与影响

- 风险：
  1. 协议兼容性：side-channel 线上格式从二元组变为三元组，P 与 D 必须同版本部署，否则握手解码失败。
  2. 通知计数脆弱性：_get_new_notifs 依赖通知语义 req_id:tp_size 与 pp_size 的配合，若某 stage 通知丢失会卡住 recv；snadampal 指出的 pp:tp 标签化通知方案未落地，重复通知仍可能被掩盖。
  3. 多线程共享状态：writer 线程与主线程共享 _sending_transfers、_pending_completion_notifs 等队列，PP 下多个 stage 并发推送，锁粒度与线程安全需要持续关注。
  4. MLA fan-out 依赖：_xfer_blocks_for_req 的 MLA 分支直接遍历 dst_xfer_side_handles[engine_id]，若该集合为空会静默跳过；且 assert len(plan.all_source_ranks) == 1 依赖 tp-mapping 行为。
  5. 能力边界：HMA + PP 与 decode 侧 PP 被显式拒绝，误用会直接启动失败，属于预期内的硬边界。 - 影响：对用户：现有 push 模式用户（pp_size = 1）行为不变，新增 PP prefill 解耦场景可服务于 TTFT 优化需求。对系统：影响 NIXL kv-connector v1 路径的 connector / scheduler / worker 多层，side-channel 协议变化要求 P/D 同版本部署。对团队：确立了 push 方案优先于 pull 的方向，并为 #47981 等 HMA/hybrid 后续扩展预留了明确边界。 - 风险标记：核心路径变更（NIXL v1 connector），side-channel 协议变更需 P/D 同版本，完成通知计数依赖通知语义，漏通知会卡住，HMA+PP 显式拒绝，无降级路径，多线程共享状态（writer 线程），MLA fan-out 依赖 dst_xfer_side_handles 非空

关联脉络

- PR #43366 pull-based PP prefill solution（PR body 提及）：本 PR 的 push 方案是此前 pull 方案的替代；PR body 明确对比两者，push 去掉 PP-aware 握手并支持未来逐层传输。
- PR #43368 pull-based PP prefill 相关 PR（PR body 提及）：PR body 声明本 PR 不是 #43366 / #43368 的重复，属于同一条 prefill/decode 分离功能线的不同实现。
- PR #47981 Draft: HMA / hybrid KV layout support for PP prefill（基于 #45880）：qianlihuang 在 issue 评论中说明该 draft 基于本 PR，核心思路是用语义化 region 元数据匹配替代基于 offset 的切片，以支持 Gemma4 等 HMA 模型。