PR #50620 完整报告

vllm-project/vllm

[Bugfix][NIXL] Include transfer mode (push/pull) in the compatibility hash

合并时间：2026-08-14 12:11

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/50620

执行摘要

- 一句话：NIXL 兼容哈希加入 push/pull 模式，并对外暴露传输模式
- 推荐动作：值得精读。重点关注三个设计决策：一是把 transfer_mode 作为哈希因子而非常规运行时校验，在握手阶段就拦截协议不兼容配对；二是通过类属性 _TRANSFER_MODE 覆盖而非重复传参，避免多处调用点漏传；三是同一 PR 内完成哈希防御与 router 元数据暴露的闭环，并配套针对性单测。对做 KV transfer 或外部路由集成的读者尤其有参考价值。

功能与动机

PR body 明确指出：push（NixlPushConnector，WRITE）与 pull（NixlConnector，READ）连接器使用不兼容的传输协议，但此前没有任何机制阻止它们在 prefill/decode 之间配对，外部 router 也无法感知传输模式。该工作是从 #49230 拆出以保持其范围聚焦，并回应 iyastreb 在 #49230 的 review 反馈（discussion_r3686841923）。关联 issue vllm-project/router#187 说明：router 需要知道 push/pull 才能对 push 模式采用并行两阶段调度，避免顺序调度下多一跳消息导致性能劣化。

实现拆解

1. 哈希因子扩展（metadata.py）：compute_nixl_compatibility_hash 新增 transfer_mode: str = "pull" 参数并加入 factors dict；NIXL_CONNECTOR_VERSION 从 6 升到 7，在版本历史中记录该因子。因 push 与 pull 协议不兼容，哈希不同即可在握手时以清晰报错提前拒绝错配。
2. Worker 侧模式传播（base_worker.py / push_worker.py）：基类定义 _TRANSFER_MODE = "pull"，push worker 覆盖为 "push"；register_kv_caches 与 _register_packed_kv_cache 两处计算 compat_hash 的调用点均透传 transfer_mode=self._TRANSFER_MODE，避免漏传导致 push 被误算成 pull 哈希。
3. Scheduler侧元数据暴露（base_scheduler.py/push_scheduler.py/pull_scheduler.py）：基类 scheduler 定义 _TRANSFER_MODE = "pull"，push scheduler 覆盖为 "push"；pull 与 push 两个 request_finished 路径返回的 kv_transfer_params 字典均新增 transfer_mode 键，router#187 据此区分 READ/WRITE producer。
4. 测试与文档：新增 test_transfer_mode_changes_compatibility_hash（push/pull 哈希互不相同、相同模式一致、默认参数等于 pull）与 test_scheduler_advertises_transfer_mode（两个 scheduler 类属性断言）；test_kv_transfer_handshake 补充 pull 端到端断言 kv_connector_metadata["transfer_mode"] == "pull"。文档

docs/features/nixl_connector_compatibility.md 更新哈希因子清单。CI 已触发两次（#83649、#83813）。

关键文件：

- vllm/distributed/kv_transfer/kv_connector/v1/nixl/metadata.py（模块 KV 传输；类别 source；类型 core-logic；符号 compute_nixl_compatibility_hash, NIXL_CONNECTOR_VERSION）：核心变更文件：compute_nixl_compatibility_hash 新增 transfer_mode 因子（默认 pull），NIXL_CONNECTOR_VERSION 6→7，所有握手兼容性判定以此为准。
- vllm/distributed/kv_transfer/kv_connector/v1/nixl/base_worker.py（模块 KV 传输；类别 source；类型 core-logic；符号 NixlBaseConnectorWorker._TRANSFER_MODE, register_kv_caches, _register_packed_kv_cache）：worker 侧模式入口：基类定义 _TRANSFER_MODE="pull"，两处 KV cache 注册调用点透传 transfer_mode 计算哈希。
- vllm/distributed/kv_transfer/kv_connector/v1/nixl/base_scheduler.py（模块 KV 传输；类别 source；类型 core-logic；符号 NixlBaseConnectorScheduler._TRANSFER_MODE）：scheduler 侧模式入口：基类定义 _TRANSFER_MODE="pull"，供 kv_transfer_params 输出使用。
- vllm/distributed/kv_transfer/kv_connector/v1/nixl/push_scheduler.py（模块 KV 传输；类别 source；类型 core-logic；符号 NixlPushConnectorScheduler._TRANSFER_MODE, request_finished）：push 侧 router 支持：覆盖 _TRANSFER_MODE="push"，并在 request_finished 返回的 kv_transfer_params 中新增 transfer_mode 键。
- vllm/distributed/kv_transfer/kv_connector/v1/nixl/push_worker.py（模块 KV 传输；类别 source；类型 core-logic；符号 NixlPushConnectorWorker._TRANSFER_MODE）：push worker 覆盖 _TRANSFER_MODE="push"，使 push 侧兼容哈希与 pull 区分开。
- vllm/distributed/kv_transfer/kv_connector/v1/nixl/pull_scheduler.py（模块 KV 传输；类别 source；类型 core-logic；符号 request_finished）：pull 侧对称改动：request_finished 返回的 kv_transfer_params 同样新增 transfer_mode 键。
- tests/v1/kv_connector/unit/test_nixl_connector.py（模块 KV 传输；类别 test；类型 test-coverage；符号 test_transfer_mode_changes_compatibility_hash, test_scheduler_advertises_transfer_mode, test_kv_transfer_handshake）：测试覆盖核心行为：哈希区分 push/pull、scheduler 宣传模式、pull 端到端 kv_transfer_params 断言。
- docs/features/nixl_connector_compatibility.md（模块 文档；类别 docs；类型 documentation）：同步更新兼容性哈希因子清单，补充 NIXL transfer mode (push vs pull) 条目。

关键符号：compute_nixl_compatibility_hash, request_finished, test_transfer_mode_changes_compatibility_hash, test_scheduler_advertises_transfer_mode, test_kv_transfer_handshake

关键源码片段

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

核心变更文件：compute_nixl_compatibility_hash 新增 transfer_mode 因子（默认 pull），NIXL_CONNECTOR_VERSION 6→7，所有握手兼容性判定以此为准。

```
def compute_nixl_compatibility_hash(
    vllm_config: VllmConfig,
    attn_backend_name: str,
    cross_layers_blocks: bool,
    transfer_mode: str = "pull",
) -> str:
    """计算 NIXL KV 传输的兼容性哈希。

    只哈希影响两个 NIXL 实例能否成功传输 KV cache 的因子。
    transfer_mode 必须参与哈希：push（WRITE）与 pull（READ）使用
    互不兼容的传输协议，两者绝不能完成握手。
    """
    from vllm import __version__ as vllm_version
    from vllm.config.utils import hash_factors

    model_config = vllm_config.model_config
    cache_config = vllm_config.cache_config
    is_hma_enabled = not vllm_config.scheduler_config.disable_hybrid_kv_cache_manager

    factors = {
        # 版本兼容性：vLLM 版本 + NIXL connector 版本（本次升到 7）
        "vllm_version": vllm_version,
        "nixl_connector_version": NIXL_CONNECTOR_VERSION,
        # 模型架构 - 影响 KV cache 形状
        "model": model_config.model,
        "dtype": str(model_config.dtype),
        "num_kv_heads": model_config.get_total_num_kv_heads(),
        "head_size": model_config.get_head_size(),
        "num_hidden_layers": model_config.get_total_num_hidden_layers(),
        # 注意力后端与 KV cache dtype 影响内存布局
        "attn_backend_name": attn_backend_name,
        "cache_dtype": str(cache_config.cache_dtype),
        "cross_layers_blocks": cross_layers_blocks,
        "is_hma_enabled": is_hma_enabled,
        "speculative_config": _get_speculative_compatibility_factors(vllm_config),
        # push（WRITE）与 pull（READ）连接器协议不兼容，必须参与哈希
        "transfer_mode": transfer_mode,
    }

    compat_hash = hash_factors(factors)
    logger.debug(
        "NIXL compatibility hash: %s (model=%s, dtype=%s, num_kv_heads=%d, "
        "cache_dtype=%s, attn_backend=%s)",
        compat_hash,
        factors["model"],
        factors["dtype"],
```

```
        factors["num_kv_heads"],
        factors["cache_dtype"],
        attn_backend_name,
    )
    return compat_hash
```

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

push 侧 router 支持：覆盖 _TRANSFER_MODE="push"，并在 request_finished 返回的 kv_transfer_params 中新增 transfer_mode 键。

```
class NixlPushConnectorScheduler(NixlBaseConnectorScheduler):
    """Push 模式调度器（基于 WRITE 的 KV 传输）。"""

    # 覆盖基类的 "pull"：兼容性哈希与 kv_transfer_params 都据此
    # 区分 push 与 pull，外部 router 才能正确路由 WRITE producer。
    _TRANSFER_MODE: str = "push"

    def request_finished(self, request, block_ids):
        # ... 前面逻辑省略：决定 delay_free_blocks、登记 finished blocks ...
        return delay_free_blocks, dict(
            do_remote_prefill=True,
            do_remote_decode=False,
            remote_block_ids=block_ids,
            remote_engine_id=self.engine_id,
            remote_request_id=request.request_id,
            remote_host=self.side_channel_host,
            remote_port=self.side_channel_port,
            tp_size=self.vllm_config.parallel_config.tensor_parallel_size,
            pp_size=self.vllm_config.parallel_config.pipeline_parallel_size,
            remote_num_tokens=remote_num_tokens,
            # 显式公布传输模式，供 vllm-router（router#187）区分 push 与 pull
            transfer_mode=self._TRANSFER_MODE,
        )
```

tests/v1/kv_connector/unit/test_nixl_connector.py

测试覆盖核心行为：哈希区分 push/pull、scheduler 宣传模式、pull 端到端 kv_transfer_params 断言。

```
@pytest.mark.skip_global_cleanup
def test_transfer_mode_changes_compatibility_hash():
    # push（WRITE）与 pull（READ）连接器的传输协议不兼容，
    # 因此它们的兼容性哈希必须不同；相同模式必须相同。默认模式为 pull。
    config = create_vllm_config()

    pull_hash = compute_nixl_compatibility_hash(
        config, "FLASH_ATTN", False, transfer_mode="pull"
    )
    push_hash = compute_nixl_compatibility_hash(
        config, "FLASH_ATTN", False, transfer_mode="push"
```

```
    )

    assert pull_hash != push_hash
    assert pull_hash == compute_nixl_compatibility_hash(
        config, "FLASH_ATTN", False, transfer_mode="pull"
    )
    assert compute_nixl_compatibility_hash(config, "FLASH_ATTN", False) == pull_hash


@pytest.mark.skip_global_cleanup
def test_scheduler_advertises_transfer_mode():
    # 各 scheduler 在 kv_transfer_params 中宣传自己的传输模式,
    # 外部 router 据此区分 pull（READ）与 push（WRITE）两种 producer。
    from vllm.distributed.kv_transfer.kv_connector.v1.nixl.pull_scheduler import (
        NixlPullConnectorScheduler,
    )
    from vllm.distributed.kv_transfer.kv_connector.v1.nixl.push_scheduler import (
        NixlPushConnectorScheduler,
    )

    assert NixlPullConnectorScheduler._TRANSFER_MODE == "pull"
    assert NixlPushConnectorScheduler._TRANSFER_MODE == "push"
```

评论区精华

iyastreb 在 metadata.py 的哈希因子新增行上评论："May we also extend the kv_transfer_params with transfer_mode? We need this field for the router which can distinguish pull from push this way"，并附上 vllm-project/router#187 链接及 push_scheduler.py 的建议 diff（在 request_finished 返回中增加 transfer_mode）。tzulingk回复：已在aa50d198f7完成——scheduler类新增_TRANSFER_MODE（基类pull、push scheduler 覆盖 push），pull 与 push 两个 request_finished 路径都宣传 transfer_mode，并补充 pull 端到端断言与 scheduler 模式检查。该反馈直接促成了 PR 第二 commit（router 支持部分），最终 iyastreb 多次 APPROVED，tlrmchlsmth 与 SageMoore 也 APPROVED。

- 在 kv_transfer_params 中暴露 transfer_mode 供外部 router 使用 (design): 作者在第二 commit（aa50d198f7）实现：scheduler 类新增 _TRANSFER_MODE，pull 与 push 两个 request_finished 路径都在 kv_transfer_params 返回 transfer_mode，并补充 pull 端到端断言与 scheduler 模式检查。

风险与影响

- 风险：
  1. 协议级版本升级：NIXL_CONNECTOR_VERSION 6→7 意味着 prefill/decode 两侧必须同步升级，否则握手失败；这是有意为之的早期失败，但滚动升级时需协调。
  2. 默认参数风险：transfer_mode="pull" 保持向后兼容，但若未来新增 push 变体遗漏覆盖 _TRANSFER_MODE，会静默按 pull 计算哈希并放行错配。

3. router 依赖：kv_transfer_params 新增字段对旧 router 透明，但依赖该字段的 router#187 必须在 vLLM 升级后同步部署。
4. 测试盲区：GPU 依赖的真实握手测试未在本地运行（macOS 无 CUDA），依赖 CI 覆盖。 - 影响：用户面：使用 NIXL KV connector 的用户将获得更早、更清晰的错配报错（握手期而非传输期）；使用 vllm-router 的用户可通过 transfer_mode 区分 push/pull producer，为 push 并行路由（issue#187）铺路。系统面：握手阶段拦截协议不兼容配对，降低故障排查成本；版本号增长要求跨实例部署协调。团队面：明确了 NIXL 兼容性哈希因子清单的维护方式，为后续协议演进提供文档模板，并形成了 vLLM 与 vllm-router 的跨仓库协作点。 - 风险标记：协议版本升级 6→7 需 prefill/decode 同步，GPU 依赖握手测试依赖 CI 未本地验证，外部 router 依赖新增字段需同步部署

关联脉络

- PR #49230 （标题未提供）：PR body 明确说明本 PR 的 transfer-mode 工作是从 #49230 拆出，同属 NIXL 兼容性哈希功能线，并回应了 #49230 review 中 iyastreb 的反馈（discussion_r3686841923）。
- PR #187 Support for NIXL push mode: vllm-project/router 仓库的关联 Issue：本 PR 在 kv_transfer_params 中新增 transfer_mode 字段正是为 router#187 的 push 并行路由提供支撑；review 讨论中也直接引用该 PR。