PR #40344 完整报告

vllm-project/vllm

[Bugfix][ROCm] Resolve MoRI connector hangs at high concurrency

合并时间：2026-05-28 22:30

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/40344

执行摘要

- 一句话：修复高并发下 MoRI 连接器挂起
- 推荐动作：建议所有使用 MoRIIO 部署 1P1D 场景的团队尽快合入并验证。推荐精读 update_connector_output 和 _mark_request_done 的实现，理解如何在不侵入调度器的情况下实现块回收。

功能与动机

修复 Issue #40340：在 1P1D 部署 DeepSeek-R1 时，当 vllm bench serve 并发 ≥128 后 MoRI 连接器无限挂起，导致服务不可用。根本问题是多项竞态条件和资源泄漏在高并发下被放大。

实现拆解

1. 禁用 in-band 通知：在 RdmaBackendConfig 中设置 enable_notification=False，因 ZMQ 已承担完成通知，高并发下 QP 发送队列耗尽会污染传输状态。
2. 处理失败传输：在 _pop_done_transfers 中增加对 Failed() 状态的判读，确保失败请求的 KV 块能被释放。
3. 保持映射存活：在生产者侧将 transfer_id<->request_id 映射保留到收到 finished_sending 通知，防止调度器提前释放未完成发送的请求。
4. KV 块超时回收：通过 update_connector_output 钩子（在 moriio_connector.py 新增）检查 _deferred_send_deadlines，超时后推断完成发送，回收泄漏的 KV 块。
5. 修复WRITE模式竞态：transfer_id_to_request_id的添加与弹出不再错位，避免通知丢失。
6. 修复 READ 模式 done_recving：READ 模式下的请求不进入 WAITING_FOR_REMOTE_KVS，不应在 done_recving 中报告。
7. 轮询改进：status.Wait() 无限忙等替换为带截止时间的轮询，并在所有忙等循环中添加 1ms 睡眠。
8. 配置参数化：新增 transfer_timeout 和 defer_timeout 两个可配置超时，通过 kv_connector_extra_config 传递（moriio_common.py）。

测试配套：无新增单元测试，但作者在 MI300X 上使用 vllm bench serve 验证了 256 与 512 并发 READ/WRITE 模式，并通过 GSM8k 评估。

关键文件：

- vllm/distributed/kv_transfer/kv_connector/v1/moriio/moriio_connector.py（模块 KV 连接器；类别 source；类型 core-logic；符号 update_connector_output, request_finished, get_finished）：核心连接器类，新增 update_connector_output 方法实现 KV 块超时回收，修复多个竞态条件和 ID 映射错误。
- vllm/distributed/kv_transfer/kv_connector/v1/moriio/moriio_engine.py（模块 传输引擎；类别 source；类型 core-logic；符号 _mark_request_done, _write_worker_loop, _process_deferred_tasks, MoRIIOWriter）：实现写工作线程和传输状态管理，新增 _mark_request_done 方法确保失败 / 超时写任务释放 KV 块，修正 done_req_ids 中的 ID 类型。
- vllm/distributed/kv_transfer/kv_connector/v1/moriio/moriio_common.py（模块 公共配置；类别 source；类型 configuration；符号 MoRIIOConfig, MoRIIOConstants）：基础配置和常量定义，新增 transfer_timeout 和 defer_timeout 配置项及默认值，为超时回收提供参数化支持。

关键符号：update_connector_output, _mark_request_done, _write_worker_loop, _process_deferred_tasks, request_finished, get_finished, MoRIIOConfig.from_vllm_config

关键源码片段

vllm/distributed/kv_transfer/kv_connector/v1/moriio/moriio_connector.py

核心连接器类，新增 update_connector_output 方法实现 KV 块超时回收，修复多个竞态条件和 ID 映射错误。

```
# vllm/distributed/kv_transfer/kv_connector/v1/moriio/moriio_connector.py

from vllm.v1.outputs import KVConnectorOutput

class MoRIIOConnector(KVConnectorBase_V1):

    def update_connector_output(self, connector_output: KVConnectorOutput) -> None:
        """每个调度步调用，用于清理超时未释放的 KV 块。"""
        assert self.connector_scheduler is not None
        self.connector_scheduler.update_connector_output(connector_output)

    def request_finished(self, request, block_ids):
        # producer 保持映射直到收到 finished_sending 通知
        if not self.is_producer:
            self.unmap_request_id(request.request_id)
        # ... 其他逻辑
        # 设置超时 deadline，超过后强制释放 KV 块
        self._deferred_send_deadlines[request.request_id] = (
            time.monotonic() + self._defer_timeout
        )
```

vllm/distributed/kv_transfer/kv_connector/v1/moriio/moriio_engine.py

实现写工作线程和传输状态管理，新增 _mark_request_done 方法确保失败 / 超时写任务释放 KV 块，修正 done_req_ids 中的 ID 类型。

```
# vllm/distributed/kv_transfer/kv_connector/v1/moriio/moriio_engine.py

def _write_worker_loop(self) -> None:
    while True:
        self._process_deferred_tasks()
        try:
            task = self._write_task_q.get(timeout=0.01)
        except Empty:
            continue
        if not self._is_remote_ready(task):
            self._deferred_tasks.append(task)
            continue
        try:
            self._execute_write_task(task)
        except Exception:
            logger.exception(
                "Write task failed for request %s, marking done",
                task.request_id,
            )
            self._mark_request_done(task.transfer_id)

def _process_deferred_tasks(self) -> None:
    if not self._deferred_tasks:
        return
    defer_timeout = self._defer_timeout
    now = time.perf_counter()
    still_deferred = []
    for task in self._deferred_tasks:
        if now - task.enqueue_time > defer_timeout:
            logger.error(
                "Deferred write task for request %s expired after %.1fs, marking done",
                task.request_id, now - task.enqueue_time,
            )
            self._mark_request_done(task.transfer_id)
            continue
        if self._is_remote_ready(task):
            try:
                self._execute_write_task(task)
            except Exception:
                self._mark_request_done(task.transfer_id)
        else:
            still_deferred.append(task)
    self._deferred_tasks = still_deferred

def _mark_request_done(self, transfer_id: str) -> None:
    """标记传输完成，释放对应 KV 块（无论成功或失败）。"""
```

```
        wrapper = self.worker.moriio_wrapper
        with wrapper.lock:
            # done_req_ids 实际存的是 transfer_id（命名历史遗留）
            wrapper.done_req_ids.append(transfer_id)
            wrapper.done_remote_allocate_req_dict.pop(transfer_id, None)
```

vllm/distributed/kv_transfer/kv_connector/v1/moriio/moriio_common.py

基础配置和常量定义，新增 transfer_timeout 和 defer_timeout 配置项及默认值，为超时回收提供参数化支持。

```
# vllm/distributed/kv_transfer/kv_connector/v1/moriio/moriio_common.py

@dataclass
class MoRIIOConfig:
    ...
    transfer_timeout: float # 传输超时，默认 30s
    defer_timeout: float # 延迟发送超时，默认 60s

    @classmethod
    def from_vllm_config(cls, vllm_config: VllmConfig) -> "MoRIIOConfig":
        ...
        transfer_timeout = float(
            extra_config.get("transfer_timeout", MoRIIOConstants.DEFAULT_TRANSFER_TIMEOUT)
        )
        defer_timeout = float(
            extra_config.get("defer_timeout", MoRIIOConstants.DEFAULT_DEFER_TIMEOUT)
        )
        return cls(
            ...
            transfer_timeout=transfer_timeout,
            defer_timeout=defer_timeout,
        )

class MoRIIOConstants:
    DEFAULT_TRANSFER_TIMEOUT = 30.0
    DEFAULT_DEFER_TIMEOUT = 60.0
```

评论区精华

关键讨论包括：

1. 调度器变更必要性：njhill 指出调度器变更可由连接器侧处理，最终全部回退，只保留 update_connector_output 钩子。
2. req_id 与 transfer_id 混淆：多处代码误用 request_id 而应使用 transfer_id，例如 _pop_done_transfers 返回的是 transfer_id 但类型名称为 req_id；done_req_ids 列表中实际存储的是 transfer_id。作者逐一修正并重命名变量。
3. envs 缓存开销：tjtanaa 提醒 envs.VLLM_MORIIO_* 访问成本高，建议缓存为配置值。作者后续将超时参数移入 kv_connector_extra_config 并计划合并 #43303 彻底迁移。

4. 异常捕获范围：gemini-code-assist[bot] 建议不要捕获 KeyboardInterrupt/SystemExit；作者指出它们不是 Exception 的子类，当前代码安全。

- 调度器变更必要性 (design): 仅保留 connector 侧 update_connector_output 钩子。
- req_id 与 transfer_id 混淆 (correctness): 已修复，统一使用 transfer_id 用于内部追踪。
- envs 缓存开销 (performance): 超时参数已移入 kv_connector_extra_config，后续通过 PR #43303 彻底迁移。
- 异常捕获范围 (style): 无变更，当前代码安全。
- 忙等循环移除风险 (correctness): 最终代码改为基于截止时间的轮询，未保留忙等。

风险与影响

- 风险：风险包括：
  - 核心路径变更：连接器是 1P1D 分离部署的瓶颈，改动直接影响所有使用 MoRIIO 的推理请求。
  - 超时参数敏感性：transfer_timeout 和 defer_timeout 默认值（30s/60s）可能在网络延迟较高时导致过早释放 KV 块。
  - 无自动化测试：虽经手动测试，但缺乏 CI 覆盖，未来回归风险需依赖硬件测试。
  - 忙等循环移除：之前忙等保证了某些竞态下的完整性，改为轮询后可能引入微妙的时序依赖。
  - 影响：对 MoRIIO 用户（ROCm MI300X 等）至关重要，修复高并发场景下服务完全挂起。对其他用户无影响。代码复杂度增加（新增超时回收逻辑、配置参数），但整体变动集中在连接器模块，隔离性好。
  - 风险标记：无自动测试覆盖，超时参数敏感，高并发竞态风险

关联脉络

- 暂无明显关联 PR