PR #46290 完整报告

vllm-project/vllm

[ROCm][P/D] Fix MoRIIO WRITE mode for mixed KV layouts

合并时间：2026-06-23 12:12

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/46290

执行摘要

- 一句话：修复 MoRIIO WRITE 混合 KV 布局正确性问题
- 推荐动作：值得精读：此 PR 展示了如何在不改动上层调度逻辑的前提下，在分布式 KV 传输中处理异构缓存几何。moriio_layout.py 中的内核块布局检测算法和 moriio_engine.py 中的写入完成跟踪机制值得借鉴。使用 MoRIIO WRITE 模式的团队应关注此 PR 的变更。

功能与动机

如 PR 描述所述，前一个 PR #46039 使 MoRIIO READ 模式支持了混合 KV 缓存布局，但 WRITE 模式仍存在两个正确性问题：

1) 跨层重用了一个 request-wide 的 offset tuple，而不同层（如密集 K/V 层 vs 仅索引层）可能有不同的缓存几何； 2) 计数注册的缓存张数而非实际由 WRITE hook 调度的层来完成 WRITE。本 PR 针对这些问题进行修复，确保混合缓存布局下 WRITE 的正确性。

实现拆解

1. 按几何计算偏移量（moriio_layout.py）：新增 _spec_dim_matches、_kernel_layout_matches、_select_kernel_block_layout 函数，自动检测内核块 KV 缓存几何（支持分离 / 交错布局），在 get_layer_transfer_geometry 中区分块大小所在轴，计算正确的 block_stride 和 kernel_blocks_per_block。
2. 缓存 WRITE 偏移计划（moriio_engine.py）：引入 _get_write_geometry_key 函数，基于张量形状、步长和 dtype 生成键；MoRIIOWriter 新增 _scheduled_writes、_scheduled_layers、_sealed_writes 字典，按 transfer_id 跟踪每个传输的调度写入层数和密封计数。schedule_write 返回 bool 标记是否真正调度，seal_pending_transfers 在模型前向后密封期望的写入数。
3. 修复完成逻辑（moriio_engine.py）：新增 _finalize_if_complete 检查调度写入是否全部完成，只有完成后才触发通知和释放。新增 _clear_transfer_state 清理传输状态，_is_transfer_terminal 检查传输是否已完成（避免重复调度）。
4. 连接器侧配套（moriio_connector.py）：实现 wait_for_save 非空（仅在 WRITE 模式下等待）；新增 _send_transfer_release 发送结构化 release 消息，_release_write_prefill_blocks 在写入完成后释放预填充块。update_state_after_alloc 中从 kv_transfer_params 获取 remote 地址而非仅解析 request_id。

5. 数据结构更新（moriio_common.py）：RemoteAllocInfo 增加 writes_expected、completion_* 字段和 transfer_offsets 字典（按几何键缓存偏移量），支持完成通知去重。
6. 单元测试（test_moriio_kv_layout.py、test_moriio_connector.py）：新增 577 行和 77 行测试代码，覆盖偏移量计算、写入完成通知、结构化 / 普通消息处理、生产者块释放、内核块布局等场景。

关键文件：

- vllm/distributed/kv_transfer/kv_connector/v1/moriio/moriio_engine.py（模块 写入引擎；类别 source；类型 core-logic；符号 _get_write_geometry_key, schedule_write, is_scheduled, seal_pending_transfers）：核心写入引擎，新增按几何计算偏移量和写入完成跟踪。
- vllm/distributed/kv_transfer/kv_connector/v1/moriio/moriio_connector.py（模块 连接器；类别 source；类型 core-logic；符号 _send_transfer_release, _release_write_prefill_blocks, wait_for_save）：连接器层新增结构化 release 和预填充块释放逻辑。
- vllm/distributed/kv_transfer/kv_connector/v1/moriio/moriio_layout.py（模块 传输布局；类别 source；类型 core-logic；符号 _spec_dim_matches, _kernel_layout_matches, _select_kernel_block_layout）：传输布局模块，新增内核块几何自动检测。
- tests/v1/kv_connector/unit/test_moriio_kv_layout.py（模块 布局测试；类别 test；类型 test-coverage；符号 _full_spec, test_separated_kv_layout_uses_kv_axis_zero_and_block_axis_one, _writer_with_fake_worker, _wrapper_for_messages）：单元测试覆盖新布局几何、写入完成通知、消息处理等。
- tests/v1/kv_connector/unit/test_moriio_connector.py（模块 连接测试；类别 test；类型 test-coverage；符号 _find_free_port, _write_consumer_scheduler_for_finished_request, send_notify, test_write_mode_finished_before_alloc_releases_prefill_blocks）：单元测试覆盖释放行为和完成路径。

关键符号：_get_write_geometry_key, schedule_write, is_scheduled, seal_pending_transfers, _finalize_if_complete, _send_transfer_release, _release_write_prefill_blocks, wait_for_save, _spec_dim_matches, _kernel_layout_matches, _select_kernel_block_layout, get_layer_transfer_geometry

关键源码片段

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

核心写入引擎，新增按几何计算偏移量和写入完成跟踪。

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

# WRITE mode 状态机关键数据结构

# 基于张量 shape、stride、dtype 生成几何键，用于缓存偏移量
WriteGeometryKey = tuple[tuple[int, ...], tuple[int, ...], torch.dtype]

def _get_write_geometry_key(kv_cache: torch.Tensor) -> WriteGeometryKey:
```

```
    return (tuple(kv_cache.shape), tuple(kv_cache.stride()), kv_cache.dtype)

class MoRIIOWriter:
    def __init__(self, worker: "MoRIIOConnectorWorker"):
        # ... 其他属性
        self._write_state_lock = threading.Lock()
        # 每个 transfer_id 已调度的写入层数
        self._scheduled_writes: dict[TransferId, int] = defaultdict(int)
        # 每个 transfer_id 已调度的层名称集合（用于去重）
        self._scheduled_layers: dict[TransferId, set[str]] = defaultdict(set)
        # 每个 transfer_id 密封后的期望写入总数（在前向之后设置）
        self._sealed_writes: dict[TransferId, int] = {}

    def schedule_write(self, task: WriteTask) -> bool:
        """调度一个写入任务，返回是否真正调度（避免重复调度和已完成传输）"""
        self.ensure_worker_started()
        if self._is_transfer_terminal(task.transfer_id):
            return False
        with self._write_state_lock:
            if self._is_transfer_terminal(task.transfer_id):
                return False
            # 跳过同一层已被调度的情况
            if task.layer_name in self._scheduled_layers[task.transfer_id]:
                return False
            self._scheduled_layers[task.transfer_id].add(task.layer_name)
            self._scheduled_writes[task.transfer_id] += 1
        self._write_task_q.put(task)
        return True

    def seal_pending_transfers(self) -> None:
        """前向完成后的密封：将当前调度计数保存为 sealed 值"""
        with self._write_state_lock:
            for xfer_id, count in self._scheduled_writes.items():
                if xfer_id not in self._sealed_writes:
                    self._sealed_writes[xfer_id] = count
            # 尝试完成所有已密封传输
            for xfer_id in list(self._sealed_writes.keys()):
                self._finalize_if_complete(xfer_id)
```

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

连接器层新增结构化 release 和预填充块释放逻辑。

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

class MoRIIOConnectorScheduler:
    def _send_transfer_release(self, transfer_id: TransferId, host: str, port: int):
        """向远端发送结构化 release 消息，通知传输完成"""
        path = make_zmq_path("tcp", host, port)
        if path not in self.paths:
```

```
            ctx = zmq.Context.instance()
            sock = make_zmq_socket(
                ctx=ctx, path=path, socket_type=zmq.DEALER, bind=False
            )
            self.paths[path] = sock
        self.paths[path].send(
            msgpack.dumps({"type": "release", "transfer_id": transfer_id})
        )

    def _release_write_prefill_blocks(
        self, request_id: ReqId, params: dict[str, Any]
    ):
        """在写入完成后释放预填充块：获取远端地址并发送 release"""
        transfer_id = params.get("transfer_id")
        if transfer_id is None:
            logger.warning(
                "Cannot release WRITE prefill blocks for request %s: "
                "missing transfer_id", request_id
            )
            return
        remote_host = params.get("remote_host")
        remote_notify_port = params.get("remote_notify_port")
        if remote_host is None or remote_notify_port is None:
            # 回退到从 request_id 解析（传统方式）
            try:
                peer_zmq = get_peer_zmq_from_request_id(
                    request_id, is_producer=False
                )
                remote_host, _, remote_notify_port = parse_moriio_zmq_address(peer_zmq)
            except ValueError:
                logger.warning(...)
                return
        remote_notify_port = int(remote_notify_port)
        for tp_index in range(self.tp_size):
            target_port = remote_notify_port + get_port_offset(
                params.get("remote_dp_rank", 0), tp_index
            )
            self._send_transfer_release(transfer_id, remote_host, target_port)
```

评论区精华

Review 中仅有一条来自 depthfirst-app[bot] 的自动安全扫描评论，指出 _send_transfer_release 方法中 host 和 port 来自用户控制的 kv_transfer_params，可能被攻击者利用连接到内部服务，严重性为 MEDIUM，建议增加验证。该评论未在合并前得到明确回复或修改，但 PR 已合并，说明团队可能评估认为该风险在 MoRIIO 使用场景中可控。

- ZMQ 连接安全风险 (security): 未在合并前修复或回复，但该函数接收的参数由调度器内部构造，外部攻击面有限。

风险与影响

- 风险：
  1. ZMQ 连接无验证（moriio_connector.py:399-410）：_send_transfer_release 根据用户提供的 host/port 发起连接，存在 SSRF 风险。但由于 MoRIIO 通常运行在受控集群内，且 kv_transfer_params 由调度器内部构造，外部攻击面较小。
1. 新状态管理竞态（moriio_engine.py）：引入 _write_state_lock 保护 _scheduled_writes、_scheduled_layers、_sealed_writes，但 _write_worker_loop 运行在独立线程中，与主线程通过 Queue 通信，需要确保锁顺序一致以防止死锁。
2. 内核块布局自动检测误判（moriio_layout.py:_select_kernel_block_layout）：当 shape[2] 和 shape[3] 都匹配 spec 且值不同时会引发 ValueError，但若值相等则直接返回，可能掩盖布局歧义。
3. 性能影响：按传输分离状态增加了字典查找和锁开销，但通常 WRITE 操作本身是 I/O 密集的，CPU 锁争用影响有限。 - 影响：影响范围：仅限使用 MoRIIO WRITE 模式进行 P/D 分离推理的用户（主要运行在 AMD ROCm 上）。 影响程度：修复了混合 KV 缓存布局（如 MiniMax-M3 等模型）下的写入正确性，对这类模型是功能修复。内部重构不会影响 API 或外部接口。 回归风险：对仅使用 READ 模式或单一布局的用户无影响；单元测试覆盖了混合布局的主要路径，E2E 验证通过 GSM8K 和基准测试。
- 风险标记：ZMQ 连接无验证，新状态管理竞态风险，内核块布局自动检测可能误判

关联脉络

- PR #46039 Add MoRIIO READ mode layout awareness: 本 PR 是 #46039 的对称纠正，共同实现混合 KV 布局在 MoRIIO READ 和 WRITE 模式下的完整支持。