PR #46039 完整报告

vllm-project/vllm

[ROCm][P/D] Support MiniMax-M3 mixed KV layouts in MoRIIO READ mode

合并时间：2026-06-21 20:55

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/46039

执行摘要

- 一句话：修复 MoRIIO READ 下混合 KV 布局的传输偏移
- 推荐动作：该 PR 代码质量高，将布局逻辑抽象为独立模块，并通过 KVCacheSpec 驱动而非硬编码形状模式。建议阅读 moriio_layout.py 中的几何计算逻辑，了解如何支持多种缓存布局的偏移量推导。社区开发者可参考此模式添加新的布局支持。

功能与动机

MiniMax-M3 注册了多种 KV 缓存布局（分离式、ROCm 交错式、3D 仅 key/MLA 式），MoRIIO 之前重用某个代表性层的布局假设和偏移，导致密集 K/V 层与仅 key/indexer 层混合时读写错误内存区域，从而破坏 P/D 部署的解码输出。issue #45885 报告了 ROCm MiniMax M3 MXFP8 Disagg not working。

实现拆解

1. 新增布局辅助模块：创建 vllm/distributed/kv_transfer/kv_connector/v1/moriio/moriio_layout.py，定义 LayerTransferGeometry 命名元组和关键函数：build_layer_to_spec 从 KVCacheConfig 构建层名到规格的映射；is_mla_cache_layer 通过 KVCacheSpec 判断是否为 MLA/ 索引器层；get_layer_transfer_geometry 根据形状和 MLA 标记计算传输几何参数（块步长、区域划分等）；compute_block_transfer_offsets 为指定的本地和远程块生成字节偏移量；iter_layer_registration_regions 和 merge_contiguous_offsets 辅助注册内存区域和合并偏移。
2. 改造连接器 worker：在 moriio_connector.py 中，向 MoRIIOConnectorWorker 构造函数传入 kv_cache_config，并添加 layer_to_spec 属性，使其能够按层查询规格而非依赖张量形状推断。新增 _is_mla_cache_layer、_get_layer_transfer_geometry、_iter_layer_registration_regions 三个私有方法，封装对 moriio_layout 模块的调用，并在 register_kv_caches 中逐层计算注册区域和传输偏移。
3. 重写注册和传输逻辑：register_kv_caches 中不再假设所有层布局一致，而是为每层独立调用 _iter_layer_registration_regions 获取注册区域列表；在构建 READ 传输任务时，对每个块逐层调用 _get_layer_transfer_geometry 和 compute_block_transfer_offsets 生成偏移量，并传入 remote_num_blocks 以正确计算远程步长。
4. 测试配套：新增 tests/v1/kv_connector/unit/test_moriio_kv_layout.py，覆盖三种布局（分离、交错、MLA 仅 key）的几何计算和混合层的偏移计算；修改 test_moriio_connector.py，更新 FakeMoRIIOConnectorWorker 以支持

kv_cache_config 参数，并改进 _make_test_kv_cache_config 使用真实的 FullAttentionSpec。

关键文件：

- vllm/distributed/kv_transfer/kv_connector/v1/moriio/moriio_layout.py（模块 KV 传输；类别 source；类型 core-logic；符号 LayerTransferGeometry, build_layer_to_spec, is_mla_cache_layer, get_layer_transfer_geometry）：核心新增文件，定义了所有布局感知的辅助函数，包括几何计算、偏移量计算和注册区域枚举。是整个 PR 的逻辑核心。
- vllm/distributed/kv_transfer/kv_connector/v1/moriio/moriio_connector.py（模块 KV 传输；类别 source；类型 dependency-wiring；符号 init, _is_mla_cache_layer, _get_layer_transfer_geometry, _iter_layer_registration_regions）：主连接器文件，被修改以支持布局感知的注册和传输。关键变更包括转发 kv_cache_config 到 worker，添加辅助方法，并重写 register_kv_caches 和 READ 传输逻辑。
- tests/v1/kv_connector/unit/test_moriio_kv_layout.py（模块 单元测试；类别 test；类型 test-coverage；符号 _full_spec, _mla_spec, _worker, _remote_meta）：新增的单元测试，覆盖了三种缓存布局的几何计算和混合层的偏移验证，确保核心逻辑正确。
- tests/v1/kv_connector/unit/test_moriio_connector.py（模块 单元测试；类别 test；类型 test-coverage）：修改了 worker 的 mock 和测试配置，使其与新接口兼容。

关键符号：get_layer_transfer_geometry, compute_block_transfer_offsets, iter_layer_registration_regions, is_mla_cache_layer, build_layer_to_spec, register_kv_caches, init(MoRIIOConnectorWorker)

关键源码片段

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

主连接器文件，被修改以支持布局感知的注册和传输。关键变更包括转发 kv_cache_config 到 worker，添加辅助方法，并重写 register_kv_caches 和 READ 传输逻辑。

```
# 导入新布局模块
from vllm.distributed.kv_transfer.kv_connector.v1.moriio.moriio_layout import (
    LayerTransferGeometry,
    build_layer_to_spec,
    compute_block_transfer_offsets,
    get_layer_transfer_geometry,
    is_mla_cache_layer,
    iter_layer_registration_regions,
)

class MoRIIOConnector(KVConnectorBase_V1):
    def __init__(self, vllm_config, role, kv_cache_config):
        super().__init__(vllm_config, role, kv_cache_config)
        # ... 其他初始化
        if role == KVConnectorRole.WORKER:
            self.connector_worker = MoRIIOConnectorWorker(
                vllm_config, self.engine_id, kv_cache_config # 转发 kv_cache_config
```

```
        )

class MoRIIOConnectorWorker:
    def __init__(self, vllm_config, engine_id, kv_cache_config):
        # ... 其他初始化
        self.layer_to_spec = build_layer_to_spec(kv_cache_config) # 构建层映射
        self.kv_cache_shapes = {} # 记录每层形状
        self.block_lens = {} # 记录每层块长度

    def register_kv_caches(self, kv_caches: dict[str, torch.Tensor]):
        """逐层注册 KV 缓存内存区域"""
        self.kv_caches = kv_caches
        # 转为字典以每层为单位
        kv_caches_base_addr = []
        for layer_name, cache in kv_caches.items():
            regions = self._iter_layer_registration_regions(layer_name)
            for region, length in regions:
                kv_caches_base_addr.append(region.data_ptr())
                # 注册到 MoRIIO 库 ...
        self.kv_cache_base_addr = kv_caches_base_addr

    def _is_mla_cache_layer(self, layer_name: str) -> bool:
        return is_mla_cache_layer(self.layer_to_spec, layer_name)

    def _get_layer_transfer_geometry(self, layer_name: str, remote_num_blocks=None):
        return get_layer_transfer_geometry(
            layer_name, self.kv_caches[layer_name],
            self.layer_to_spec, remote_num_blocks
        )

    def _iter_layer_registration_regions(self, layer_name: str):
        return iter_layer_registration_regions(
            layer_name, self.kv_caches[layer_name], self.layer_to_spec
        )
```

tests/v1/kv_connector/unit/test_moriio_kv_layout.py

新增的单元测试，覆盖了三种缓存布局的几何计算和混合层的偏移验证，确保核心逻辑正确。

```
@staticmethod
def _full_spec(block_size=4):
    return FullAttentionSpec(
        block_size=block_size, num_kv_heads=2, head_size=3, dtype=torch.bfloat16
    )

def test_separated_kv_layout_uses_kv_axis_zero_and_block_axis_one():
    # 分离布局 : [2, num_blocks, block_size, heads, dim]
    cache = torch.empty((2, 8, 4, 2, 3), dtype=torch.bfloat16)
    worker = _worker({"layer": cache}, {"layer": _full_spec()})
    geometry = moriio_layout.get_layer_transfer_geometry(
```

```
        "layer", cache, worker.layer_to_spec, remote_num_blocks=16
    )
    assert geometry.block_stride == 24
    assert geometry.split_kv_regions
    # 验证偏移计算
    offsets = moriio_layout.compute_block_transfer_offsets(
        "layer", cache, worker.layer_to_spec,
        [1, 3], [4, 5], _remote_meta().num_blocks
    )
    assert offsets == ([48, 144, 432, 528], [192, 240, 960, 1008], [48, 48, 48, 48])

def test_interleaved_kv_layout_uses_block_axis_zero_and_kv_axis_one():
    # 交错布局 : [num_blocks, 2, block_size, heads, dim]
    cache = torch.empty((8, 2, 4, 2, 3), dtype=torch.bfloat16)
    geometry = moriio_layout.get_layer_transfer_geometry(
        "layer", cache, worker.layer_to_spec, remote_num_blocks=16
    )
    assert geometry.block_stride == 48
    assert not geometry.split_kv_regions

def test_mla_key_only_layout_transfers_one_slab_per_block():
    # 3D 索引器布局 : [num_blocks, block_size, latent_dim]
    cache = torch.empty((8, 4, 3), dtype=torch.bfloat16)
    worker = _worker({"layer": cache}, {"layer": _mla_spec()})
    geometry = moriio_layout.get_layer_transfer_geometry(
        "layer", cache, worker.layer_to_spec, remote_num_blocks=16
    )
    assert geometry.transfers_per_block == 1
    assert geometry.local_kv_stride is None

def test_mixed_layers_compute_distinct_offsets_per_layer():
    # 混合布局: 验证各层偏移独立计算
    kv_caches = {
        "separated": torch.empty((2, 8, 4, 2, 3), dtype=torch.bfloat16),
        "interleaved": torch.empty((8, 2, 4, 2, 3), dtype=torch.bfloat16),
        "indexer": torch.empty((8, 4, 3), dtype=torch.bfloat16),
    }
    worker = _worker(kv_caches, {
        "separated": _full_spec(),
        "interleaved": _full_spec(),
        "indexer": _mla_spec(),
    })
    offsets_sep = moriio_layout.compute_block_transfer_offsets(
        "separated", kv_caches["separated"], worker.layer_to_spec,
        [1, 3], [4, 5], _remote_meta().num_blocks
    )
    # 验证偏移长度正确
    assert len(offsets_sep[0]) == 4 # 2 blocks x 2 transfers
```

评论区精华

Reviewer inkcherry 建议使用 kv_cache_config 中的 per-layer spec 驱动分类，而非从张量形状推断布局，以增强可维护性。作者 junkang1991 回复说明实现已采用该方式——通过 KVCacheSpec（而非形状）判断 MLA 层，形状仅用于物理偏移计算。此外，团队内部分阶段规划了后续的 WRITE 偏移缓存和异构 TP 映射 PR，确保当前 READ 修复保持专注。

- 使用 KVCacheSpec 驱动分类而非形状推断 (design): 接受建议，实现已经遵循此方向。
- 后续 WRITE 模式修复和异构 TP 映射规划 (other): 当前 PR 仅聚焦 READ 模式，WRITE 和 TP 映射留待后续。
- 测试覆盖和环境限制 (testing): 测试充分，环境限制合理。

风险与影响

- 风险：
  1. 回归风险：仅修改了 MoRIIO READ 模式，WRITE 模式沿用原有逻辑；在混合布局的 WRITE 场景中可能仍存在类似问题，但 PR 已声明 WRITE 偏移缓存留待后续修复。
  2. 依赖风险：布局识别强依赖 KVCacheConfig 中 KVCacheSpec 的正确性；若模型注册了错误或缺失的规格，可能导致异常。
  3. 性能影响：逐层计算偏移引入了少量额外开销，但仅发生在 READ 任务构建阶段，不影响核心传输路径。
  4. 测试覆盖：新增的单元测试验证了主要布局路径，但未覆盖所有可能的张量形状组合（如 5 维非标准轴顺序）。 - 影响：影响范围：仅影响使用 MoRIIO 连接器在 ROCm 平台上进行 Prefill/Decode 分离部署的用户，特别是 MiniMax-M3 模型。对其他后端（如 Mooncake）无影响。对密集 K/V 模型（如 Qwen3）通过回归测试验证无退化。影响程度：核心功能修复——之前 MiniMax-M3 在 P/D 部署中完全无法工作，此 PR 使其恢复正确。GSM8K 测试结果从 0% 提升至 ~95%。
- 风险标记：WRITE 模式未同步修复，依赖 KVCacheSpec 正确性，仅覆盖 MoRIIO 后端，单元测试环境限制（仅 ROCm）

关联脉络

- PR #45885 [Bug]: ROCm MiniMax M3 MXFP8 Disagg not working: 关联的 issue，报告了 MiniMax-M3 在 MoRIIO READ 模式下的 bug，PR 旨在修复此问题。
- PR #46205 [KV Offload] Support packed HMA KV cache layout: 同为 KV 缓存布局相关的变更，但针对 offload 系统，而非 MoRIIO。可视为并行工作。
- PR #46231 [Bugfix] Defer offload reads while transfers are pending: 同一时期对 KV 传输系统的 bugfix，但针对 offload 路径，与 MoRIIO 无关。