

PR #53698 完整报告

vllm-project/vllm

[Bugfix][ROCm][Disagg] Fix MoRIIO shared KV memory region registration

合并时间: 2026-08-27 00:10

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/53698>

执行摘要

- 一句话: 修复 MoRIIO 共享 KV 内存区注册回归, GSM8K 0→96.8%
- 推荐动作: 值得精读。该 PR 展示了共享 KV backing 布局 (#51718) 对下游 RDMA 注册的连锁影响, 以及页对齐、单一内存区 vs 按层内存区、per-layer 偏移缓存等工程决策。建议重点关注 `_build_shared_kv_mr` 的负偏移越界问题是否在后续版本跟进修复, 并留意 `allocate_kv_cache` 页对齐对非 MoRIIO 后端的潜在影响。

功能与动机

PR body 明确说明这是对 #51718 引入回归的修复: This PR fixes a regression from #51718. That PR changed MLA models to store all KV layers in one shared GPU buffer, with each layer as a view into it. MoRIIO was not updated: it still registered each layer view for RDMA, which failed because those views are not page-aligned (RegisterRdmaMemoryRegion errno:22), and KV was copied to the wrong addresses. 结果是 MoRIIO 分离式 prefill/decode 在 nightly 上 GSM8K 仅 0/1319, 功能完全不可用。

实现拆解

1. 分配侧页对齐 (vllm/v1/worker/utils.py) : `allocate_kv_cache` 创建共享 buffer 时把 `raw_size` 向上取整到 4096 字节整数倍再 `torch.zeros`。由于 `torch.zeros` 返回的地址不保证页对齐, 只有 buffer 自身跨页边界后才能构造页对齐的 RDMA 内存区; 该改动对所有走 KV cache tensor 的推理后端生效, 是 RDMA 注册成功的前提。
2. 连接器注册单一 MR (`moriiio_connector.py`) : 新增 `_build_shared_kv_mr`, 用 `untyped_storage()` 判断所有层是否共享同一 backing; 若是, 则将整块 backing 构造连续 uint8 视图, 遍历 `_iter_layer_registration_regions` 求所有层注册区间覆盖的最大字节范围, 向上取整得到 `reg_nbytes`, 用 `torch.as_strided` 构造页对齐起点后注册为单一 MR, 同时把每层相对 MR 起点的字节偏移写入 `kv_layer_mr_offset`。 `register_kv_caches` 按 `shared_backing` 分支: 共享走单一 MR 路径, 非共享保留原有按层注册路径。
3. 远端偏移与 WRITE 寻址 (`moriiio_connector.py` + `moriiio_engine.py`) : 新增 `_remote_layer_mr_offset`, 用远端引擎元数据中的层基地址 `kv_caches_base_addr[idx]` 减远端 MR 基地址得到远端层偏移; `_prepare_transfer_plan` 把传输偏移缓存键从仅 `geometry` 升级为 `(task.layer_name, *_get_write_geometry_key(layer_cache))`, 并传递 `remote_engine_id`, 避免同 `geometry` 的不同层复用同一份偏移 (这正是旧 bug 写错地址的根源), 配合 `_compute_block_transfer_offsets` 内部叠加 MR 偏移, 确保写入正确的

RDMA 地址。

4. 类型与测试配套: moriio_common.py 新增 TransferOffsetsKey 类型别名, 使 RemoteAllocInfo.transfer_offsets 的键类型与 per-layer 键一致并通过 mypy; 测试文件将 test_write_transfer_plan_caches_offsets_per_geometry 改写为 test_write_transfer_plan_caches_offsets_per_layer_geometry, 改用 _write_task() 构造任务, 断言 dense0/dense1/indexer 三层各自独立计算并缓存偏移 (旧断言中 dense0 与 dense1 共享一份偏移)。提交历史显示后续还有 ruff E501 格式修复与 mypy 类型修复两个配套提交。

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 _build_shared_kv_mr, _remote_layer_mr_offset, register_kv_caches): 核心修复所在: 新增 _build_shared_kv_mr 为共享 backing 注册单一页对齐 MR 并记录每层本地偏移, 新增 _remote_layer_mr_offset 计算远端层偏移, register_kv_caches 按共享 / 非共享分支注册。
- 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 _prepare_transfer_plan): WRITE 传输计划缓存键从仅 geometry 改为 (layer_name, geometry), 并传递 remote_engine_id, 保证各层独立计算偏移, 消除旧 bug 中同 geometry 层复用偏移导致写错地址的问题。
- vllm/v1/worker/utils.py (模块 KV 分配; 类别 source; 类型 core-logic; 符号 allocate_kv_cache): allocate_kv_cache 将共享 KV backing 大小向上取整到 4096 字节页对齐, 是 MoRiIO 单一 RDMA MR 注册成功的前提, 且影响所有走 KV cache tensor 的推理路径。
- 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 test_write_transfer_plan_caches_offsets_per_layer_geometry): 更新 WRITE 传输缓存测试以匹配 per-layer 键语义, 验证每层独立计算偏移且三层都被调度, 直接覆盖旧 bug 的回归场景。
- vllm/distributed/kv_transfer/kv_connector/v1/moriio/moriio_common.py (模块 公共类型; 类别 source; 类型 data-contract; 符号 TransferOffsetsKey, RemoteAllocInfo): 定义 TransferOffsetsKey 类型别名, 使 RemoteAllocInfo.transfer_offsets 的键类型与 per-layer 键一致, 是 mypy 通过的必要配套。

关键符号: allocate_kv_cache, register_kv_caches, _build_shared_kv_mr, _remote_layer_mr_offset, _prepare_transfer_plan, _compute_block_transfer_offsets

关键源码片段

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

核心修复所在: 新增 _build_shared_kv_mr 为共享 backing 注册单一页对齐 MR 并记录每层本地偏移, 新增 _remote_layer_mr_offset 计算远端层偏移, register_kv_caches 按共享 / 非共享分支注册。

```
# 为共享 KV backing 构造单一页对齐的 RDMA 内存区 (MR), 返回 MR tensor 与每层相对 MR 起点的字节偏移
```

```

def _build_shared_kv_mr(
    self, kv_caches: dict[str, torch.Tensor]
) -> tuple[torch.Tensor, dict[str, int]]:
    page_size = 4096
    # 取任一层 view 的 uint8 视角，以便按字节操作整个共享 backing
    backing = next(iter(kv_caches.values())).view(torch.uint8)
    nbytes = backing.untyped_storage().nbytes()
    base = torch.as_strided(backing, (nbytes,), (1,), 0)
    ptr = base.data_ptr()
    slack = ptr % page_size
    end = 0
    # 统计所有层注册区间相对 backing 起点的最大结束偏移
    for layer_name in kv_caches:
        _, region_len = next(
            iter(self._iter_layer_registration_regions(layer_name))
        )
        end = max(end, kv_caches[layer_name].data_ptr() - ptr + region_len)
    # 向上取整到页边界，保证 MR 起始地址页对齐（RDMA 注册硬性要求）
    reg_nbytes = ((slack + end + page_size - 1) // page_size) * page_size
    if reg_nbytes > nbytes:
        raise ValueError(
            f"shared KV backing too small for page-aligned MR: "
            f"need {reg_nbytes}, have {nbytes}"
        )
    # 注意：slack > 0 时负 storage offset 会使 view 头部越出分配边界，
    # 这段越界内存会被随后的 register_local_tensor 注册给远端 RDMA
    reg = torch.as_strided(base, (reg_nbytes,), (1,), 0 if slack == 0 else -slack)
    mr_ptr = reg.data_ptr()
    # 返回单一 MR 及每个 layer 相对 MR 起点的字节偏移
    return reg, {
        name: cache.data_ptr() - mr_ptr for name, cache in kv_caches.items()
    }

```

计算远端引擎中某个 layer 在其共享 MR 内的字节偏移

```

def _remote_layer_mr_offset(
    self,
    layer_name: str,
    remote_moriio_meta: MoRIIOAgentMetadata,
    remote_engine_id: EngineId | None,
) -> int:
    # 非共享 backing 或未指定远端引擎时走旧语义（偏移为 0）
    if not self.kv_layer_mr_offset or not remote_engine_id:
        return 0
    idx = self.layer_base_addr_index[layer_name]
    metas = self.layer_name_to_remote_kv_cache_metadata.get(
        remote_engine_id, {}
    ).get(layer_name)
    if not metas or idx >= len(remote_moriio_meta.kv_caches_base_addr):

```

```

    return 0
# 远端 MR 基地址 = 远端元数据解析出的 buffer 地址;
# 层偏移 = 层基地址 - MR 基地址, WRITE 时叠加到远端地址上
mr_base = self.moriiio_wrapper.get_unpack_memory_metadata(metas[0]).data
return remote_moriiio_meta.kv_caches_base_addr[idx] - mr_base

```

vllm/v1/worker/utils.py

allocate_kv_cache 将共享 KV backing 大小向上取整到 4096 字节页对齐, 是 MoRIIO 单一 RDMA MR 注册成功的前提, 且影响所有走 KV cache tensor 的推理路径。

```

def allocate_kv_cache(
    kv_cache_config: KVCacheConfig,
    device: torch.device,
    layout: KVCacheLayout,
    kernel_block_sizes: list[int] | None = None,
) -> dict[str, torch.Tensor]:
    """分配 KV cache 并将每层视作 [B, H, N, C]。

    所有 KVCacheTensor 共享同一块 backing, 层 l 的块 b 起始于
    offset + l * layer_stride + b * block_stride, 因此 RDMA 传输方
    (如 MoRIIO) 可以把整块 backing 注册为单一内存区 (MR)。
    """
    if not kv_cache_config.kv_cache_tensors:
        return {}

    sizes = {tensor.size for tensor in kv_cache_config.kv_cache_tensors}
    assert len(sizes) == 1, "KV cache tensors must share one backing allocation."
    raw_size = sizes.pop()
    page_size = 4096
    # 向上取整到 4KB 页边界: RDMA 注册要求内存区起始地址页对齐,
    # 若直接按 raw_size 分配, torch 分配器返回的地址不保证页对齐。
    buf = torch.zeros(
        ((raw_size + page_size - 1) // page_size) * page_size,
        dtype=torch.int8,
        device=device,
    )

    kv_caches: dict[str, torch.Tensor] = {}
    for tensor in kv_cache_config.kv_cache_tensors:
        layer_name = tensor.layers[0]
        group_id, group = next(
            (group_id, group)
            for group_id, group in enumerate(kv_cache_config.kv_cache_groups)
            if layer_name in group.layer_names
        )
        spec = group.kv_cache_spec
        if isinstance(spec, UniformTypeKVCacheSpecs):
            spec = spec.kv_cache_specs[layer_name]

```

```

num_blocks = kv_cache_config.num_blocks
kernel_block_size = None
if kernel_block_sizes is not None and group_id < len(kernel_block_sizes):
    kernel_block_size = kernel_block_sizes[group_id]

views = create_kv_cache_views(
    buf,
    spec,
    num_blocks,
    layout,
    tensor,
    kernel_block_size=kernel_block_size,
)
kv_caches.update(zip(tensor.layers, views))
return kv_caches

```

评论区精华

唯一的实质性 review 评论来自 depthfirst-app[bot]，标注为 LOW 严重性：

`_build_shared_kv_mr` 在 GPU 分配地址非 4K 对齐 (`slack > 0`) 时，`torch.as_strided` 使用负 storage offset 创建 tensor view，该 view 延伸到实际分配起点之前最多 4095 字节，这段越界区域随后通过 `register_local_tensor` 注册为 RDMA 内存区，使未拥有的 GPU 内存（可能包含其他请求的 KV 数据或模型数据）可能被远端 RDMA 对等方访问。该评论未获得作者回复，PR 仍由维护者 shen-shanshan 批准合并；此外 pre-commit 两次失败（ruff E501、mypy 类型）均由作者在后续提交中修复，三次 /ci run 均通过。

- `_build_shared_kv_mr` 负偏移导致越界内存暴露 (security): 标注为 LOW 严重性并给出修复建议，但未获作者回复，PR 仍由维护者批准合并，问题遗留至后续版本。
- 传输计划缓存从 geometry 级改为 per-layer 级 (design): 采用 per-layer 键消除错误复用，测试覆盖三层各自独立偏移，设计与实现一致。
- pre-commit 与 CI 检查 (style): 全部检查通过后由 shen-shanshan 批准合并。

风险与影响

- 风险：
 1. 越界内存暴露（未解决）：`_build_shared_kv_mr` 在 `slack > 0` 时通过负 storage offset 构造注册区域，头部最多 4095 字节落在实际分配之外，远端 RDMA 节点可能读到相邻 GPU 内存中的其他请求 KV 或模型权重，构成潜在数据泄露向量，当前版本未修复。
 2. 全局分配行为变化：`allocate_kv_cache` 的页对齐对所有后端生效，每个 KV buffer 最多多占 4095 字节，显存开销可忽略；但若未来后端假设 backing 大小与 KV 容量精确相等，可能触发断言或 shape 不匹配。
 3. 双路径分支回归面：`register_kv_caches` 现在区分共享 / 非共享 backing 两条注册路径，非 MLA 模型走旧逻辑，需要 CI 覆盖验证；per-layer 缓存键使 `transfer_offsets` 条目数增多，内存占用略有上升。- 影响：直接影响 MoRIIO（AMD ROCm 上的 RDMA

KV 传输) 分离式推理的可用性: DeepSeek-V3、Kimi 2.5 等 MLA 模型从完全不可用 (GSM8K 0/1319) 恢复到正常水平 (0.968)。对非 MoRIIO 用户影响很小, 主要是 KVcache 分配向上取整到页边界带来的少量显存开销及潜在越界窗口; 对团队而言, 该修复恢复了 ROCm disaggregated inference 的关键链路, 并暴露了 KV-cache 布局标准化对下游 RDMA 注册组件的联动要求。- 风险标记: 越界内存暴露未修复, 核心路径 RDMA 注册变更, 全局 KV 分配页对齐影响面广

关联脉络

- PR #51718 [6/N][KV-Cache Layout Refactor] Standardize KV cache layout: 本 PR 修复的回归正由它引入: MLA 模型共享 KV backing 后 MoRIIO 按层 view 注册 RDMA 失败, 二者构成 KV-cache 布局标准化系列的前后篇。
- PR #53663 [Bugfix][Mooncake] Fix Mamba prefill truncation ordering: 同为 kv-connector 模块的回归修复, 反映 KV 布局 / 连接器改动需要各传输插件联动回归。
- PR #42644 [Bugfix] Thread kv_transfer_params into engine for /inference/v1/generate (disagg): 同属 disagg KV 传输链路正确性修复, 保护 MoRIIO 等连接器的分离式推理语义。