PR #47685 完整报告

vllm-project/vllm

[ROCm] Align mixed encoder-decoder KV cache views in V2 runner

合并时间：2026-07-07 12:09

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/47685

执行摘要

- 一句话：修复 ROCm V2 runner 混合 KV 缓存视图对齐问题
- 推荐动作：该 PR 值得精读，特别是对 ROCm 平台或涉及混合 attention 后端的开发者。设计决策中，通过 stride 调整而非数据拷贝来对齐视图是一种高效方案。建议关注后续是否添加单元测试以及是否引入平台门控。

功能与动机

PR #47035 标准化了旧模型运行器路径，但 V2 路径中 _reshape_kv_cache 仍存在混合布局语义：decoder self-attention 使用 ROCm 的 K/V-first 布局，而 cross-attention 使用 blocks-first 后端视图，导致块 ID 索引到不同的物理字节，引发 Nemotron Parse 生成测试失败。该 PR 旨在消除这种不一致，使共享分配上的所有视图物理地址一致。

实现拆解

实现按以下步骤进行：

1. 在 _reshape_kv_cache 末尾新增 elif has_attn and kv_cache_config is not None 分支，调用 _align_mixed_attention_kv_cache_views。
2. 新函数 _align_mixed_attention_kv_cache_views 遍历 attention 组，通过 get_kv_cache_block_dim 获取每个后端的块维度（0 表示 blocks-first，1 表示 K/V-first）。
3. 遍历 kv_cache_config.kv_cache_tensors 中的共享分配，检查是否被不同块维度的层共享。若同时存在 0 和 1，说明视图冲突。
4. 对块维度为 0 的层调用 _restride_blocks_first_kv_cache_to_kv_first_storage 重新设置 stride，使 blocks-first 视图在物理 K/V-first 存储上正确映射。
5. 辅助函数 _restride 就地修改 tensor 的 stride 属性，避免数据拷贝。

该改动仅涉及一个文件，未添加测试，依赖现有集成测试覆盖。

关键文件：

- vllm/v1/worker/gpu/attn_utils.py（模块 KV 缓存；类别 source；类型 core-logic；符号 _align_mixed_attention_kv_cache_views, _restride_blocks_first_kv_cache_to_kv_first_storage）：唯一变更文件，包含核心修复逻辑。新增两个函数：_align_mixed_attention_kv_cache_views 检测共享分配中的视图冲突，_restride_blocks_first_kv_cache_to_kv_first_storage 调整 blocks-first 视图的步长。

关键符号：_align_mixed_attention_kv_cache_views，
_restride_blocks_first_kv_cache_to_kv_first_storage

关键源码片段

vllm/v1/worker/gpu/attn_utils.py

唯一变更文件，包含核心修复逻辑。新增两个函数：_align_mixed_attention_kv_cache_views 检测共享分配中的视图冲突，_restride_blocks_first_kv_cache_to_kv_first_storage 调整 blocks-first 视图的步长。

```
# 在 _reshape_kv_cache 末尾，处理完 Mamba 布局后添加：
elif has_attn and kv_cache_config is not None:
    _align_mixed_attention_kv_cache_views(
        attn_groups=attn_groups,
        kv_caches=kv_caches,
        kernel_block_sizes=kernel_block_sizes,
        cache_dtype=cache_dtype,
        kv_cache_config=kv_cache_config,
    )


def _align_mixed_attention_kv_cache_views(
    attn_groups: Iterable[AttentionGroup],
    kv_caches: dict[str, Any],
    kernel_block_sizes: list[int],
    cache_dtype: str,
    kv_cache_config: KVCacheConfig,
) -> None:
    """对齐共享 attention KV 视图，当不同后端布局不一致时。
    Encoder-decoder 模型可以在 decoder self-attention（使用 ROCM_ATTN，
    块维度为 1，K/V-first 布局）和 cross-attention（使用 blocks-first 后端）
    之间共享一个原始 KV 缓存分配。此函数保持物理存储为 ROCM_ATTN 期望的
    K/V-first 布局，并重新设置 blocks-first 逻辑视图的步长，使得块 ID 索引
    到相同的物理字节。
    """
    block_dims_by_layer: dict[str, int] = {}
    for group in attn_groups:
        kv_cache_spec = group.kv_cache_spec
        if not isinstance(kv_cache_spec, AttentionSpec):
            continue
        if group.kv_cache_group_id >= len(kernel_block_sizes):
            continue
        block_dim = group.backend.get_kv_cache_block_dim(
            kernel_block_sizes[group.kv_cache_group_id],
            kv_cache_spec.num_kv_heads,
            kv_cache_spec.head_size,
            cache_dtype_str=cache_dtype,
        )
```

```
        for layer_name in group.layer_names:
            if layer_name in kv_caches:
                block_dims_by_layer[layer_name] = block_dim

    for kv_tensor in kv_cache_config.kv_cache_tensors:
        if kv_tensor.block_stride > 0:
            continue
        shared_block_dims = {
            block_dims_by_layer[layer_name]
            for layer_name in kv_tensor.shared_by
            if layer_name in block_dims_by_layer
        }
        if 0 not in shared_block_dims or 1 not in shared_block_dims:
            continue

        for layer_name in kv_tensor.shared_by:
            if block_dims_by_layer.get(layer_name) == 0:
                _restride_blocks_first_kv_cache_to_kv_first_storage(
                    kv_caches[layer_name]
                )


def _restride_blocks_first_kv_cache_to_kv_first_storage(
    kv_cache: torch.Tensor,
) -> None:
    """通过 as_strided_ 调整 stride，使 blocks-first 视图在
    K/V-first 物理存储上正确映射。
    """
    assert kv_cache.ndim >= 3
    assert kv_cache.shape[1] == 2
    page_size = kv_cache.shape[2:].numel()
    num_blocks = kv_cache.shape[0]
    expected_tail_stride = torch.empty(kv_cache.shape[2:]).stride()
    # 后续根据 expected_tail_stride 计算新 stride 并调用 as_strided_
    ...
```

评论区精华

Review 中主要讨论了两点：

1. 是否 ROCm 特有：njhill 提问是否是 ROCm 特定的要求，避免不必要的工作。AndreasKaratzas 提议可以 *gate under rocm*，但 njhill 表示如果确实是 ROCm 特有最好避免门控，需要更好理解。最终合并时未加平台门控，因为该逻辑对 CUDA 可能也是安全的（只是多一次属性检查）。
2. 复现验证：yewentao256 要求补充复现命令和日志，AndreasKaratzas 提供了 Buildkite job 链接验证修复。

- 补充复现信息 (other): 修复通过验证，未进一步质疑。
- 是否 ROCm 特有？考虑门控 (design): 暂时不添加平台门控，后续再评估。

风险与影响

- 风险：
  - 性能风险：新函数每次 KV 缓存分配时运行，但仅在混合布局存在时进行实际 stride 调整，开销极小。
  - 兼容性风险：修改了共享 KV 缓存的视图，可能影响其他依赖 blocks-first 布局的组件（如 KV Connector、推测解码）。但当前仅在检测到混合布局时操作，且物理存储保持 K/V-first 不变，风险可控。
  - 测试覆盖：无直接单元测试，仅依赖 Nemotron Parse 集成测试，可能遗漏其他混合布局场景。
- 影响：
  - 用户影响：修复了 ROCm 上 Nemotron Parse 模型生成错误的 bug，这些用户可直接受益。
  - 系统影响：新增两个内部函数，仅在 encoder-decoder 且存在共享分配时激活，不影响纯 decoder 或非共享模型。
  - 团队影响：提供了处理多 attention 后端布局差异的通用机制，便于后续支持更多后端。
  - 风险标记：非 CUDA 测试覆盖不足，共享视图调整可能影响其他组件

关联脉络

- PR #47035 Unknown: 引入共享 KV 缓存视图布局不一致回归，本 PR 修复其副作用。