PR #43945 完整报告

vllm-project/vllm

[ROCm][CI] Fix AITER unified attention for encoder-decoder cross-attention

合并时间：2026-05-29 16:43

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/43945

执行摘要

- 一句话：修复 ROCm AITER cross-attention 共享 KV 缓存布局问题
- 推荐动作：推荐合并。修正逻辑清晰、变更最小、已通过回归测试。可作为「架构演进中保持后端兼容性」的典型案例精读：展示了在大规模 layout 标准化过程中，如何通过细粒度的条件分支保护混合使用不同布局的共享资源，避免回归。

功能与动机

修复 vllm-project/vllm#43660 引入的回归。该 PR 标准化了 ROCM_AITER_UNIFIED_ATTN 后端的 KV 缓存布局为 blocks-first，但未考虑 Nemotron Parse 等 encoder-decoder 模型中，decoder 自注意力使用 ROCM_ATTN（仍为 K/V-first 布局）、交叉注意力使用 ROCM_AITER_UNIFIED_ATTN 且共享同一 raw KV allocation 的场景。直接 unbind(1) 会导致 block ID 在不同注意力类型间误解相同的物理字节，从而产生错误的注意力输出。

实现拆解

1. 新增 _split_kv_cache 方法：在 RocmAiterUnifiedAttentionImpl 类中定义新的私有方法 _split_kv_cache(self, kv_cache: torch.Tensor) -> tuple[torch.Tensor, torch.Tensor]，负责将 [num_blocks, 2, block_size, num_kv_heads, head_size] 的 KV 缓存拆分为 key 和 value 张量。
2. 条件分支处理 encoder-decoder 兼容性：方法内部判断 attn_type 是否为 ENCODER_DECODER。若非该类型，则沿用原有的 kv_cache.unbind(1) 从维度 1 拆分（blocks-first）；若为该类型，则通过 as_strided 将缓存视图重新解释为 (2, num_blocks, block_size, num_kv_heads, head_size)，使得 unbind(0) 得到 K/V-first 布局的 key 和 value，与共享分配的 ROCM_ATTN 布局一致。
3. 替换三处直接 k=kv_cache.unbind(1) 调用：在 forward、do_kv_cache_update 和 do_rope_and_kv_cache_update 方法中，将原有代码 key_cache, value_cache = kv_cache.unbind(1) 替换为 key_cache, value_cache = self._split_kv_cache(kv_cache)，确保所有 KV 缓存拆分逻辑统一走条件兼容路径。
4. 无其他文件变更：所有修改集中在 vllm/v1/attention/backends/rocm_aiter_unified_attn.py，仅 1 个文件，+27/-3 行，变更范围极窄，风险低。

关键文件：

- vllm/v1/attention/backends/rocm_aiter_unified_attn.py（模块 注意力后端；类别 source；类型 core-logic；符号 _split_kv_cache）：所有变更在此文件中。新增 _split_kv_cache 方法并替换了 3 处直接 unbind(1) 调用，以支持 encoder-decoder 交叉注意力共享 K/V-first 布局的 KV 缓存。

关键符号：_split_kv_cache

关键源码片段

vllm/v1/attention/backends/rocm_aiter_unified_attn.py

所有变更在此文件中。新增 _split_kv_cache 方法并替换了 3 处直接 unbind(1) 调用，以支持 encoder-decoder 交叉注意力共享 K/V-first 布局的 KV 缓存。

```
# 文件：vllm/v1/attention/backends/rocm_aiter_unified_attn.py
# 新增方法，用于在 encoder-decoder 交叉注意力路径中兼容 shared K/V-first layout

def _split_kv_cache(
    self, kv_cache: torch.Tensor
) -> tuple[torch.Tensor, torch.Tensor]:
    # 对于非 encoder-decoder 类型，直接使用标准 blocks-first 布局
    if self.attn_type != AttentionType.ENCODER_DECODER:
        return kv_cache.unbind(1)

    # 对于 encoder-decoder 交叉注意力：
    # 因与 decoder 自注意力共享同一 raw KV allocation（后者使用 K/V-first），
    # 需通过 as_strided 将当前 blocks-first 视图重新解释为 K/V-first。
    num_blocks, _, block_size, num_kv_heads, head_size = kv_cache.shape
    block_stride = block_size * num_kv_heads * head_size
    kv_cache = kv_cache.as_strided(
        (2, num_blocks, block_size, num_kv_heads, head_size),
        (
            num_blocks * block_stride, # stride for dim 0 (K vs V)
            block_stride, # stride for dim 1 (blocks)
            num_kv_heads * head_size, # stride for dim 2 (block pos)
            head_size, # stride for dim 3 (heads)
            1, # stride for dim 4 (head dim)
        ),
    )
    return kv_cache.unbind(0) # 现在 dim 0 分离 K 和 V

# 替换原有直接调用的例子（forward 方法中）：
# key_cache, value_cache = kv_cache.unbind(1) # 旧代码
key_cache, value_cache = self._split_kv_cache(kv_cache) # 新代码
```

评论区精华

无 review 评论。仅维护者 tjtanaa 单次审批通过，无讨论线程。PR 描述明确给出了回归原因、bisect 结论和复现命令，上下文清晰。

- 暂无高价值评论线程

风险与影响

- 风险：风险极低：1）仅修改单一文件中的单一方法调用；2）变更逻辑由 attn_type 条件分支保护，encoder-decoder 模型之外的路径行为不变；3）as_strided 不复制数据，仅改变视图，无显存开销；4）已通过 Nemotron Parse 测试（test_nemotron_parse.py::test_models）验证修复。潜在风险：as_strided 的步长计算若存在逻辑错误可能导致数据错位，但 PR 中正确推导了步长（block_stride = block_size * num_kv_heads * head_size），且测试已通过。
- 影响：影响范围精确：仅修复 ROCm 平台上使用 ROCM_AITER_UNIFIED_ATTN 后端的 encoder-decoder 多模态模型（如 NVIDIA Nemotron Parse）的 KV 缓存访问错误。其他模型或后端不受影响。对用户而言，该修复恢复了混合后端下交叉注意力的正确性，消除了静默错误可能。
- 风险标记：核心路径变更，缺少测试覆盖

关联脉络

- PR #43660 [Attention][AMD] Standardize kv layout to blocks first for AMD: 本 PR 修复了 PR #43660 在 encoder-decoder 混合后端场景下引入的回归，因此是直接的修复关联。