PR #52737 完整报告

vllm-project/vllm

[ROCm][Perf] Fuse DeepSeek-V4 mHC post/pre and RMSNorm with AITER

合并时间：2026-08-20 12:51

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/52737

执行摘要

- 一句话：ROCm 融合 DSV4 mHC 与 RMSNorm 内核，吞吐约增 1%
- 推荐动作：值得精读，尤其是对 kernel 融合接入模式和 ROCm 后端开发有兴趣的工程师。重点关注三点：_aiter_ops.py 中 mhc_fused_post_pre 的返回值顺序转换与空 token 分支、model.py 中融合能力的条件探测（hc_mult == 4 + hidden size 白名单）、以及 mhc.py 各 forward_hip 中 AITER/TileLang/torch 的三级回退结构。若后续推广类似融合，建议补上覆盖 fake/meta 与回退路径的单元测试，并考虑把 hidden size 白名单收敛为 AITER 侧的查询接口，避免多处硬编码。

功能与动机

PR body 开宗明义：'This PR improves the ROCm DeepSeek-V4 mHC path by wiring the existing AITER fused mHC operators into vLLM'。此前 ROCm 上 mHC 路径以独立 kernel 序列执行，AITER 已提供融合版本却未被 vLLM 使用，存在明显的 kernel launch 与中间访存开销；profile 对比显示融合后关键段 '38.35% faster'。作者以 SA InferenceX 8k1k 负载做 A/B 验证，确认端到端吞吐与 TPOT 一致受益，并在 gsm8k 上验证精度未回退（30-shot 下融合路径 exact_match 0.8681 vs main 0.8461）。

实现拆解

1. 扩展 mHC pre 算子数据契约：涉及 vllm/model_executor/kernels/mhc/aiter.py 与 vllm/_aiter_ops.py，mhc_pre_aiter / rocm_aiter_ops.mhc_pre 新增 norm_weight、norm_eps 可选参数并原样透传给 AITER 内核 aiter.ops.mhc.mhc_pre；_mhc_pre_aiter_fake 同步扩展签名以保持 meta 设备 shape 推断一致。这样 RMSNorm 权重可下推到 pre 内核执行。
2. 新增融合 mHC post + pre 算子封装：aiter.py 注册 mhc_fused_post_pre_aiter custom op 及 _mhc_fused_post_pre_aiter_fake（fake 仅做 shape 推断）；_aiter_ops.py 新增 rocm_aiter_ops.mhc_fused_post_pre 静态方法，负责 dtype/shape 校验、num_tokens == 0 空 tensor 分支、在 torch.device(residual_flat.device) 上下文执行 AITER 内核，并把 AITER 返回顺序 (post_mix, comb_mix, layer_input, next_residual) 重排为 vLLM 调用方顺序 (residual_cur, post_mix, comb_mix, layer_input)。
3. 模型层接入与能力探测：vllm/models/deepseek_v4/amd/model.py 新增 _AITER_MHC_FUSED_RMSNORM_SIZES 白名单（{1280, 2560, 4096, 7168}）；use_fused_mhc 改为 AITER 优先（需要 hidden_size % 256 == 0 且 hc_mult == 4），

否则回退 TileLang；新增 fuse_mhc_rmsnorm 标志，决定 attn/ffn 的 RMSNorm 是否折叠进融合内核，折叠后跳过独立 attn_norm 调用；hc_pre 与 _forward_fused_post_pre透传 norm_weight、norm_eps。

4. 后端分发层收紧条件：vllm/model_executor/layers/mhc.py 的 MHCPreOp.forward_hip、MHCPostOp.forward_hip 在 hidden_size % 256 == 0 之外新增 hc_mult == 4 检查（AITER 内核前置条件，来自 tpopp 的 review），并补齐 n_splits、norm_weight、norm_eps 向 torch.ops.vllm.mhc_pre_aiter 的透传；MHCFusedPostPreOp.forward_hip 优先走 AITER 融合，再回退 TileLang。
5. 配套与测试：vllm/models/deepseek_v4/amd/dspark.py 仅更新模块头注释，说明 use_fused_mhc 在 AITER/TileLang 下为 True、仅 torch fallback 下为 False。PR 未新增自动化测试文件，验证依赖手工 benchmark 与 gsm8k 精度测试，这是主要风险点。

关键文件：

- vllm/model_executor/kernels/mhc/aiter.py（模块 算子层；类别 source；类型 core-logic；符号 mhc_fused_post_pre_aiter, _mhc_fused_post_pre_aiter_fake, mhc_pre_aiter, _mhc_pre_aiter_fake）：新增 AITER 融合 post+pre 自定义算子入口与 fake 实现，并扩展 mhc_pre_aiter 契约，是整个融合能力在 vLLM 侧的注册点。
- vllm/_aiter_ops.py（模块 算子封装；类别 source；类型 core-logic；符号 mhc_fused_post_pre, mhc_pre）：AITER 算子封装核心：新增 mhc_fused_post_pre 静态方法，处理形状校验、空 token 分支与返回值重排，是本 PR 数据契约调整的中枢。
- vllm/models/deepseek_v4/amd/model.py（模块 模型层；类别 source；类型 core-logic；符号 DeepseekV4DecoderLayer, hc_pre, _forward_fused_post_pre）：模型接入层：引入 hidden size 能力白名单与 fuse_mhc_rmsnorm 决策，条件化下推 RMSNorm，控制融合路径的前向行为。
- vllm/model_executor/layers/mhc.py（模块 分发层；类别 source；类型 core-logic；符号 MHCPreOp.forward_hip, MHCPostOp.forward_hip, MHCFusedPostPreOp.forward_hip）：后端分发层：收紧 AITER 前置条件并补齐参数透传，决定各平台（AITER/TileLang/torch）的算子路由。
- vllm/models/deepseek_v4/amd/dspark.py（模块 模型层；类别 source；类型 documentation）：模块头注释更新，说明 use_fused_mhc 语义从 AITER 路径下禁用尾部 mhc_post 变为 AITER/TileLang 下启用、仅 torch fallback 下禁用。

关键符号：mhc_fused_post_pre_aiter, _mhc_fused_post_pre_aiter_fake, mhc_fused_post_pre, mhc_pre, DeepseekV4DecoderLayer.init, DeepseekV4DecoderLayer.hc_pre, DeepseekV4DecoderLayer._forward_fused_post_pre, MHCPreOp.forward_hip, MHCPostOp.forward_hip, MHCFusedPostPreOp.forward_hip

关键源码片段

vllm/model_executor/kernels/mhc/aiter.py

新增 AITER 融合 post+pre 自定义算子入口与 fake 实现，并扩展 mhc_pre_aiter 契约，是整个融合能力在 vLLM 侧的注册点。

```
def mhc_fused_post_pre_aiter(
```

```
    x: torch.Tensor,
    residual: torch.Tensor,
    post_layer_mix: torch.Tensor,
    comb_res_mix: torch.Tensor,
    fn: torch.Tensor,
    hc_scale: torch.Tensor,
    hc_base: torch.Tensor,
    rms_eps: float,
    hc_pre_eps: float,
    hc_sinkhorn_eps: float,
    hc_post_mult_value: float,
    sinkhorn_repeat: int,
    n_splits: int = 1,
    tile_n: int = 1,
    norm_weight: torch.Tensor | None = None,
    norm_eps: float = 0.0,
) -> tuple[torch.Tensor, torch.Tensor, torch.Tensor, torch.Tensor]:
    """ROCm 上通过 AITER 执行融合的 mHC post + 下一个 mHC pre。

    返回 vLLM 顺序的元组 (residual_cur, post_mix_cur, comb_mix_cur, layer_input_cur),
    供模型层直接替换原来独立的 mhc_post 与 mhc_pre 调用。
    """
    hidden_size = residual.shape[-1]
    assert hidden_size % 256 == 0 # AITER 内核要求的对齐条件
    from vllm._aiter_ops import rocm_aiter_ops

    return rocm_aiter_ops.mhc_fused_post_pre(
        x,
        residual,
        post_layer_mix,
        comb_res_mix,
        fn,
        hc_scale,
        hc_base,
        rms_eps,
        hc_pre_eps,
        hc_sinkhorn_eps,
        hc_post_mult_value,
        sinkhorn_repeat,
        norm_weight,
        norm_eps,
    )


def _mhc_fused_post_pre_aiter_fake(
    x: torch.Tensor,
    residual: torch.Tensor,
    post_layer_mix: torch.Tensor,
    comb_res_mix: torch.Tensor,
```

```
    fn: torch.Tensor,
    hc_scale: torch.Tensor,
    hc_base: torch.Tensor,
    rms_eps: float,
    hc_pre_eps: float,
    hc_sinkhorn_eps: float,
    hc_post_mult_value: float,
    sinkhorn_repeat: int,
    n_splits: int = 1,
    tile_n: int = 1,
    norm_weight: torch.Tensor | None = None,
    norm_eps: float = 0.0,
) -> tuple[torch.Tensor, torch.Tensor, torch.Tensor, torch.Tensor]:
    # fake 实现只做 shape 推导：AITER 的 Python 包装器在分配中间张量时
    # 不带显式 device 参数，meta 设备或编译阶段依赖 fake 输出来推断形状。
    hc_mult = residual.shape[-2]
    hidden_size = residual.shape[-1]
    outer_shape = residual.shape[:-2]

    post_mix = torch.empty(
        *outer_shape, hc_mult, 1, dtype=torch.float32, device=residual.device
    )
    comb_mix = torch.empty(
        *outer_shape, hc_mult, hc_mult, dtype=torch.float32, device=residual.device
    )
    layer_input = torch.empty(
        *outer_shape, hidden_size, dtype=torch.bfloat16, device=residual.device
    )
    next_residual = torch.empty_like(residual)
    return next_residual, post_mix, comb_mix, layer_input
```

vllm/models/deepseek_v4/amd/model.py

模型接入层：引入 hidden size 能力白名单与 fuse_mhc_rmsnorm 决策，条件化下推 RMSNorm，控制融合路径的前向行为。

```
# AITER mhc_pre_big_fuse_rmsnorm 支持的 hidden size 白名单；
# 白名单之外的尺寸仍走独立的 RMSNorm 内核，保证正确性优先。
_AITER_MHC_FUSED_RMSNORM_SIZES = frozenset({1280, 2560, 4096, 7168})


class DeepseekV4DecoderLayer(nn.Module):
    def __init__(
        self,
        vllm_config,
        prefix,
        topk_indices_buffer: torch.Tensor | None = None,
        aux_stream_list: list[torch.cuda.Stream] | None = None,
    ):
        ...
```

```
        self.mhc_pre = MHCPreOp()
        self.mhc_post = MHCPostOp()
        self.mhc_fused_post_pre = MHCFusedPostPreOp()

        # AITER mHC 内核（pre/post/fused）要求 hc_mult == 4，
        # 该约束来自 AITER mhc_kernels.cu 的核函数实现，缺失会导致未定义行为
        use_aiter_mhc = (
            HAS_AITER_MHC and self.hidden_size % 256 == 0 and self.hc_mult == 4
        )
        # 融合路径优先级：AITER > TileLang > torch 原生
        self.use_fused_mhc = use_aiter_mhc or HAS_TILELANG_MHC
        # 只有当前后端支持该 hidden size 的融合 RMSNorm 时才下推 norm 权重
        if use_aiter_mhc:
            self.fuse_mhc_rmsnorm = self.hidden_size in _AITER_MHC_FUSED_RMSNORM_SIZES
        else:
            self.fuse_mhc_rmsnorm = HAS_TILELANG_MHC and self.use_fused_mhc

    def _forward_fused_post_pre(
        self,
        x: torch.Tensor,
        positions: torch.Tensor,
        input_ids: torch.Tensor | None,
        post_mix: torch.Tensor | None = None,
        res_mix: torch.Tensor | None = None,
        residual: torch.Tensor | None = None,
    ) -> tuple[torch.Tensor, torch.Tensor, torch.Tensor, torch.Tensor]:
        # 按能力探测结果决定是否把 attn RMSNorm 折叠进融合内核
        attn_norm_weight = self.attn_norm.weight if self.fuse_mhc_rmsnorm else None
        attn_norm_eps = (
            self.attn_norm.variance_epsilon if self.fuse_mhc_rmsnorm else 0.0
        )

        if residual is None:
            # 首层没有上一层残差，走独立的 hc_pre（仍可携带 RMSNorm 权重）
            residual = x
            x, post_mix, res_mix = self.hc_pre(
                x,
                self.hc_attn_fn,
                self.hc_attn_scale,
                self.hc_attn_base,
                norm_weight=attn_norm_weight,
                norm_eps=attn_norm_eps,
            )
        else:
            residual, post_mix, res_mix, x = self.mhc_fused_post_pre(
                x,
                residual,
                post_mix,
                res_mix,
```

```
            self.hc_attn_fn,
            self.hc_attn_scale,
            self.hc_attn_base,
            self.rms_norm_eps, # rms_eps
            self.hc_eps, # hc_pre_eps
            self.hc_eps, # hc_sinkhorn_eps
            self.hc_post_alpha, # hc_post_mult_value
            self.hc_sinkhorn_iters, # sinkhorn_repeat
            norm_weight=attn_norm_weight,
            norm_eps=attn_norm_eps,
        )

    # 融合路径下 RMSNorm 已由内核完成，跳过独立调用避免重复计算
    if not self.fuse_mhc_rmsnorm:
        x = self.attn_norm(x)
    x = self.attn(positions, x, None)
    ...
```

评论区精华

tpopp 在 review 中指出 AITER 的 mHC 内核在 mhc_kernels.cu:2324 处隐含 hc_mult == 4 的前置条件，建议 model.py 同步检查 config.hc_mult == 4；作者在后续 commit 中补上该检查，并说明当前 DSV4 默认 hc_mult = 4，因此对现有测试结果没有影响。tpopp 还分享了一份几乎同时完成的参考实现，并提示单独修改 aiter.ops.mhc.mhc_pre 'actually hurts perf'，佐证了采用整体融合而非局部改写方向。合入时 tjtanaa 给出 LGTM，但留言希望 AMD 团队继续跟进准确率问题（lmeval 0.95 ± 0.01 @ conc 256、30-shot），该疑虑未在 PR 内完全闭环。

- AITER 内核 hc_mult == 4 前置条件检查 (correctness): 作者在后续 commit 中为 use_aiter_mhc、MHCPreOp.forward_hip、MHCPostOp.forward_hip 统一补上 hc_mult == 4 检查，并说明当前 DSV4 默认 hc_mult 就是 4，因此对现有结果无影响。
- 并行参考实现与 mhc_pre 独立改动的性能反效果 (performance): 作者采用 AITER mhc_fused_post_pre 整体融合方案而非修改 mhc_pre 单算子，与参考实现的性能结论一致。
- 合入前准确率问题确认 (testing): PR 已合并，但准确率疑虑未在 PR 内完全闭环，需 AMD 侧后续跟进确认。

风险与影响

- 风险：无自动化测试：5 个改动文件全部为源码，PR 未配套任何测试（如 fake/ 空 token 分支形状一致性、hc_mult != 4 时的回退路径），后续回归只能依赖手工 benchmark。硬编码白名单：_AITER_MHC_FUSED_RMSNORM_SIZES 是离线的 hidden size 集合，未来新尺寸模型会静默走 standalone RMSNorm，性能落差不会报错；若 AITER 内核后续支持新尺寸而白名单未更新，融合机会被浪费。空 token 分支语义：mhc_fused_post_pre 的 num_tokens == 0 分支返回 torch.empty_like(residual_flat).view_as(residual) 作为 next_residual，与 fake 实现基于 residual.shape[:-2] 的形状推断存在隐含的一致性假设

，一旦外维形状推断不一致会造成隐性 shape 错误。TTFT 波动：conc=1 时 Mean TTFT 上升 9.86%，融合路径在低并发下的延迟特征不同，若客户场景以低并发为主需重新评估收益口径。数值精度：合入时审阅者仍要求 AMD 侧确认准确率，虽然 gsm8k 30-shot 结果融合方向反而更高（0.8681 vs 0.8461），但单任务不足以覆盖数值风险。

- 影响：用户 / 产品：仅影响 ROCm + AITER + DeepSeek-V4 用户，端到端吞吐提升约 1%、TPOT 改善约 1%，profile 级关键段提升 38%；CUDA/XPU 行为不变，无 API 变化。系统：新增 mhc_fused_post_pre_aiter 自定义 op（含 fake 注册），扩展了 mhc_pre_aiter 的 kernel 层数据契约，任何直接调用该 op 的代码（当前主要是 MHCPreOp.forward_hip）需要同步参数。团队：为后续 ROCm kernel 融合提供了可复制的模式（能力探测白名单 + 自定义 op fake 实现 + 三级后端回退 + 参数透传），模型层与 kernel 层的协作边界更清晰。
- 风险标记：缺少自动化测试覆盖，硬编码 hidden size 白名单，精度疑虑未闭环，AITER 前置条件依赖 hc_mult == 4，低并发下 TTFT 上升

关联脉络

- PR #53004 [ROCm][CI] Speed up test_rocm_aiter_qk_norm_rope_kvcache_fusion: 同属 ROCm AITER 融合算子路径的系统化工作，将相关 CI 测试从 2.5 小时裁剪到 6 分钟，与本 PR 一前一后打通 ' 融合实现 + 快速验证 ' 的闭环。
- PR #52839 [refactor] consolidate cp attn ops: DeepSeek 系列注意力算子整合重构，梳理了模型层算子分发抽象，与本 PR 在 deepseek 模型家族上的后端算子路由演进同向。
- PR #53021 [Model] Remove unused DeepseekV32Indexer forward: 同为 DeepSeek 家族模型层清理，反映 DeepSeek V4 相关代码正在快速演进与收敛。