

PR #53540 完整报告

vllm-project/vllm

[ROCm][Perf] Fuse SWA q/kv RMSNorm and q FP8 group quant for DeepSeek-V4

合并时间: 2026-08-28 00:37

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/53540>

执行摘要

- 一句话: 融合 DSV4 q/kv RMSNorm 与 FP8 量化, ROCm 热路径 2.9 倍提速
- 推荐动作: 值得精读。核心看两点: 一是如何用『类型门控 + cached_property』在共享代码路径上做平台特化且保证 fallback; 二是 pre-quantized activation 直接驱动 block-scaled GEMM 的数据契约设计, 避免 apply_weights 二次量化造成的双重舍入。配套测试中的 ULP 对比与 strided view 用例也很有参考价值。

功能与动机

PR body 指出每个 decode step 的 SWA token 插入管线在 ROCm 上要执行三步: Triton `fused_q_kv_rmsnorm` 归一化、aiter per-1x128 动态量化 bf16 q-lora (喂给 block-scaled FP8 `wq_b` GEMM)、以及 indexer 的 `wq_b` 对共享 q-lora 的第二次相同量化 ('the same quant a second time in the indexer's `wq_b`'). fusion 后量化直接从 fp32 累加器读取, 'singlerounding, equivalent-or-better precision'。实测热路径从 3 次 launch (14 μ s 277ns) 降为 1 次 (4 μ s 919ns, 2.90x)。

实现拆解

本 PR 通过 4 个步骤完成融合, 核心思路是把「归一化 + 量化」下沉到单个 AITER 内核, 并把量化结果作为数据契约在 attention 与 indexer 之间共享。

1. 注册融合 custom op: 在 `vllm/_aiter_ops.py` 中新增 `_fused_mla_dual_rms_norm_group_quant_impl` 与配套的 `_fake` 实现, 在 `register_ops_once()` 中注册名为 `fused_mla_dual_rms_norm_group_quant` 的 op, 并新增静态访问器 `fused_qk_rmsnorm_group_quant`。real 实现调用 `aiter.ops.fused_qk_rmsnorm_group_quant`, 一次 pass 完成 q 的 RMSNorm + FP8 group quant (输出 (M, N/128) 的 fp32 scales) 与 kv 的 RMSNorm; fake 实现仅分配相同形状的占位张量, 保证元设备 / 编译路径可用。
2. 重构共享注意力路径: 在 `vllm/models/deepseek_v4/attention.py` 中把 `forward` 里原先内联的 `split + fused_q_kv_rmsnorm` 抽成 `_split_qkv_and_norm` (共享路径返回 `qr_scale=None`), 并把 `qr_scale` 作为新参数贯穿 `_prepare_and_attn_eager` / `_prepare_and_attn`, 同时传给 indexer 调用; `wq_b` 投影改为 `_wq_b_proj`, 当 `qr_scale` 为 None 时等价于原 `self.wq_b(qr)`, 从而保证 CUDA/XPU 路径行为完全不变。

3. ROCm 特化与条件门控：在 `vllm/models/deepseek_v4/amd/rocm.py` 中覆写 `_split_qkv_and_norm`，并用 `functools.cached_property` 实现 `_wq_b_uses_aiter_block_scaled`——仅当 aiter 的 fp8 linear 开关开启、且 attention 的 `wq_b` 与 `indexer` 的 `wq_b` 都是 `Fp8BlockScaledMMLinearKernel` 时才启用融合路径，否则回退共享实现。启用时返回预量化的 fp8 `qr` 与 `qr_scale`，`_wq_b_proj` 侧通过新增的 `apply_pre_quantized_block_scaled_mm` 直连 linear 的 block-scaled GEMM，绕过 `apply_weights` 的二次量化；该函数仅对 column/replicated 类 linear 有效（输出即本地 TP 分片，无需 all-reduce）。
4. 测试配套：在 `tests/kernels/core/test_rocm_aiter_ops.py` 中新增两个测试：
`test_rocm_aiter_mla_dual_rmsnorm_group_quant_vs_sequential` 验证融合 op 与 `rms_norm -> group_fp8_quant` 序列在反量化域的误差预算（M=1 覆盖单 token decode，kv 侧约束 1 个 bf16 ULP）；`test_rocm_aiter_mla_dual_rmsnorm_group_quant_strided_split_view` 验证 DeepSeek-V4 实际产生的 split-view (strided) 输入与 contiguous 输入结果一致，并引入 `fp8_ulp_distance` 工具。

关键文件：

- `vllm/models/deepseek_v4/amd/rocm.py`（模块 ROCm 后端；类别 source；类型 data-contract；符号 `apply_pre_quantized_block_scaled_mm`，`_wq_b_uses_aiter_block_scaled`，`_split_qkv_and_norm`）：ROCm 特化主路径：覆写 `_split_qkv_and_norm`，新增 `_wq_b_uses_aiter_block_scaled` 条件门控与 `apply_pre_quantized_block_scaled_mm` 预量化 GEMM 入口，是本次融合能否启用的决策点。
- `vllm/_aiter_ops.py`（模块 算子注册；类别 source；类型 core-logic；符号 `_fused_mla_dual_rms_norm_group_quant_impl`，`_fused_mla_dual_rms_norm_group_quant_fake`，`fused_qk_rmsnorm_group_quant`）：新增 `fused_mla_dual_rms_norm_group_quant` custom op 的 real/fake 实现与注册，是融合内核进入 vLLM 算子体系的入口。
- `vllm/models/deepseek_v4/attention.py`（模块 注意力层；类别 source；类型 data-contract；符号 `_split_qkv_and_norm`，`_wq_b_proj`）：共享注意力路径的数据契约调整：抽出 `_split_qkv_and_norm` 与 `_wq_b_proj`，`qr_scale` 贯穿至 `indexer`，是所有平台共用的基础改动。
- `tests/kernels/core/test_rocm_aiter_ops.py`（模块 内核测试；类别 test；类型 test-coverage；符号 `test_rocm_aiter_mla_dual_rmsnorm_group_quant_vs_sequential`，`test_rocm_aiter_mla_dual_rmsnorm_group_quant_strided_split_view`，`run`）：新增两个内核级测试，验证融合 op 与序列化实现的数值等价性及 strided split-view 输入一致性，是精度回归的守门员。

关键符号：`_fused_mla_dual_rms_norm_group_quant_impl`，`_fused_mla_dual_rms_norm_group_quant_fake`，`fused_qk_rmsnorm_group_quant`，`_split_qkv_and_norm`，`_wq_b_proj`，`_wq_b_uses_aiter_block_scaled`，`apply_pre_quantized_block_scaled_mm`，`test_rocm_aiter_mla_dual_rmsnorm_group_quant_vs_sequential`，`test_rocm_aiter_mla_dual_rmsnorm_group_quant_strided_split_view`

关键源码片段

vllm/models/deepseek_v4/amd/rocm.py

ROCm 特化主路径：覆写 `_split_qkv_and_norm`，新增 `_wq_b_uses_aiter_block_scaled` 条件门控与 `apply_pre_quantized_block_scaled_mm` 预量化 GEMM 入口，是本次融合能否启用的决策点。

```
# vllm/models/deepseek_v4/amd/rocm.py
# 对预量化激活执行 block-scaled fp8 GEMM：融合内核写出 fp8 qr 与 per-1x128 scales，
# 本函数用它们直接驱动 linear 的 block-scaled GEMM，绕过 apply_weights 对 fp8
输入的重复量化。
# 仅适用于 wq_b 这类 column/replicated linear：其输出就是本地 TP 分片，无需 all-reduce。
def apply_pre_quantized_block_scaled_mm(
    linear: torch.nn.Module,
    x_fp8: torch.Tensor,
    x_scale: torch.Tensor,
) -> torch.Tensor:
    from vllm.model_executor.kernels.linear.scaled_mm.BlockScaledMMLinearKernel import (
        FP8BlockParams,
    )

    # 从 linear 的量化参数中取权重 block scale（兼容 weight_scale_inv 表示）。
    params = FP8BlockParams.from_layer(linear)
    weight_scale = (
        params.weight_scale
        if params.weight_scale_inv is None
        else params.weight_scale_inv
    )
    kernel = linear.quant_method.fp8_linear
    out = kernel.apply_block_scaled_mm(
        A=x_fp8, B=params.weight, As=x_scale, Bs=weight_scale
    )
    return out.to(dtype=kernel.config.out_dtype)

# 两个 wq_b GEMM 都走 aiter block-scaled fp8 内核时才启用融合路径。
# 模型构建后 linear kernel 与 aiter 开关即固定，故在首个 forward 时求值并缓存。
# 若 quant 或 GEMM 与 aiter 实现不一致，则回退共享路径，保证正确性优先。
@functools.cached_property
def _wq_b_uses_aiter_block_scaled(self) -> bool:
    from vllm._aiter_ops import rocm_aiter_ops
    from vllm.model_executor.kernels.linear.scaled_mm import (
        Fp8BlockScaledMMLinearKernel,
    )

    if not rocm_aiter_ops.is_linear_fp8_enabled():
        return False

    # attention 与 indexer 两侧的 wq_b 必须都是 block-scaled kernel,
```

```

# 否则预量化 (qr_fp8, qr_scale) 契约不成立。
linears = [self.wq_b]
if self.indexer is not None:
    linears.append(self.indexer.wq_b)
for linear in linears:
    kernel = getattr(getattr(linear, "quant_method", None), "fp8_linear", None)
    if not isinstance(kernel, Fp8BlockScaledMMLinearKernel):
        return False
return True

```

vllm/_aiter_ops.py

新增 `fused_mla_dual_rms_norm_group_quant` custom op 的 real/fake 实现与注册，是融合内核进入 vLLM 算子体系的入口。

```

# vllm/_aiter_ops.py
# AITER 融合 MLA q/kv RMSNorm 与 q 的 FP8 分组量化。
# 支撑 DeepSeek-V4 ROCm attention 路径的 custom op fused_mla_dual_rms_norm_group_quant:
# q latent 直接从 fp32 归一化累加器量化（跳过 bf16 中间表示，单次舍入），
# kv latent 只做 RMSNorm 到 bf16，供 fused insert 内核使用。
def _fused_mla_dual_rms_norm_group_quant_impl(
    q: torch.Tensor,
    q_weight: torch.Tensor,
    kv: torch.Tensor,
    kv_weight: torch.Tensor,
    q_epsilon: float,
    kv_epsilon: float,
    group_size: int,
    transpose_scale: bool,
) -> tuple[torch.Tensor, torch.Tensor, torch.Tensor]:
    from aiter.ops.fused_qk_rmsnorm_group_quant import (
        fused_qk_rmsnorm_group_quant,
    )

    mq, nq = q.shape
    q_out = torch.empty((mq, nq), dtype=FP8_DTYPE, device=q.device)
    q_scale = torch.empty((mq, nq // group_size), dtype=torch.float32, device=q.device)
    kv_normed = torch.empty(kv.shape, dtype=kv.dtype, device=kv.device)

    # split 视图可直接传入（单元内 stride），内核自行处理 strided 输入。
    fused_qk_rmsnorm_group_quant(
        q_out_quantized=q_out,
        q_out_scale=q_scale,
        q=q,
        q_weight=q_weight,
        q_epsilon=q_epsilon,
        k_out=kv_normed,
        k=kv,
        k_weight=kv_weight,
    )

```

```

        k_epsilon=kv_epsilon,
        group_size=group_size,
        transpose_scale=transpose_scale,
    )
    return q_out, q_scale, kv_normed

```

fake 实现：仅供元设备 / 编译追踪使用，形状与 dtype 与真实实现一致。

```

def _fused_mla_dual_rms_norm_group_quant_fake(
    q: torch.Tensor,
    q_weight: torch.Tensor,
    kv: torch.Tensor,
    kv_weight: torch.Tensor,
    q_epsilon: float,
    kv_epsilon: float,
    group_size: int,
    transpose_scale: bool,
) -> tuple[torch.Tensor, torch.Tensor, torch.Tensor]:
    mq, nq = q.shape
    q_out = torch.empty((mq, nq), dtype=FP8_DTYPE, device=q.device)
    q_scale = torch.empty((mq, nq // group_size), dtype=torch.float32, device=q.device)
    kv_normed = torch.empty(kv.shape, dtype=kv.dtype, device=kv.device)
    return q_out, q_scale, kv_normed

```

vllm/models/deepseek_v4/attention.py

共享注意力路径的数据契约调整：抽出 `_split_qkv_and_norm` 与 `_wq_b_proj`，`qr_scale` 贯穿至 `indexer`，是所有平台共用的基础改动。

```

# vllm/models/deepseek_v4/attention.py
# 拆分 fused q-lora / kv 投影并对两半分别做 RMSNorm。
# qr_scale 在共享路径上恒为 None；ROCm 子类覆写本方法后，
# 返回预量化的 fp8 qr 及其 per-1x128 fp32 scales，供下游 wq_b 投影直接消费。
def _split_qkv_and_norm(
    self, qr_kv: torch.Tensor
) -> tuple[torch.Tensor, torch.Tensor | None, torch.Tensor]:
    qr, kv = qr_kv.split([self.q_lora_rank, self.head_dim], dim=-1)
    qr, kv = fused_q_kv_rmsnorm(
        qr,
        kv,
        self.q_norm.weight.data,
        self.kv_norm.weight.data,
        self.eps,
    )
    return qr, None, kv

```

将 q-lora 输入投影到 query heads。

qr_scale 仅由 ROCm 融合路径设置：此时 qr 已是 fp8 量化且带 per-1x128 scales，

必须绕过 linear 内部的再次量化（`apply_weights` 会把 fp8 输入再量化一次）。

```
def _wq_b_proj(
    self, qr: torch.Tensor, qr_scale: torch.Tensor | None = None
) -> torch.Tensor:
    if qr_scale is None:
        # 共享路径：维持原有行为，由 linear 自己完成量化。
        return self.wq_b(qr)
    # ROCm 融合路径：直连 block-scaled GEMM，跳过 apply_weights。
    return apply_pre_quantized_block_scaled_mm(self.wq_b, qr, qr_scale)
```

评论区精华

Review 中唯一的技术交锋集中在 `vllm/models/deepseek_v4/attention.py` 的 `_wq_b_proj`：tjtanaa 指出 'This function seems to be duplicated.'。作者解释有两个 `_wq_b_proj`（SWA 的 `wq_b` 与 indexer 的 `wq_b`），融合内核启用后两侧 linear 都必须跳过输入量化，因此重复是刻意设计。tjtanaa 还要求性能数据，作者补充了 kernel profile 前后对比与端到端 A/B benchmark，并获 AndreasKaratzas 批准合并。

- `_wq_b_proj` 方法重复问题 (design): 作者回应：SWA 的 `wq_b` 与 indexer 的 `wq_b` 是两个独立 linear，融合内核启用后两侧都必须跳过输入量化，因此各自需要该分支，属于刻意设计。
- 性能数据补充 (question): 数据补充后获 reviewer 认可，并由 AndreasKaratzas 批准合并。

风险与影响

- 风险：主要风险集中在数值与兼容性上。第一，量化源从 bf16 中间值变为 fp32 累加器直接量化，属单次舍入，理论精度更优，但输入分布存在微小漂移，GSM8K 实测 0.9431 达标，其他数据集未被覆盖。第二，`attention.py` 的 `forward / _prepare_and_attn` 签名变化影响所有平台，`qr_scale=None` 分支提供保底，但若 ROCm 子类返回 fp8 qr 而调用链某处遗漏传 `qr_scale`，会导致 `apply_weights` 对 fp8 输入二次量化，属隐性正确性风险；该风险由 `_wq_b_uses_aiter_block_scaled` 门控与贯穿式参数缓解，但测试仅覆盖 op 层面而非完整 attention 集成。第三，AITER 内核版本演进可能改变数值行为，ULP 约束测试在 aiter 不可用时直接 skip，CI 覆盖有限。第四，`fused_qk_rmsnorm_group_quant` 无零行保护，依赖调用方保证 `num_tokens > 0`。第五，benchmark 中并发 1 时 TTFT 恶化 10.29% (462.05ms vs 418.93ms)，作者未解释原因，建议关注低并发场景的调度开销变化。
- 影响：用户侧：在 AMD MI350X/MI355X 等支持 AITER FP8 的 ROCm 环境下，DeepSeek-V4 SWA 解码热路径 kernel 级提速约 2.90x，端到端吞吐提升 0.5%~0.8%、TPOT 下降约 0.5%~1%，长上下文场景（indexer 参与）额外省去第三次量化 launch；非 ROCm、非 AITER 或 kernel 类型不符时按门控回退旧路径，行为不变。系统侧：`_aiter_ops.py` 增加了新的 custom op 注册模板，`attention.py` 的数据契约扩展（`qr_scale` 贯穿）为后续 MLA 类模型的预量化 GEMM 接入提供了通用接口。团队侧：该模式可复用于其他 ROCm 后端的归一化 + 量化融合优化。
- 风险标记：ROCm/AITER 特定路径，条件门控依赖 kernel 类型，共享 attention 签名变更，单次舍入精度行为漂移，低并发 TTFT 上升 10%

关联脉络

- PR #53396 [Kimi K3][Kernel] Support DS conv-state layout in fused KDA decode kernel: 同属 GPU 后端 fused kernel 性能优化脉络 (Kimi K3 融合 KDA decode kernel 补齐 DS 布局), 与本 PR 的 kernel fusion 方法论同源, 但无文件或模块交集, 仅作对照参考。