PR #43474 完整报告

vllm-project/vllm

[Kernel] Add mhc_pre_big_fuse_with_norm_tilelang

合并时间：2026-05-25 09:19

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/43474

执行摘要

- 一句话：融合 RMSNorm 的 MHC pre 内核与 DeepSeek V4 MoE 重构
- 推荐动作：值得精读，尤其关注 TileLang 内核如何实现算子融合，以及高层模型如何配合 kernel 接口传递额外参数。该 PR 展示了从 kernel 到模型层的完整融合优化链路。

功能与动机

该 PR 从 sglang 项目复制 kernel 实现（https://github.com/sgl-project/sglang/blob/main/python/sglang/srt/layers/mhc.py#L478），目的是通过融合 RMSNorm 减少 kernel launch 和显存读写，提升 MHC pre-block 的推理效率。同时在 DeepSeek V4 模型上验证了精度无损。

实现拆解

1. 新增 TileLang 融合内核：在 vllm/_tilelang_ops.py 中定义 mhc_pre_big_fuse_with_norm_tilelang，该内核在计算 layer_input = sum_i pre_mix_i * residual_i 的同时应用 RMSNorm（使用 norm_weight 和 norm_eps），从而避免单独的 normalization kernel。
2. 修改 kernel 调度函数：在 vllm/model_executor/kernels/mhc/tilelang.py 的 mhc_pre_tilelang 中根据 norm_weight 是否为 None 选择调用旧内核或新融合内核，同时将 norm_weight/norm_eps 参数透传到 mhc_fused_post_pre_tilelang 等函数。
3. 更新 MHC 层接口：在 vllm/model_executor/layers/mhc.py 中，MHCPreOp.forward_cuda 和 MHCFusedPostPreOp.forward_cuda 增加 norm_weight 和 norm_eps 参数，并通过 custom op 映射到 tilelang kernel。
4. 重构 DeepSeek V4 MoE：在 vllm/models/deepseek_v4/nvidia/model.py 中，将 NormGateLinear 替换为 GateLinear + 独立 RMSNorm，在 MoE forward 时将 norm_weight 显式传递给 MHC 层，从而利用融合内核。
5. 调整 MTP 权重映射：在 vllm/models/deepseek_v4/nvidia/mtp.py 中更新权重 remapping 逻辑，将旧前缀 .ffn.norm_gate 更新为 .ffn.gate 以匹配重构后的结构。

关键文件：

- vllm/_tilelang_ops.py（模块 TileLang 内核；类别 source；类型 core-logic；符号 mhc_pre_big_fuse_with_norm_tilelang）：核心改动，新增 mhc_pre_big_fuse_with_norm_tilelang 内核，实现 RMSNorm 与 MHC pre-block 的融合计算。

- vllm/models/deepseek_v4/nvidia/model.py（模块 DSv4 模型；类别 source；类型 data-contract；符号 DeepseekV4MoE, GateLinear, RMSNorm）：重构 MoE 模块，以支持传递 norm_weight 给 MHC 内核。
- vllm/model_executor/kernels/mhc/tilelang.py（模块 MHC Kernel；类别 source；类型 data-contract；符号 mhc_pre_tilelang, mhc_fused_post_pre_tilelang, _mhc_pre_tilelang_fake）：修改 mhc_pre_tilelang 函数，根据 norm_weight 是否提供分发到不同 tilelang kernel。
- vllm/model_executor/layers/mhc.py（模块 MHC 层；类别 source；类型 data-contract；符号 MHCPreOp.forward_cuda, MHCFusedPostPreOp.forward_cuda）：MHCPreOp 和 MHCFusedPostPreOp 增加 norm_weight/eps 参数以传递到 kernel。
- vllm/models/deepseek_v4/nvidia/mtp.py（模块 MTP 权重；类别 source；类型 data-contract）：调整权重映射以匹配 MoE 内部结构变更（从 norm_gate 到 gate）。

关键符号：mhc_pre_big_fuse_with_norm_tilelang, mhc_pre_tilelang, mhc_fused_post_pre_tilelang, _mhc_pre_tilelang_fake, MHCPreOp.forward_cuda, MHCFusedPostPreOp.forward_cuda, DeepseekV4MoE.init, DeepseekV4MoE.forward

关键源码片段

vllm/_tilelang_ops.py

核心改动，新增 mhc_pre_big_fuse_with_norm_tilelang 内核，实现 RMSNorm 与 MHC pre-block 的融合计算。

```
# SPDX-License-Identifier: Apache-2.0
# 本函数从 sglang 移植，将 RMSNorm 融合到 MHC pre-block 中
# 主要输入：gemm_out_mul, gemm_out_sqrsum（GEMM 输出）, norm_weight（RMSNorm 权重）
# 主要输出：post_mix, comb_mix, layer_input（已应用 RMSNorm）

@tilelang.jit(
    pass_configs={
        tilelang.PassConfigKey.TL_DISABLE_WARP_SPECIALIZED: True,
        tilelang.PassConfigKey.TL_DISABLE_TMA_LOWER: True,
        tilelang.PassConfigKey.TL_PTXAS_REGISTER_USAGE_LEVEL: 10,
    },
)
def mhc_pre_big_fuse_with_norm_tilelang(
    gemm_out_mul, gemm_out_sqrsum, hc_scale, hc_base,
    residual, post_mix, comb_mix, layer_input, norm_weight,
    hidden_size: int, rms_eps: float, hc_pre_eps: float,
    hc_sinkhorn_eps: float, hc_post_mult_value: float,
    sinkhorn_repeat: int, norm_eps: float,
    n_splits: int = 16, hc_mult: int = 4, gemm_last_dim: int = -1,
):
    num_tokens = T.dynamic("num_tokens")
    hc_mult3 = hc_mult * (2 + hc_mult)
    if gemm_last_dim < 0:
        gemm_last_dim = hc_mult3
```

```
    hidden_block = math.gcd(1024, hidden_size)

    # 声明张量类型
    gemm_out_mul: T.Tensor[[n_splits, num_tokens, gemm_last_dim], T.float32]
    gemm_out_sqrsum: T.Tensor[[n_splits, num_tokens], T.float32]
    norm_weight: T.Tensor[[hidden_size], T.bfloat16]
    # ... 其他张量声明省略

    with T.Kernel(num_tokens, threads=96) as i:
        rms = T.alloc_fragment(1, T.float32)
        mixes = T.alloc_fragment(hc_mult3, T.float32)
        T.clear(mixes)
        rms[0] = 0

        # 累积所有 split 的平方和用于计算 RMS
        for i_split in T.serial(n_splits):
            rms[0] += gemm_out_sqrsum[i_split, i]
        rms[0] = T.rsqrt(rms[0] / (hc_mult * hidden_size) + rms_eps)

        # 计算 mix logits（经 RMS scale）
        for j in T.Parallel(hc_mult3):
            mixes[j] = 0
            for i_split in T.serial(n_splits):
                mixes[j] += gemm_out_mul[i_split, i, j]
            mixes[j] *= rms[0]

        # 计算 post_mix、comb_mix（Sinkhorn 迭代）...
        #  （省略中间计算，聚焦关键融合点）

        # 融合 write：将 layer_input 与 norm_weight 逐元素乘实现 RMSNorm
        # 这是与旧内核的核心区别——此处还应用了 norm_weight
        for i0_h in T.Pipelined(hidden_size // hidden_block, num_stages=2):
            # ... 累积 ol 后，应用 RMSNorm
            for i1_h in T.Parallel(hidden_block):
                ol[i1_h] = ol[i1_h] * norm_weight[i0_h * hidden_block + i1_h]
            T.copy(ol, layer_input[i, i0_h * hidden_block])
        T.pdl_trigger()
```

vllm/models/deepseek_v4/nvidia/model.py

重构 MoE 模块，以支持传递 norm_weight 给 MHC 内核。

```
  # 原代码使用 NormGateLinear（内部包含 RMSNorm + gate），现拆分为独立层
  # self.norm_gate = NormGateLinear(...) # 旧
  # 新代码：
  self.gate = GateLinear(
      input_size=config.hidden_size,
      output_size=config.n_routed_experts,
      bias=False,
      out_dtype=torch.float32,
```

```
    prefix=f"{prefix}.gate",
)
# 独立的 RMSNorm 权重（与原来 norm_gate 中的 norm 相同）
self.ffn_norm = RMSNorm(config.hidden_size, eps=config.rms_norm_eps)
# 在 forward 中，将 ffn_norm.weight 作为 norm_weight 参数传入 MHC pre 层
# 调用示例：
post_mix, comb_mix, layer_input = MHCPreOp.forward_cuda(
    residual, fn, hc_scale, hc_base, ...,
    norm_weight=self.ffn_norm.weight,
    norm_eps=self.ffn_norm.variance_epsilon,
)
# 同时将原本位于 norm_gate 上的 e_score_correction_bias 和 tid2eid 移至 gate 对象
self.gate.e_score_correction_bias = ...
self.gate.tid2eid = ...
```

vllm/model_executor/kernels/mhc/tilelang.py

修改 mhc_pre_tilelang 函数，根据 norm_weight 是否提供分发到不同 tilelang kernel。

```
def mhc_pre_tilelang(
    residual: 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,
    norm_weight: torch.Tensor | None = None, # 新增
    norm_eps: float = 1e-6, # 新增
) -> tuple[torch.Tensor, torch.Tensor, torch.Tensor]:
    # ... 前置检查，如果 norm_weight 不为 None，则确保类型和连续性
    if norm_weight is not None:
        assert norm_weight.shape == (hidden_size,)
        if norm_weight.dtype != torch.bfloat16:
            norm_weight = norm_weight.to(torch.bfloat16)
        if not norm_weight.is_contiguous():
            norm_weight = norm_weight.contiguous()

    # ... GEMM 计算部分不变 ...

    # 根据 norm_weight 是否存在派发到不同内核
    if norm_weight is None:
        mhc_pre_big_fuse_tilelang(
            gemm_out_mul, gemm_out_sqrsum, hc_scale, hc_base,
            residual_flat, post_mix, comb_mix, layer_input,
            hidden_size, rms_eps, hc_pre_eps, hc_sinkhorn_eps,
            hc_post_mult_value, sinkhorn_repeat, n_splits, hc_mult,
```

```
        )
    else:
        mhc_pre_big_fuse_with_norm_tilelang(
            gemm_out_mul, gemm_out_sqrsum, hc_scale, hc_base,
            residual_flat, post_mix, comb_mix, layer_input,
            norm_weight, # 传递 norm_weight 到新内核
            hidden_size, rms_eps, hc_pre_eps, hc_sinkhorn_eps,
            hc_post_mult_value, sinkhorn_repeat, norm_eps,
            n_splits, hc_mult,
        )
```

评论区精华

审查中有四个主要讨论点：

- AMD 兼容性：作者 jeejeelee 询问 AMD 开发者 tjtanaa 该内核是否能在 ROCm 上利用，暂无回复。
- norm_eps 默认值：审核者 zyongye 建议将默认值从 0.0 改为 1e-6 以防意外未传值；最终在 mhc_pre_tilelang 中改为 1e-6，但 mhc_fused_post_pre_tilelang 及层接口仍为 0.0（部分采纳）。
- fp32 权重支持：zyongye 询问 TileLang 是否难以支持 fp32 权重（当前代码将 norm_weight 转换为 bfloat16），未得到明确回复。
- 单内核 vs 双内核：zyongye 建议使用编译器常量（如 do_norm）使单一内核通过静态分支区分是否做 norm，以减少双内核维护成本；作者未采纳。
  - AMD ROCm 支持询问 (question)：尚未得到确认，可能需后续兼容。
  - norm_eps 默认值建议 (correctness)：已部分采纳：在 mhc_pre_tilelang 中改为 1e-6，但 mhc_fused_post_pre_tilelang 及层接口仍为 0.0。
  - TileLang 支持 fp32 权重 (design)：未得到明确回复，目前仅支持 bfloat16，未来可考虑扩展。
  - 使用编译器常量替代双内核分支 (design)：作者未回应，当前仍采用两套内核分别处理。

风险与影响

- 风险：
  1. 新内核未充分测试：PR 仅进行了精度验证（GSM8K、AIME25、GPQA Diamond），未包含单元测试或压力测试，可能存在边界条件下的数值问题。
  2. AMD ROCm 未支持：该融合内核仅针对 NVIDIA CUDA，ROCm 路径会回退到旧的 Torch 实现，但未验证功能正确性。
  3. norm_eps 默认值不一致：不同层和 kernel 接口的默认值不统一（0.0 vs 1e-6），可能导致上层调用时遗漏参数产生数值差异。
  4. 双内核维护负担：带 norm 和不带 norm 的两个 tilelang kernel 增加了代码量和后续维护复杂性。 - 影响：对用户：使用 DeepSeek V4 模型（特别是 NVIDIA 平台）将自动受益于 RMSNorm 融合带来的性能提升；对系统：保持了向后兼容性，未提供 norm_weight 时行为与之前相同；对团队：需要维护两条 kernel 路径，未来可考虑合并

为单一内核。 - 风险标记：新内核引入，AMD 兼容性未知，双内核维护负担，norm_eps 默认值不一致，缺少测试覆盖

关联脉络

- PR #42680 [MoE] Migrate W4A8 CT to oracle kernel setup: 该 PR 也重构了 MoE 结构（涉及 NormGateLinear），与本 PR 的 MoE 重构有共同上下文。
- PR #43385 [ROCm] [DSv4] [Perf] Support DeepSeek v4 MTP: 该 PR 为 DeepSeek V4 添加 MTP 支持，同样修改了 model.py 和 mtp.py，同一功能线。