PR #30216 完整报告

sgl-project/sglang

[CPU] add fused_qk_gemma_norm and refactor norm kernel implementation

合并时间：2026-07-07 08:52

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/30216

执行摘要

- 一句话：重构 CPU norm 内核为 trait 框架，新增 fused_qk_gemma_norm
- 推荐动作：值得精读。本 PR 展示了 C++ 模板元编程在 SIMD 内核库中的典型应用，利用 NormTraits 和 NormReduce 实现零开销抽象。对于计划在 sgl-kernel 中添加新 norm 变体的开发者，框架设计模式有直接参考价值，建议重点阅读 NormParams 和 NormReduce 的实现。

功能与动机

CPU norm 内核实现存在大量重复的 reduce 到 scale 逻辑，每个变体独立维护，增加新变体困难。同时为支持 Qwen3.5 模型的推理，需要 fused_qk_gemma_norm 操作，因此进行重构并复用新框架添加该操作。

实现拆解

1. 统一输入布局：在 norm.cpp 中引入 NormParams 结构，自动将 2D/3D/4D 张量映射为逻辑 [B, H, T, D]，通过 input_offset 和 output_offset 统一处理非连续步幅，输出的最后一维保证连续。
2. 定义编译期变体标记：新增 NormMode 枚举区分五种模式（L2Norm、RMSNorm、GemmaNorm、LayerNorm、RMSNormGated），NormTraits 模板通过 has_weight、has_bias、has_shift、has_mean、has_gate 编译期常量控制行为，并特化 AVX512 版本以使用直接内联操作。
3. 向量化 reduce + apply 框架：NormReduceGeneric 实现通用循环（适用于任意精度和尾数处理），NormReduce<M, BFloat16, D> 提供 AVX512 BF16 专用加速路径（仅当 D 为 32/64/128/256/512 时启用）。每个 kernel 只需调用 NormReduce<M, scalar_t, D>::apply，框架自动处理并行、归约和仿射变换。
4. 迁移旧内核：将 l2norm_cpu、rmsnorm_cpu、gemma*_rmsnorm_cpu、layernorm_cpu、fused_add_rmsnorm_cpu、fused_rmsnorm_gated_cpu 等全部依赖新框架，删除原有重复实现。入口函数参数和 torch.ops.sgl_kernel.* 签名完全不变，保证向后兼容。
5. 新增 fused QK Gemma norm：在 torch_extension_cpu.cpp 注册 fused_qk_gemma_rmsnorm_cpu 和 fused_qk_gemma_rmsnorm_with_gate_cpu；在 qwen3_5.py 中，于 CPU 条件分支调用它们完成 Q、K 的归一化融合，避免两次独立的 norm 调用。

6. 测试与工具调整：test_norm.py 改用 pytest 参数化，增加非对齐 hidden_size（33）的边界测试；utils.py 修复 make_non_contiguous 实现，确保非连续输入正确生成而不改变形状。

关键文件：

- sgl-kernel/csrc/cpu/norm.cpp（模块 归一化内核；类别 source；类型 core-logic；符号 NormParams, NormTraits, NormReduce, fused_qk_gemma_rmsnorm_cpu）：核心实现文件：引入 NormParams、NormMode、NormTraits、NormReduce 等模板，将全部 norm kernel 迁移至统一框架，并新增 fused_qk_gemma_norm 逻辑。
- test/registered/cpu/test_norm.py（模块 CPU 测试；类别 test；类型 test-coverage；符号 TestNorm, test_l2norm, test_rmsnorm, test_gemma_rmsnorm）：测试全面重构：改用 pytest 参数化，新增 Gemma4、gated RMSNorm、非对齐尺寸等边界用例，确保内核精度。
- sgl-kernel/csrc/cpu/vec.h（模块 向量化工具；类别 source；类型 core-logic；符号 _mm512_exp_u20_ps）：新增 AVX512 快速 exp 实现 _mm512_exp_u20_ps，用于 RMSNormGated 的 SiLU 门控加速，与原有 fexp_u20 互为补充。
- sgl-kernel/csrc/cpu/torch_extension_cpu.cpp（模块 算子注册；类别 source；类型 core-logic；符号 fused_qk_gemma_rmsnorm_cpu, fused_qk_gemma_rmsnorm_with_gate_cpu）：注册新增的 fused_qk_gemma_rmsnorm 系列算子到 PyTorch 库，使 Python 端可调用。
- python/sglang/srt/models/qwen3_5.py（模块 模型定义；类别 source；类型 data-contract；符号 fused_qk_gemma_rmsnorm, fused_qk_gemma_rmsnorm_with_gate）：在 CPU 条件下启用 fused_qk_gemma_rmsnorm 调用，完成模型 QK norm 融合。
- test/registered/cpu/utils.py（模块 测试工具；类别 test；类型 test-coverage；符号 make_non_contiguous）：修复 make_non_contiguous 函数，确保非连续输入的正确构造，影响所有测试用例的输入质量。

关键符号：fused_qk_gemma_rmsnorm_cpu, fused_qk_gemma_rmsnorm_with_gate_cpu, rmsnorm_cpu, gemma_rmsnorm_cpu, gemma3_rmsnorm_cpu, gemma4_rmsnorm_cpu, l2norm_cpu, layernorm_cpu, fused_add_rmsnorm_cpu, fused_rmsnorm_gated_cpu, NormReduce::apply

关键源码片段

sgl-kernel/csrc/cpu/norm.cpp

核心实现文件：引入 NormParams、NormMode、NormTraits、NormReduce 等模板，将全部 norm kernel 迁移至统一框架，并新增 fused_qk_gemma_norm 逻辑。

```
// sgl-kernel/csrc/cpu/norm.cpp — 核心框架（NormParams + NormMode + NormReduce）

struct NormParams {
  // 统一输入为 [B, H, T, D] 逻辑布局，
  // 2D -> [B, 1, 1, D] ；3D -> [B, 1, T, D] ；4D -> [B, H, T, D]
  // 假设输入最后一维连续，输出同样连续。
```

```
  int64_t B{1}, H{1}, T{1}, D{1};
  int64_t i_strideB{0}, i_strideH{0}, i_strideT{0};
  float eps{1e-5f};
  float shift{0.f};
  const void* weight{nullptr};
  const void* bias{nullptr};

  explicit NormParams(const at::Tensor& input, float eps_)
      : ndim(input.dim()), eps(eps_) {
    TORCH_CHECK(ndim >= 2 && ndim <= 4, "Expected 2D/3D/4D, got ", ndim, "D");
    B = input.size(0);
    D = input.size(ndim - 1);
    i_strideB = input.stride(0);
    // 根据 ndim 解析 H、T 和步幅
    switch (ndim) {
      case 2: break; // [B, D]
      case 3: T = input.size(1); i_strideT = input.stride(1); break; // [B, T, D]
      case 4: H = input.size(1); T = input.size(2);
              i_strideH = input.stride(1); i_strideT = input.stride(2); break;
    }
  }
  inline int64_t rows() const { return B * H * T; }
  inline int64_t input_offset(int64_t b, int64_t h, int64_t t) const {
    return b * i_strideB + h * i_strideH + t * i_strideT;
  }
  inline int64_t output_offset(int64_t b, int64_t h, int64_t t) const {
    return ((b * H + h) * T + t) * D;
  }
};

enum class NormMode {
  L2Norm, // y = x / sqrt(mean(x^2) + eps)
  RMSNorm, // y = x * weight / sqrt(mean(x^2) + eps)
  GemmaNorm, // y = x * (weight + scale_shift) / sqrt(mean(x^2) + eps)
  LayerNorm, // y = (x - mean(x)) * weight / sqrt(var(x) + eps) + bias
  RMSNormGated, // y = x * weight / sqrt(mean(x^2) + eps) * SiLU(gate)
};

// 编译期 trait：通过静态常量控制每个 norm 变体的差异
template <NormMode M> struct NormTraits : NormTraitsBase {};

template <> struct NormTraits<NormMode::RMSNorm> : NormTraitsBase {
  static constexpr bool has_weight = true;
};

template <> struct NormTraits<NormMode::GemmaNorm> : NormTraitsBase {
  static constexpr bool has_weight = true;
  static constexpr bool has_shift = true;
  // apply_shift 在标量和 AVX512 上均有特化
```

```
};

template <> struct NormTraits<NormMode::LayerNorm> : NormTraitsBase {
  static constexpr bool has_weight = true;
  static constexpr bool has_bias = true;
  static constexpr bool has_mean = true;
};

template <> struct NormTraits<NormMode::RMSNormGated> : NormTraitsBase {
  static constexpr bool has_weight = true;
  static constexpr bool has_gate = true;
  // apply_gate 使用 _mm512_exp_u20_ps 快速 sigmoid
};

template <NormMode M, typename scalar_t, int D> struct NormReduce;
// 通用路径（NormReduceGeneric）与 AVX512 BF16 特化路径
// 通过 traits 自动选择 reduce + scale + apply 逻辑
```

test/registered/cpu/test_norm.py

测试全面重构：改用 pytest 参数化，新增 Gemma4、gated RMSNorm、非对齐尺寸等边界用例，确保内核精度。

```
# test/registered/cpu/test_norm.py — 以 test_l2norm 为例的 pytest 参数化测试

import pytest
import torch
from utils import make_non_contiguous, precision

DTYPES = [torch.float16, torch.bfloat16]
DTYPE_IDS = ["float16", "bfloat16"]
eps = 1e-6

class TestNorm:

    def _forward_native(self, x, weight, variance_epsilon=eps, residual=None):
        # 原生 PyTorch 参考实现
        orig_dtype = x.dtype
        x = x.to(torch.float32)
        if residual is not None:
            x = x + residual.to(torch.float32)
            residual = x.to(orig_dtype)
        variance = x.pow(2).mean(dim=-1, keepdim=True)
        x = x * torch.rsqrt(variance + variance_epsilon)
        x = x.to(orig_dtype) * weight
        return x if residual is None else (x, residual)

    @pytest.mark.parametrize("dtype", DTYPES, ids=DTYPE_IDS)
    @pytest.mark.parametrize("hidden_size", [2048, 512])
    @pytest.mark.parametrize("batch_size", [32, 121])
```

```
    def test_l2norm(self, batch_size, hidden_size, dtype):
        # L2Norm 等价于 weight=ones 的 RMSNorm
        x = torch.randn([batch_size, hidden_size], dtype=dtype)
        fake_ones_weight = torch.ones(hidden_size, dtype=dtype)
        out = torch.ops.sgl_kernel.l2norm_cpu(x, eps)
        ref_out = self._forward_native(x, fake_ones_weight, eps)
        atol = rtol = precision[ref_out.dtype]
        torch.testing.assert_close(ref_out, out, atol=atol, rtol=rtol)
```

sgl-kernel/csrc/cpu/vec.h

新增 AVX512 快速 exp 实现 _mm512_exp_u20_ps，用于 RMSNormGated 的 SiLU 门控加速，与原有 fexp_u20 互为补充。

```
// sgl-kernel/csrc/cpu/vec.h — 新增 _mm512_exp_u20_ps（对标 Aten exp_u20）

// 快速指数近似，使用 5 阶泰勒多项式 + 2 的幂缩放
inline __attribute__((always_inline)) __m512 _mm512_exp_u20_ps(const __m512 values) {
  const __m512 vec_factorial_1 = _mm512_set1_ps(0.999999701f);
  const __m512 vec_factorial_2 = _mm512_set1_ps(0.499991506f);
  const __m512 vec_factorial_3 = _mm512_set1_ps(0.166676521f);
  const __m512 vec_factorial_4 = _mm512_set1_ps(0.0418978221f);
  const __m512 vec_factorial_5 = _mm512_set1_ps(0.00828929059f);
  const __m512 vec_exp_log2ef = _mm512_castsi512_ps(_mm512_set1_epi32(0x3fb8aa3b)); //
  log2(e)
  const __m512 vec_half = _mm512_set1_ps(0.5f);
  const __m512 vec_one = _mm512_set1_ps(1.f);
  const __m512 vec_two = _mm512_set1_ps(2.f);
  const __m512 vec_ln2f = _mm512_castsi512_ps(_mm512_set1_epi32(0x3f317218));
  const __m512 vec_ln_flt_min = _mm512_castsi512_ps(_mm512_set1_epi32(0xc2aeac50));
  const __m512 vec_ln_flt_max = _mm512_castsi512_ps(_mm512_set1_epi32(0x42b17218));
  const __m512i vec_127 = _mm512_set1_epi32(0x0000007f);
  const int n_mantissa_bits = 23;

  // 将输入钳位到可表示范围
  auto less_ln_flt_min_mask = _mm512_cmp_ps_mask(values, vec_ln_flt_min, 1);
  auto vec_src = _mm512_min_ps(values, vec_ln_flt_max);
  vec_src = _mm512_max_ps(vec_src, vec_ln_flt_min);

  // 分解为整数部分和分数部分：fx = floor(x * log2(e) + 0.5)
  auto vec_fx = _mm512_fmadd_ps(vec_src, vec_exp_log2ef, vec_half);
  auto vec_fx_i = _mm512_cvt_roundps_epi32(vec_fx, _MM_FROUND_TO_NEG_INF | _MM_
  FROUND_NO_EXC);
  vec_fx = _mm512_cvtepi32_ps(vec_fx_i);
  auto vec_exp_poly = _mm512_fnmadd_ps(vec_fx, vec_ln2f, vec_src); // x - fx * ln2

  // 用 Horner 法计算 exp(frac) 多项式
  auto vec_res = _mm512_fmadd_ps(vec_exp_poly, vec_factorial_5, vec_factorial_4);
  vec_res = _mm512_fmadd_ps(vec_exp_poly, vec_res, vec_factorial_3);
  vec_res = _mm512_fmadd_ps(vec_exp_poly, vec_res, vec_factorial_2);
```

```
  vec_res = _mm512_fmadd_ps(vec_exp_poly, vec_res, vec_factorial_1);
  vec_res = _mm512_fmadd_ps(vec_exp_poly, vec_res, vec_one);

  // 乘以 2^n
  auto vec_exp_number = _mm512_sub_ps(vec_fx, vec_one);
  auto vec_two_pow_n_i = _mm512_add_epi32(_mm512_cvtps_epi32(vec_exp_number), vec_127);
  vec_two_pow_n_i = _mm512_slli_epi32(vec_two_pow_n_i, n_mantissa_bits);
  auto vec_two_pow_n = _mm512_mask_blend_ps(less_ln_flt_min_mask,
      _mm512_castsi512_ps(vec_two_pow_n_i), _mm512_set1_ps(0.f));
  vec_res = _mm512_mul_ps(vec_res, vec_two_pow_n);
  vec_res = _mm512_mul_ps(vec_res, vec_two);
  return vec_res;
}
```

评论区精华

无 review 讨论（PR 由作者自行合并）。

- 暂无高价值评论线程

风险与影响

- 风险：重构涉及整个 norm 核心逻辑，存在精度回归风险。测试覆盖了多种变体（包括非对齐尺寸 33）并与原生 PyTorch 输出比较，CI 可检测。新增的快速 exp 实现 _mm512_exp_u20_ps（基于 Aten exp_u20 映射）与原有 _mm512_fexp_u20_ps 并存，需确保两者对应同一计算语义以避免 SiLU 门控精度偏差。AVX512 BF16 专用路径仅在特定维度激活，若未来硬件不支持，应自动降级到通用路径（当前未显式检测，需依赖编译期宏）。
- 影响：对 CPU 后端的 norm 操作进行全面重构，公开 API 不变，用户无需修改代码。性能在支持 AVX512 的 BF16 场景下提升显著（>10×）。新增的 fused_qk_gemma_norm 使得 Qwen3.5 模型在 CPU 上能够高效执行 QK norm 融合，减少显式 kernel launch 开销。测试代码迁移至 pytest，与标准 CI 工具链更兼容。
- 风险标记：核心路径变更，模板复杂性，AVX512 精度

关联脉络

- 暂无明显关联 PR