PR #31171 完整报告

sgl-project/sglang

[CPU] add fused input proj for qwen3.5

合并时间：2026-07-15 15:06

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/31171

执行摘要

- 一句话：为 Qwen3.5 CPU 添加融合输入投影内核
- 推荐动作：推荐阅读。该 PR 展示了如何在 SGLang 中为特定 CPU 硬件（Intel AMX）高效实现内核融合的完整流程：从 C++ 内核实现、Torch 扩展注册、torch.compile 兼容、到模型集成和测试。LazyValue 条件启用模式值得借鉴。对于关注 CPU 推理性能的开发者有直接参考价值。

功能与动机

主要动机是优化 Qwen3.5 在 CPU（特别是 Intel AMX）上的推理性能，通过将两个相邻的线性投影融合到一个内核中，提高计算效率。PR 描述原文：'Adds fused_input_proj_cpu to compute in_proj_qkvz and in_proj_ba in one CPU kernel.'

实现拆解

实现拆解

1. 实现融合内核：在 sgl-kernel/csrc/cpu/model/qwen3.cpp 中新增 fused_input_proj_kernel_impl 模板函数，使用 block_size_m/block_size_n 分块和 tinygemm_kernel 执行 GEMM，parallel_2d 并行，通过 is_first 标志区分输出到 out（qkvz）和 out2（ba），在一个内核中同时计算两个投影。同时简化 fused_qkvzba_split_reshape_cat_contiguous_impl，移除未使用的 k_tp 参数，并用 CHECK_INPUT_SHAPE_DTYPE<false> 宏替代分散的 CHECK 宏。
2. 注册 Torch 扩展 op：在 sgl-kernel/csrc/cpu/torch_extension_cpu.cpp 中声明 fused_input_proj_cpu 函数并注册到 TORCH_LIBRARY_FRAGMENT(sgl_kernel, m)，声明签名接受 hidden_states、qkvz_weight、ba_weight 和 is_vnni 标志，返回两个 Tensor。
3. 添加 torch.compile fake 注册：在 python/sglang/srt/model_executor/cpu_graph_runner.py 中通过 @register_cpu_compile_fake("fused_input_proj_cpu") 注册 fake 实现，使该 op 在 torch.compile 图形捕获阶段可用，避免编译错误。
4. 集成到 Qwen3.5 模型：在 python/sglang/srt/models/qwen3_5.py 中，于 Qwen3_5GatedDeltaNet.__init__ 内添加 _fused_input_proj_cpu_enabled LazyValue，条件判定为 _is_cpu、bf16 dtype、无 bias、且 AMX 后端支持。在 _forward_input_proj 方法中，当条件满足时调用融合 op，否则走原有两条单独线性投影。在 forward 方法中，统一 CPU 和 GPU 路径，移除之前 CPU 独有分支，直接使用

fused_qkvzba_split_reshape_cat_contiguous（在 CPU 上映射为 fused_qkvzba_split_reshape_cat_contiguous_cpu）。

5. 更新测试：在 test/registered/cpu/test_qwen3.py 中将类式 unittest 测试改为函数式 pytest 测试，添加 test_fused_input_proj 函数，使用随机 bf16 张量验证融合 op 输出与两次独立线性投影一致。跳过 ARM64 环境（AMX 不可用）。

关键文件：

- sgl-kernel/csrc/cpu/model/qwen3.cpp（模块 内核实现；类别 source；类型 core-logic；符号 fused_input_proj_kernel_impl, fused_input_proj_cpu, fused_qkvzba_split_reshape_cat_contiguous_impl）：核心变更：新增融合输入投影内核实现，简化现有接口。
- test/registered/cpu/test_qwen3.py（模块 测试；类别 test；类型 test-coverage；符号 test_fused_input_proj, test_fused_qkvzba_split_reshape_cat, test_fused_qkvzba_split_reshape_cat_contiguous）：测试配套：添加融合内核的 pytest 测试，并将现有测试迁移至 pytest 风格。
- python/sglang/srt/models/qwen3_5.py（模块 模型集成；类别 source；类型 core-logic；符号 _fused_input_proj_cpu_enabled, _forward_input_proj, forward）：模型集成：通过 LazyValue 启用融合路径，简化 forward 中 CPU 分支。
- python/sglang/srt/model_executor/cpu_graph_runner.py（模块 编译支持；类别 source；类型 data-contract；符号 _）：为融合 op 添加 torch.compile fake 注册，使图形捕获正常工作。
- sgl-kernel/csrc/cpu/torch_extension_cpu.cpp（模块 扩展注册；类别 source；类型 configuration）：注册 fused_input_proj_cpu op 到 Torch 扩展库。

关键符号：fused_input_proj_kernel_impl, fused_input_proj_cpu, fused_qkvzba_split_reshape_cat_contiguous_impl, fused_qkvzba_split_reshape_cat_cpu, test_fused_input_proj, test_fused_qkvzba_split_reshape_cat, test_fused_qkvzba_split_reshape_cat_contiguous, Qwen3_5GatedDeltaNet._forward_input_proj, Qwen3_5GatedDeltaNet.forward

关键源码片段

sgl-kernel/csrc/cpu/model/qwen3.cpp

核心变更：新增融合输入投影内核实现，简化现有接口。

```
/*
 * 融合输入投影内核：同时计算 out = input @ weight.T 和 out2 = input @ weight2.T
 * 将两个 GEMM 合并到一个并行循环中，通过 nb_start 与 N 的关系判断当前块属于 qkvz 还是 ba 输出。
 * 使用 BLOCK_M x BLOCK_N 分块，支持 brgemm 优化，对 Ctmp 使用 float32 累加以提高精度。
 */
template <typename scalar_t>
void fused_input_proj_kernel_impl(
    scalar_t* __restrict__ out,
    scalar_t* __restrict__ out2,
```

```
    const scalar_t* __restrict__ input,
    const scalar_t* __restrict__ weight,
    const scalar_t* __restrict__ weight2,
    int64_t M, int64_t N, int64_t N2, int64_t K) {
  constexpr int64_t BLOCK_M = block_size_m();
  constexpr int64_t BLOCK_N = block_size_n();
  const int64_t MB = div_up(M, BLOCK_M);
  const int64_t NB = div_up(N + N2, BLOCK_N);
  const bool use_brgemm = can_use_brgemm<scalar_t>(M);
  parallel_2d(MB, NB, [&](int64_t mb0, int64_t mb1, int64_t nb0, int64_t nb1) {
    alignas(64) float Ctmp[BLOCK_M * BLOCK_N];
    loop_2d<scalar_t>(mb0, mb1, nb0, nb1, BLOCK_N * K, [&](int64_t mb, int64_t nb, int64_t) {
      int64_t mb_start = mb * BLOCK_M;
      int64_t mb_size = std::min(M - mb_start, BLOCK_M);
      int64_t nb_start = nb * BLOCK_N;
      const bool is_first = nb_start < N;
      int64_t local_nb_start = is_first ? nb_start : nb_start - N;
      int64_t nb_size = std::min((is_first ? N : N2) - local_nb_start, BLOCK_N);
      scalar_t* __restrict__ curr_out = is_first ? out : out2;
      const scalar_t* __restrict__ curr_weight = is_first ? weight : weight2;
      int64_t local_out_strideM = is_first ? N : N2;
      tinygemm_kernel<scalar_t>(input + mb_start * K,
          curr_weight + local_nb_start * K,
          curr_out + mb_start * local_out_strideM + local_nb_start,
          Ctmp, mb_size, nb_size, K, K, nb_size, local_out_strideM, use_brgemm);
    });
    if (use_brgemm) { at::native::cpublas::brgemm_release(); }
  });
}
```

python/sglang/srt/models/qwen3_5.py

模型集成：通过 LazyValue 启用融合路径，简化 forward 中 CPU 分支。

```
# 延迟评估的启用条件，仅在 CPU/bf16/no bias/AMX 支持时融合
self._fused_input_proj_cpu_enabled = LazyValue(
    lambda: (
        _is_cpu
        and self.in_proj_qkvz.weight.dtype == torch.bfloat16
        and self.in_proj_ba.weight.dtype == torch.bfloat16
        and self.in_proj_qkvz.bias is None
        and self.in_proj_ba.bias is None
        and use_intel_amx_backend(self.in_proj_qkvz)
        and use_intel_amx_backend(self.in_proj_ba)
    )
)

# 在 _forward_input_proj 中使用
elif self._fused_input_proj_cpu_enabled.value:
    projected_states_qkvz, projected_states_ba = (
```

```
        torch.ops.sgl_kernel.fused_input_proj_cpu(
            hidden_states,
            self.in_proj_qkvz.weight,
            self.in_proj_ba.weight,
            True,
        )
    )
```

评论区精华

Review 中共有 3 条来自 gemini-code-assist[bot] 的评论，均为改进建议：

- 整除性检查（高优先级）：建议在启用条件中增加 TP 下维度是否被 64 整除的检查，不满足时 fallback。作者回复称 CPU TP padding 已确保维度是 32 的倍数，故未采纳。
- 向上取整除（高优先级）：建议在 CPU 路径中使用向上取整除法 (x + y - 1) // y 代替整数除法，避免 TP 数大于头数时出现零值。该评论未得到作者回复，但最终 PR 仍使用整数除法，可能因为 CPU TP 场景下不会出现该问题。
- Tensor 参数传递约定（中优先级）：建议将 C++ 扩展中 tensor 参数从非 const 引用改为值传递，以兼容临时对象。该评论未得到回复，代码仍使用引用传递（已合并）。

整体上讨论集中于代码健壮性和风格约定，无重大设计争议。

- TP 下整除性检查 (correctness): 作者回复 CPU TP padding 确保 32 对齐，未采纳。
- CPU 头数计算使用向上取整 (correctness): 未得到作者回复，PR 仍使用整数除法。
- Tensor 参数传递方式 (style): 未得到回复，保持引用传递。

风险与影响

- 风险：技术风险：

1. 融合内核启用条件严格：仅在 _is_cpu、bf16、no bias、AMX 支持 同时满足时启用，其他情况 fallback 到独立线性投影，因此引入回归概率低。但若 TP 下维度 padding 不足，仍可能因内核中 TORCH_CHECK 导致崩溃（作者声称 padding 确保 32 对齐）。
2. C++ 扩展参数传递方式：fused_input_proj_cpu 的 tensor 参数使用非 const 引用，可能阻止某些调用情景（如传递临时对象），但当前仅从 Python 触发，冲突风险低。
3. 整数除法潜在问题：CPU 路径中使用 // 计算 num_k_heads_tp，若 attn_tp_size > num_k_heads 将得零，下游内核可能因尺寸异常崩溃。但 CPU TP 场景下 attn_tp_size 通常不会超过头数，风险可控。
4. 测试覆盖：新增的融合内核测试覆盖了 batch=7 的随机形状，但未覆盖 TP 场景或大 batch。总体风险较低。 - 影响：影响分析：

- 用户影响：使用 Qwen3.5 模型在 Intel CPU（支持 AMX）上的推理性能将提升，因为两个线性投影合并在一个内核中执行，减少内存带宽和内核启动开销。其他模型或硬件不受影响。
- 系统影响：新增的 fused_input_proj_cpu op 依赖 sgl-kernel CPU 库，编译时需支持 AMX 指令集。启用条件包括 use_intel_amx_backend，需要运行时检测。
- 团队影响：简化了 CPU Qwen3 内核接口，移除了冗余参数，方便后续维护。将测试迁移到 pytest 风格，提高一致性。

- 风险标记：内核条件启用风险低，整数除法潜在问题，C++ 引用风格风险低

关联脉络

- 暂无明显关联 PR