PR #30604 完整报告

sgl-project/sglang

[CPU] update fla.cpp to support when num_head_v is not multiples of 16

合并时间：2026-07-10 09:21

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/30604

执行摘要

- 一句话：解除 CPU FLA 内核中 num_head_v 必须是 16 倍数的限制
- 推荐动作：建议阅读 fla.cpp 中 cumsum_kernel 的 AVX512 实现，了解如何通过 mask 和向上取整消除硬件向量宽度对齐限制。测试文件展示了 pytest 参数化的最佳实践。整体变更安全、清晰，值得参考。

功能与动机

Qwen3.5 models when doing TP with 3, 4, 6 may end up with having num_head_v not 16x — PR body

实现拆解

1. 修改 cumsum_kernel 接口（fla.cpp）

- 增加 hb_size 参数，表示当前头部块实际包含的头数（非 16 倍数时最后一块可能不足 16）。
- 移除 size 参数（原为整个序列长度），替换为 mb_size（当前 chunk 内的 token 数）和 hb_size。

2. 适配 AVX512 实现（fla.cpp）

- 在 cumsum_kernel<float> 特化中，使用 _mm512_maskz_loadu_ps(vmask, ...) 只加载有效的 hb_size 个 float，避免读取越界内存。
- 存储时增加 if (j < hb_size) 判断，仅写入实际的头部列，跳过剩余位置。

3. 调整 chunk_local_cumsum_kernel_impl（fla.cpp）

- 将 HB = Hv / BLOCK_H 改为 div_up(Hv, BLOCK_H)，确保当 Hv 不是 16 倍数时能覆盖所有头部。
- 在循环中计算 hb_size = min(Hv - hb*BLOCK_H, BLOCK_H)，传递给 cumsum_kernel。
- 移除原 TORCH_CHECK(Hv % BLOCK_H == 0) 断言。

4. 重构测试（test_mamba.py）

- 从 unittest.TestCase 类风格改为独立 pytest 参数化函数，使用 @pytest.mark.parametrize 注入多组参数。
- 新增 (B=1, T=128, HK=3, HV=6, K=128, V=128) 组合，非 16 倍数的典型场景。

- 移除对 CustomTestCase 的依赖，直接使用 torch.testing.assert_close 进行精度比较。

关键文件：

- sgl-kernel/csrc/cpu/mamba/fla.cpp（模块 CPU 算子；类别 source；类型 core-logic；符号 cumsum_kernel, chunk_local_cumsum_kernel_impl）：核心 CPU 内核文件，解除 num_head_v 必须是 16 倍数的限制，引入 mask load 和部分头部写入，是实现兼容性的关键改动。
- test/registered/cpu/test_mamba.py（模块 测试层；类别 test；类型 test-coverage；符号 TestMambaAttention, test_chunk_gated_delta_rule, test_fused_gdn_gating, test_fused_sigmoid_gating_delta_rule_update）：测试文件，从 unittest 重构为 pytest 参数化，新增非 16 倍数参数组合，验证内核兼容性，保障回归覆盖。

关键符号：cumsum_kernel::apply, chunk_local_cumsum_kernel_impl, test_chunk_gated_delta_rule

关键源码片段

sgl-kernel/csrc/cpu/mamba/fla.cpp

核心 CPU 内核文件，解除 num_head_v 必须是 16 倍数的限制，引入 mask load 和部分头部写入，是实现兼容性的关键改动。

```
// cumsum_kernel<float> 特化：支持非 16 倍数头部数
// hb_size: 实际处理的头部数量 (1..BLOCK_H)
// mb_size: 当前 chunk 内的 token 数
template <int CHUNK_SIZE, int BLOCK_H>
struct cumsum_kernel<float, CHUNK_SIZE, BLOCK_H> {
  static inline void apply(
      float* __restrict__ out,
      const float* __restrict__ input,
      int mb_size,
      int hb_size,
      int ld_src,
      int ld_dst) {
    // BLOCK_H 固定为 16（AVX512 vector width）
    static_assert(BLOCK_H == 16);
    // hb_size 必须为正且不超过 BLOCK_H
    TORCH_CHECK(hb_size > 0 && hb_size <= BLOCK_H);
    // 生成头部掩码：仅低 hb_size 位为 1
    const __mmask16 vmask = static_cast<__mmask16>((1u << hb_size) - 1u);

    __m512i va[16];
    __m512 vsum = _mm512_set1_ps(0.f);

    for (int i = 0; i < CHUNK_SIZE; i += 16) {
      Unroll<16>{}([&](auto j) {
        __m512 v;
        if (i + j < mb_size) {
          // 使用掩码加载，只读取有效的 hb_size 个 float
```

```
          v = _mm512_maskz_loadu_ps(vmask, input + (i + j) * ld_src);
        } else {
          v = _mm512_setzero_ps();
        }
        vsum = _mm512_add_ps(vsum, v);
        va[j] = _mm512_castps_si512(vsum);
      });
      // 转置后存储
      transpose_16x16_32bit(va);
      Unroll<16>{}([&](auto j) {
        if (j < hb_size) {
          _mm512_storeu_si512(out + j * ld_dst + i, va[j]);
        }
      });
    }
  }
};

// chunk_local_cumsum_kernel_impl 中调用处的关键变化
int64_t HB = div_up(Hv, int64_t(BLOCK_H)); // 向上取整，确保覆盖所有头部
// 循环内计算当前块的 hb_size
int64_t hb_size = std::min(Hv - hb * BLOCK_H, int64_t(BLOCK_H));
// 传递给 cumsum_kernel
cumsum_kernel<scalar_t, CHUNK_SIZE, BLOCK_H>::apply(
    gsum_ptr, g_ptr, mb_size, hb_size, Hv, CHUNK_SIZE);
```

test/registered/cpu/test_mamba.py

测试文件，从 unittest 重构为 pytest 参数化，新增非 16 倍数参数组合，验证内核兼容性，保障回归覆盖。

```
# 参数化测试：覆盖 16 倍数和非 16 倍数两种头部配置 @pytest.mark.parametrize(
("B", "T_PER_SEQ", "HK", "HV", "K", "V", "POOL_SIZE"),    [        # 第一个参数组合：
HK=3, HV=6（非 16 倍数，模拟 TP 场景）          (1, 128, 3, 6, 128, 128, 17),        # 第
二个参数组合：HK=16, HV=32（标准的 16 倍数，保持回归）          (1, 128, 16, 32, 128,
128, 17),    ], ) deftest_chunk_gated_delta_rule(B, T_PER_SEQ, HK, HV, K, V,
POOL_SIZE):    # 初始化输入张量，HK 和 HV 由参数传入    query_ = torch.randn((B, T,
HK, K), dtype=torch.bfloat16)    key_ = torch.randn((B, T, HK, K),
dtype=torch.bfloat16)    value_ = torch.randn((B, T, HV, V), dtype=torch.bfloat16)    # .
.. 其余逻辑保持不变    # 调用 CPU 内核    core_attn_out, returned_state =
torch.ops.sgl_kernel.chunk_gated_delta_rule_cpu(        query=query, key=key,
value=value, g=g, beta=beta,        initial_state=initial_state, output_final_state=True,
    cu_seqlens=cu_seqlens, head_first=False,
use_qk_l2norm_in_kernel=use_qk_l2norm_in_kernel,
initial_state_indices=cache_indices,    )    # 与参考实现比较精度
torch.testing.assert_close(core_attn_out, core_attn_out_ref, ...)  （其余测试函数同样改为
参数化形式，略。）
```

评论区精华

此 PR 无实质性 review 讨论，仅由作者自行合并。PR 评论中出现一个 bot 的 quota 警告，与功能无关。

- 暂无高价值评论线程

风险与影响

- 风险：1. 性能风险：_mm512_maskz_loadu_ps 和条件存储可能比全宽加载略有开销，但仅在最后一个不足 16 头的块触发，且 AVX512 mask 指令通常只有 1 个额外周期，整体影响极小。 2. 越界风险：div_up 和 hb_size 计算确保索引不越界，但需确认所有调用路径均传递正确的 hb_size。当前仅在 chunk_local_cumsum_kernel_impl 使用，无其他调用点。 3. 回归风险：测试新增了非 16 倍数参数组合，并可复用原有 16 倍数组合，通过 CI 确保回归覆盖。但测试未覆盖所有可能的分支（如 use_qk_l2norm_in_kernel=False）。
- 影响：用户侧：CPU 推理用户现在可以运行 Qwen3.5 等使用非标准头部数（如 3、4、6）的模型，拓宽了兼容性。原有 16 倍数头部数的模型不受影响。 系统侧：无明显影响。CPU 内核的改动仅影响 chunk_local_cumsum_kernel_impl 路径，其他路径（如 decay_mask_kernel、l2norm_kernel）未改动。 团队侧：测试框架从 unittest 迁移到 pytest 参数化，统一了测试风格，便于后续添加新参数组合。
- 风险标记：mask 加载性能损失，向上取整边界越界

关联脉络

- 暂无明显关联 PR