PR #29117 完整报告

sgl-project/sglang

[CPU] optimize GDN prefill performance

合并时间：2026-06-25 09:04

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29117

执行摘要

该 PR 显著优化了 CPU 上 GDN（Gated Delta Network）的 prefill 性能，通过融合计算阶段和利用 AVX512/AMX 指令，实现约 1.5 倍加速。同时将状态格局从 KV 改为 VK 以对齐现有 Triton 实现，并更新了测试用例。

功能与动机

原 CPU 内核存在大量中间张量物化和冗余计算，性能未能匹配 Triton 融合流水线。该 PR 旨在缩小差距，提升 CPU 推理吞吐量："Optimize the CPU GDN (Gated Delta Network) kernels in fla.cpp to better match the fused Triton pipeline and improve prefill/decode performance on AVX512/AMX CPUs."

实现拆解

1. 算法融合：在 chunk_gated_delta_rule_kernel_impl 中将 intra-chunk 的 KKT 求解、下三角求解和 w/u 重计算融合，避免物化矩阵 A；将 inter-chunk 的状态更新和输出计算融合，避免对中间变量 h 和 v_new 的显式存储。
2. AVX512 核心：编写专用 AVX512 内核处理 decay mask、tril solve 和 beta/g 掩码，减少循环开销。
3. VNNI2 打包与状态布局对齐：新增 pack_vnni2 函数完成 FP32 到 BF16 的 VNNI2 格式转换并融合 element-wise 缩放；将状态布局从 KV 改为 VK 以匹配 Triton 实现，允许使用 amx-bf16 矩阵乘更新状态。
4. 测试适配：在 test/registered/cpu/test_mamba.py 中更新参考函数（如 chunk_gated_delta_rule_update 和 sigmoid_gating_delta_rule_update），加入必要的状态转置，确保精度测试通过。

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

核心优化实现，大幅重写 GDN 内核

```
// 将矩阵从 [K/2, 2, N] FP32 转换为 [K/2, N, 2] BF16,
// 同时将 src 乘以 exp(g_last) 以融合 element-wise 缩放。
template <typename scalar_t, int K, int N>
void pack_vnni2(scalar_t* __restrict__ dst, float* __restrict__ src,
                const float g_last, int ld_src, int ld_dst) {
  static_assert(K % 32 == 0);
  static_assert(N % 32 == 0);
  const float scale = std::exp(g_last);
```

```
#if defined(CPU_CAPABILITY_AVX512)
  constexpr int KB = K / 2;
  constexpr int NB = N / 32;
  __m512i s0, s1, d0, d1;
  __m512 vd = _mm512_set1_ps(scale);
  const auto trans = [&](auto i) {
    constexpr int kb = i / NB;
    constexpr int nb = i % NB;
    constexpr int k0 = kb * 2 + 0;
    constexpr int k1 = kb * 2 + 1;
    // 加载 k0 行和 k1 行的各一半（32 个元素每份）
    __m512 v00 = _mm512_loadu_ps(src + k0 * ld_src + nb * 32);
    __m512 v01 = _mm512_loadu_ps(src + k0 * ld_src + nb * 32 + 16);
    __m512 v10 = _mm512_loadu_ps(src + k1 * ld_src + nb * 32);
    __m512 v11 = _mm512_loadu_ps(src + k1 * ld_src + nb * 32 + 16);
    // FP32 -> BF16 并交错打包
    s0 = (__m512i)_mm512_cvtne2ps_pbh(v01, v00);
    s1 = (__m512i)_mm512_cvtne2ps_pbh(v11, v10);
    // 转置为 [K/2, N/32, 32, 2] 布局
    std::tie(d0, d1) = transpose_2x32_16bit(s0, s1);
    _mm512_storeu_si512(dst + kb * ld_dst * 2 + nb * 32 * 2, d0);
    _mm512_storeu_si512(dst + kb * ld_dst * 2 + nb * 32 * 2 + 32, d1);
    // 在原 src 上更新乘以 exp(g_last) 以用于后续计算
    _mm512_storeu_ps(src + k0 * ld_src + nb * 32, _mm512_mul_ps(v00, vd));
    _mm512_storeu_ps(src + k0 * ld_src + nb * 32 + 16, _mm512_mul_ps(v01, vd));
    _mm512_storeu_ps(src + k1 * ld_src + nb * 32, _mm512_mul_ps(v10, vd));
    _mm512_storeu_ps(src + k1 * ld_src + nb * 32 + 16, _mm512_mul_ps(v11, vd));
  };
  Unroll<KB * NB>{}(trans);
#else
  // 通用路径：逐元素转换与缩放
  for (int k = 0; k < K; k += 2) {
    for (int n = 0; n < N; ++n) {
      const float v0 = src[(k + 0) * ld_src + n];
      const float v1 = src[(k + 1) * ld_src + n];
      dst[(k >> 1) * ld_dst * 2 + n * 2 + 0] = static_cast<scalar_t>(v0);
      dst[(k >> 1) * ld_dst * 2 + n * 2 + 1] = static_cast<scalar_t>(v1);
      src[(k + 0) * ld_src + n] = v0 * scale;
      src[(k + 1) * ld_src + n] = v1 * scale;
    }
  }
#endif
}
```

sgl-kernel/csrc/cpu/vec.h

新增 16 位矩阵转置函数支持 VNNI2 packing

```
// 使用 AVX512 对 16x16 的 16 位矩阵进行原位转置（输入 / 输出均为 __m256i 数组）。
inline void transpose_16x16_16bit(__m256i* v) {
```

```
    __m256i v1[16];
    // 第一层：交错 16 位通道
    v1[0] = _mm256_unpacklo_epi16(v[0], v[1]);
    v1[1] = _mm256_unpackhi_epi16(v[0], v[1]);
    v1[2] = _mm256_unpacklo_epi16(v[2], v[3]);
    v1[3] = _mm256_unpackhi_epi16(v[2], v[3]);
    // ... 重复到 v[14], v[15]
    //（完整实现见源文件）
    // 第二层：交错 32 位通道
    v[0] = _mm256_unpacklo_epi32(v1[0], v1[2]);
    // ...
    // 最终通过 _mm256_permute2x128_si256 交换 128 位通道得到转置结果
}
```

评论区精华

无 review 讨论。

风险与影响

- 状态布局变更：从 KV 改为 VK 需要确保所有调用方知晓，测试已覆盖。
- AVX512 依赖：优化路径依赖特定指令集，非 AVX512 CPU 性能提升有限。
- 精度风险：融合计算可能引入数值差异，需持续监控。
- 影响范围：仅影响 CPU 上 GDN 注意力计算，GPU 和其他模型不受影响。

关联脉络

该 PR 是 sgl-kernel 中对 CPU 计算优化的持续工作，与之前针对 NPU 的 FA3 适配（#26724）及 DeepSeek 精度修复（#29042）均属于硬件后端优化系列。后续可能进一步统一 CPU 与 GPU 的 kernel 接口。