PR #29382 完整报告

sgl-project/sglang

[CPU] use faster exp in silu_and_mul

合并时间：2026-06-29 09:38

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29382

执行摘要

该 PR 将 CPU 端 silu_and_mul 激活函数的向量化实现中的 exp 替换为 exp_u20 近似，实现 6% 至 29% 的性能提升。变更仅涉及一行代码，风险低，收益明确。

功能与动机

silu_and_mul 是 LLM gated MLP 中频繁使用的激活函数。在 CPU 推理中，其向量化路径使用 std::exp（基于 sleef u10 实现）计算指数部分。PyTorch 的 Vectorized<float> 提供了 exp_u20 近似函数，精度为 20 ULP 但速度更快。该 PR 旨在利用这一近似加速 CPU 推理，同时保持数值结果在可接受范围内。

实现拆解

1. 定位变更点：文件 sgl-kernel/csrc/cpu/activation.cpp 中 silu_and_mul_cpu 函数的向量化 lambda。
2. 替换指数函数：将 x.neg().exp() 改为 x.neg().exp_u20()，标量路径的 std::exp(-x) 保持不变以保留精度回退。
3. 测试验证：执行 test/registered/cpu/test_activation.py 通过，并提供了详细的精度对比和性能基准（见 PR body 表格）。

sgl-kernel/csrc/cpu/activation.cpp

核心变更文件，替换 silu_and_mul 向量化路径中的 exp 为 exp_u20

```
// 文件：sgl-kernel/csrc/cpu/activation.cpp
// 改动：将向量化版本中的 .exp() 替换为 .exp_u20()，
// exp_u20 是 PyTorch 提供的更快近似，精度约 20 ULP，
// 标量回退路径仍使用 std::exp 以保证数值稳定性。
at::Tensor silu_and_mul_cpu(at::Tensor& input) {
  auto sizes = input.sizes().vec();
  int64_t last_dim = input.ndimension() - 1;
  int64_t d = sizes[last_dim] / 2;
  sizes[last_dim] = d;
  int64_t num_tokens = input.numel() / input.size(-1);
  at::Tensor out = at::empty(sizes, input.options());

  AT_DISPATCH_REDUCED_FLOATING_TYPES(input.scalar_type(), "silu_and_mul", [&] {
    using Vec = at::vec::Vectorized<float>;
    act_and_mul_kernel_impl(
```

```
        out.data_ptr<scalar_t>(),
        input.data_ptr<scalar_t>(),
        num_tokens,
        d,
        // 标量回退：使用标准 exp，确保精度
        [](float x) { return x / (1.f + std::exp(-x)); },
        // 向量化路径：使用 exp_u20 近似，加速推理
        [](Vec x) { return x / (Vec(1.f) + x.neg().exp_u20()); });
  });
  return out;
}
```

评论区精华

无 review 讨论。

风险与影响

- 精度风险：exp_u20 的误差约 20 ULP，可能在某些极端输入下与 exp 结果略有偏差。但 SiLU 激活函数对指数部分误差不敏感，实际影响可以忽略。
- 回归风险：仅修改一行，且经过测试验证，回归可能性极低。
- 影响：所有使用 CPU silu_and_mul 的模型将受益，加速可达 29%。

关联脉络

该 PR 是 CPU 内核性能优化的延续。类似地，sgl-kernel 中其他激活函数（如 gelu_tanh_and_mul）未来也可采用相同的近似策略以获得额外加速。