PR #26516 完整报告

sgl-project/sglang

Add sliding-window mask support to TorchNativeAttnBackend

合并时间：2026-05-28 16:06

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26516

执行摘要

- 一句话：为 TorchNativeAttnBackend 添加滑动窗口掩码支持
- 推荐动作：该 PR 修复了一个影响正确性的 bug，实现清晰且改动范围小，值得合并。建议关注后续的测试 PR，以确保滑动窗口掩码逻辑在各种情况下（如 prefix caching、PD 分离等）的正确性。

功能与动机

TorchNativeAttnBackend._run_sdpa_forward_extend 和 _run_sdpa_forward_decode 之前忽略了 layer.sliding_window_size，始终传递 is_causal=True 给 scaled_dot_product_attention。需要滑动窗口注意力的模型（例如 Mistral、Gemma）在 torch_native 后端上因此产生了错误的输出——每个查询都关注了完整的 prefix，而不管请求的窗口大小。

实现拆解

1. 新增 _make_sliding_window_mask 静态方法（torch_native_backend.py 第 27-40 行）：该方法根据 q_len、kv_len、sliding_window_size、device 和 query_offset 生成一个布尔掩码张量。掩码中 mask[q, k] 为 True 当且仅当 k >= q + query_offset - sliding_window_size且k<=q+query_offset，即只允许query关注其滑动窗口内的key。
2. 修改 _run_sdpa_forward_extend 方法（第 62 行新增 sliding_window_size 参数，第 133-143 行实现逻辑）：当 sliding_window_size is not None 且 > -1 时，调用 _make_sliding_window_mask 生成掩码，并将 attn_mask 传入 SDPA，同时将 is_causal 设为 False（因为 SDPA 无法同时使用 is_causal=True 和自定义 attn_mask）。
3. 修改 _run_sdpa_forward_decode 方法（第 175 行新增参数，第 234-247 行实现逻辑）：与 extend 类似，但区别在于 decode 时 query 只有最后一个 token，因此 query_offset 设置为 seq_len_kv - seq_len_q 以反映 query 在 KV 序列中的实际位置。
4. 修改 forward_extend 和 forward_decode 方法（第 310-318 行和第 369-376 行）：从 layer.sliding_window_size 获取值，并仅在满足条件时传递（即 causal=True、非 cross-attention 且 sliding_window_size 有效）。非滑动窗口路径保持不变。

关键文件：

- python/sglang/srt/layers/attention/torch_native_backend.py（模块 注意力层；类别 source；类型 core-logic；符号 _make_sliding_window_mask）：核心变更文件，新增

_make_sliding_window_mask 方法并修改 _run_sdpa_forward_extend 和 _run_sdpa_forward_decode 以支持滑动窗口掩码。

关键符号：_make_sliding_window_mask, _run_sdpa_forward_extend, _run_sdpa_forward_decode, forward_extend, forward_decode

关键源码片段

python/sglang/srt/layers/attention/torch_native_backend.py

核心变更文件，新增 _make_sliding_window_mask 方法并修改 _run_sdpa_forward_extend 和 _run_sdpa_forward_decode 以支持滑动窗口掩码。

```
# python/sglang/srt/layers/attention/torch_native_backend.py

    @staticmethod
    def _make_sliding_window_mask(
        *,
        q_len: int,
        kv_len: int,
        sliding_window_size: int,
        device: torch.device,
        query_offset: int = 0,
    ) -> torch.Tensor:
        # 生成滑动窗口布尔掩码
        # 对于 query 位置 q（绝对位置 = query_offset + q），
        # 只允许关注 key 位置 k 满足：k >= q_pos - window 且 k <= q_pos
        q_pos = torch.arange(
            query_offset, query_offset + q_len, device=device
        ).unsqueeze(1) # shape: [q_len, 1]
        k_pos = torch.arange(kv_len, device=device).unsqueeze(0) # shape: [1, kv_len]
        return (k_pos <= q_pos) & (k_pos >= q_pos - sliding_window_size)

    # 在 _run_sdpa_forward_extend 中（类似改动也适用于 decode）：
    # ...
    attn_mask = None
    is_causal = causal
    if sliding_window_size is not None and sliding_window_size > -1:
        # 当启用滑动窗口时，构建布尔掩码并传给 SDPA
        # 同时必须将 is_causal 设为 False，因为 SDPA 不支持 is_causal=True 与自定义 attn_mask 同时使用
        attn_mask = self._make_sliding_window_mask(
            q_len=seq_len_kv,
            kv_len=seq_len_kv,
            sliding_window_size=sliding_window_size,
            device=per_req_query.device,
        )
        is_causal = False
    per_req_out_redudant = (
        scaled_dot_product_attention(
```

```
            per_req_query_redudant.unsqueeze(0),
            per_req_key.unsqueeze(0),
            per_req_value.unsqueeze(0),
            attn_mask=attn_mask,
            enable_gqa=enable_gqa,
            scale=scaling,
            is_causal=is_causal,
        )
        .squeeze(0)
        .movedim(query.dim() - 2, 0)
    )
```

评论区精华

PR 没有 review 评论。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 性能影响：对于启用滑动窗口的模型，每次 SDPA 调用前会进行 CPU 侧的 O(q_len × kv_len) 布尔掩码分配，但 PR 描述指出该开销相对于 SDPA 本身可以忽略。对于非滑动窗口模型，没有任何额外开销。
  2. 正确性：需要确保 query_offset 的计算在 decode 路径中正确反映 query 的位置。当前实现 query_offset = seq_len_kv - seq_len_q 假设 query 是序列的最后一个 token，这在 decode 场景下是正确的。
  3. 边界情况：sliding_window_size 为 -1 或 None 时，走原有 is_causal=True 路径，行为不变。
  4. 兼容性：仅在 torch_native 后端生效，不影响其他注意力后端。 - 影响：直接影响使用 torch_native 注意力后端的用户，特别是运行 Mistral、Gemma 等滑动窗口模型的用户。修复后这些模型将产生正确的注意力输出。对非滑动窗口模型无影响。测试方面，PR 提到后续会有专门的单元测试矩阵 PR 来进行验证。 - 风险标记：核心路径变更，无直接测试覆盖

关联脉络

- 暂无明显关联 PR