PR #36430 完整报告

sgl-project/sglang

[CPU] Fix truncated KV prefix in intel_amx spec verify

合并时间：2026-08-27 09:42

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36430

执行摘要

- 一句话：修复 intel_amx 后端 spec verify 的 KV 前缀截断
- 推荐动作：该 PR 值得精读，尤其是对于理解 speculative decoding 中 KV 元数据传递与 kernel 前缀推导的机制。关键设计决策在于保留元数据预计算值，而非从 forward_batch 直接读取，这提示了在修改前需深入理解元数据与 kernel 的契约。此外，该 PR 也暴露了测试覆盖不足的隐患，读者可关注后续 #35881 的跟进。

功能与动机

在 CPU 上使用 intel_amx 注意力后端进行 speculative decoding 时，模型输出与目标模型自身生成的输出不符，准确率严重下降。根据 PR 描述，问题根因是在 TARGET_VERIFY 模式下 _build_extend_metadata 将 seq_lens 设置为 forward_batch.seq_lens + num_draft_tokens，而 extend_attention_cpu 通过减法（seq_len_prefix = seq_lens - extend_seq_lens）推导提交前缀；但 forward_extend 和 forward_decode 中重复从 forward_batch.seq_lens 读取并覆盖了由元数据解析出的 seq_lens，导致前缀长度少了 num_draft_tokens，使得目标模型把每个 draft token 与自身上下文中的空洞计算，greedy verify 接受了目标模型自身 argmax 不会产生的 token。该修复解决了 #24013 引入的回归。

实现拆解

本 PR 专注于 python/sglang/srt/layers/attention/intel_amx_backend.py 的微调，改动量极小（+0/-2），具体步骤：

1. 删除 forward_extend 中重复的 seq_lens 覆盖：在 forward_extend 中，原代码先通过 self.extend_metadata 解包出 seq_lens, extend_seq_lens, extend_start_loc, tree_mask，随后又执行 seq_lens = forward_batch.seq_lens 将其覆盖，此覆盖已删除。现在保留从元数据解析出的 seq_lens，仅在其为 int32 时转换为 int64。
2. 删除 forward_decode 中重复的 seq_lens 覆盖：在 forward_decode 中，原代码在 draft_decode_metadata 非空时使用其中包含的展开长度，但随后在 reshape 后再次执行 seq_lens = forward_batch.seq_lens 覆盖，此覆盖已删除。
3. 保持 int64 转换逻辑：两处均保留 if seq_lens.dtype != torch.int64: seq_lens = seq_lens.to(torch.int64) 的转换。
4. 验证影响：非 speculative 的 extend 路径不受影响，因为该场景下 forward_batch.seq_lens 本就等于前缀加扩展长度。

5. 测试配套：本 PR 未添加单元测试，但根据评论者 ekintel 的后续说明，将在 follow-up PR #35881 中补齐覆盖两处站点的回归测试。

关键文件：

- python/sglang/srt/layers/attention/intel_amx_backend.py（模块 注意力后端；类别 source；类型 core-logic；符号 forward_extend, forward_decode）：核心修复文件，删除 forward_extend 与 forward_decode 中对 seq_lens 的重复覆盖，直接解决 KV 前缀截断问题。

关键符号：forward_extend, forward_decode

关键源码片段

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

核心修复文件，删除 forward_extend 与 forward_decode 中对 seq_lens 的重复覆盖，直接解决 KV 前缀截断问题。

```
# python/sglang/srt/layers/attention/intel_amx_backend.py
# forward_extend: 保留 extend_metadata 中解析出的 seq_lens，不再用 forward_batch.seq_lens 覆盖

def forward_extend(...):
    # ... 省略前面的 KV 写入逻辑 ...

    # 预计算一次：规范验证批次不携带 extend_* 字段（见 _build_extend_metadata）
    seq_lens, extend_seq_lens, extend_start_loc, tree_mask = self.extend_metadata

    _, max_extend_len = self.forward_metadata
    # 关键修复：删除原有的 seq_lens = forward_batch.seq_lens 覆盖
    # 在 TARGET_VERIFY 模式下，seq_lens = forward_batch.seq_lens + num_draft_tokens
    # 因此必须保留此值，kernel 才能通过 seq_len_prefix = seq_lens - extend_seq_lens 得到正确前缀
    if seq_lens.dtype != torch.int64:
        seq_lens = seq_lens.to(torch.int64)

    self.extend_attention_fwd(
        q.view(-1, layer.tp_q_head_num, layer.qk_head_dim),
        k, v,
        o.view(-1, layer.tp_q_head_num, layer.v_head_dim),
        self.token_to_kv_pool.get_key_buffer(layer.layer_id),
        self.token_to_kv_pool.get_value_buffer(layer.layer_id),
        self.req_to_token_pool.req_to_token,
        forward_batch.req_pool_indices,
        seq_lens, # 使用保留的 seq_lens
        extend_seq_lens,
        extend_start_loc,
        max_extend_len,
        layer.scaling,
        layer.logit_cap,
```

```
            layer.is_cross_attention,
            layer.sliding_window_size + 1,
            forward_batch.encoder_lens,
            sinks,
            tree_mask,
        )
        return o.view(-1, layer.tp_q_head_num * layer.v_head_dim)

    # forward_decode: 同样删除覆盖，保留 draft_decode_metadata 中的展开长度

    def forward_decode(...):
        # ... 省略前面逻辑 ...
        if self.draft_decode_metadata is not None:
            req_to_token, seq_lens, req_pool_indices = self.draft_decode_metadata
        else:
            req_to_token = self.req_to_token_pool.req_to_token
            req_pool_indices = forward_batch.req_pool_indices
            seq_lens = forward_batch.seq_lens

        q = q.reshape(-1, layer.tp_q_head_num * layer.qk_head_dim)
        # 关键修复：不再用 forward_batch.seq_lens 覆盖 seq_lens，保留 draft_decode_metadata
        中的候选长度
        # 否则 speculative_num_steps > 1 时每个候选的扩展长度会被丢弃，导致验证错误
        if seq_lens.dtype != torch.int64:
            seq_lens = seq_lens.to(torch.int64)

        # ... 以下逻辑保持不变 ...
```

评论区精华

讨论主要集中在 CI 失败分析和后续跟进：

- CI 失败归因：htzo 指出 Xeon build-test 失败源于 MMLU corpus 403，ROCm 失败源于连接问题（scheduler 崩溃非本 PR 引起），基础 PR 测试本身即失败，并推测 #36281 可能是元凶。
- 测试覆盖和遗留问题的解决：ekintel 在评论中提出在 #35881 中跟进两处潜在漏洞：补充覆盖两个修改点的回归测试，以及移除 test_spec_eagle_parity_cpu 中过时的 disabled="EAGLE3 numerical parity mismatches on CPU intel_amx" 标记，因为那些不匹配正是本次修复的前向扩展症状。该测试已在 Xeon 上双向验证：main 上 4 passed，058f99c5^ 上 2 failed, 2 passed，仅两个 speculative 用例失败。它通过 stub 内核入口，无需模型或服务器。
- CI 失败归因 (other): CI 失败与本次改动无关，未阻塞合并。
- 补充回归测试与清理过时 marker (testing): 通过评论达成将在后续 PR 补足的共识，本 PR 未包含测试。

风险与影响

- 风险：本 PR 的改动极小，但涉及核心注意力路径，存在以下潜在风险：
- 回归风险：删除覆盖可能导致在非 speculative 场景下 seq_lens 值不正确，尽管 PR 描述称非 speculative extend 路径中 forward_batch.seq_lens 与本值相同，但风险仍在，尤其是对于特殊模型或边界情况。
- 测试覆盖缺失：缺少直接的单元测试，可能无法捕获未来类似回归。
- 兼容性风险：此修复依赖于 _build_extend_metadata 的实现，若未来该函数的行为改变，需要同步审视此修复。
- 性能影响：无此风险，删除多余读取反而略微加快。
- 影响：影响范围：主要影响 CPU 上使用 intel_amx 注意力后端的 speculative decoding 用户，修复了准确率急剧下降的问题。非 speculative 用户无影响。团队方面，该修复维护了 CPU 后端的正确性，并为同类问题提供了模式。影响程度：中等偏高，定点修复并已用基线准确率验证。
- 风险标记：缺少测试覆盖，核心路径变更，依赖特定元数据约定

关联脉络

- PR #24013 [VLM] Batch cross-request ViT encoding and reuse attention metadata: 该 PR 被指为本回归的引入源，本 PR 正是修复它造成的 seq_lens 覆盖问题。
- PR #35881 [CPU] Follow-up on intel_amx spec verify fix: 由维护者 ekintel 在评论中提出，作为本 PR 的后续跟进，用于补齐回归测试和清理过时 marker。