PR #8666 完整报告

sgl-project/sglang

[CPU] Fix issues when running llama3.2-11B vision model with image tasks

合并时间：2026-05-21 13:09

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/8666

执行摘要

- 一句话：修复 CPU 上 LLaMA 3.2 11B 视觉模型的 Cross-Attention 与设备绑定等问题
- 推荐动作：该 PR 值得精读，尤其关注以下设计决策：如何通过参数化扩展已有注意力内核以支持 cross-attention、跨注意力时 KV 索引偏移的计算逻辑、以及 Python 后端如何兼容多个注意力后端。C++ 内核中条件编译和后处理保留的权衡也值得参考。

功能与动机

根据 PR 描述：'This PR fixes several issues encountered when running the LLaMA 3.2 11B vision model with image tasks on CPU.' 主要动机是解决 CPU 上运行该视觉模型时出现的 KV 缓存访问错误、设备硬编码、属性访问异常等问题，使其图像任务能够正常执行。

实现拆解

1. C++ 内核扩展（sgl-kernel/csrc/cpu/extend.cpp、decode.cpp）：在 extend_attention_kernel_impl 和 decode_attention_kernel_impl 等函数签名中增加 encoder_lens、is_cross_attn 和 has_encoder_lens 参数。在循环中根据 is_cross_attn 分支决定 KV 范围（来自 encoder_lens 或常规序列长度），并通过 kv_offset 调整请求到 token 的索引偏移。cross-attention 情况下跳过 stage2 三角形部分，但保留后处理。
2. 注意力后端改造：
   - torch_native_backend.py：在 _run_sdpa_forward_extend 和 _run_sdpa_forward_decode 中增加 encoder_lens 和 is_cross_attn 参数。在循环中根据标志设置 start_kv 和 end_kv，从 encoder_lens 或偏移后取值。同时更新 forward_extend 和 forward_decode 以传递 encoder_lens 和 layer.is_cross_attention。
   - intel_amx_backend.py：类似地，在 forward_extend 和 forward_decode 中根据 layer.is_cross_attention 选择 cache_loc（out_cache_loc 或 encoder_out_cache_loc），并将 is_cross_attention 和 encoder_lens 传递给底层 C++ 内核。当 k、v 为 None 时跳过缓存存储。
3. 模型修复（python/sglang/srt/models/mllama.py）：移除 torch.cuda.current_device() 硬编码，改用 self.device；修正 mm_inputs 属性访问错误。
4. 线性层 bias 初始化（python/sglang/srt/layers/linear.py）：将 LinearLayer 的 bias 从 torch.empty 改为 torch.zeros，解决某些模型（如 LLaMA 3.2 11B）缺少 bias 张量但权重存在时精度问题。随后限定在 CPU 上使用 zeros。

5. 量化层维度支持（python/sglang/srt/layers/quantization/unquant.py）：在 weight_packed_linear 前展平超过 2D 的输入，计算后再恢复形状。
6. 基准测试（python/sglang/bench_serving.py）：修正 LLaMA 模型的 chat template 解析，将图像字段从 image_url 改为 image 类型。
7. 测试更新：更新 test/registered/cpu/test_extend.py、test/registered/cpu/test_decode.py、test/registered/cpu/test_gemm.py 等，新增 cross-attention 场景的测试用例。

关键文件：

- sgl-kernel/csrc/cpu/extend.cpp（模块 CPU 内核；类别 source；类型 core-logic；符号 extend_attention_kernel_impl）：核心变更：为 extend 注意力内核增加 cross-attention 支持，引入 encoder_lens、is_cross_attn 参数，调整 KV 索引偏移和循环范围。
- sgl-kernel/csrc/cpu/decode.cpp（模块 CPU 内核；类别 source；类型 core-logic；符号 decode_attention_kernel_impl, decode_attention_grouped_kernel_impl）：核心变更：类似地增强 decode 注意力内核，加入 cross-attention 支持和 encoder_lens 偏移。同时修复了空 split 时 logit 值未初始化的问题。
- python/sglang/srt/layers/attention/torch_native_backend.py（模块 注意力后端；类别 source；类型 dependency-wiring；符号 _run_sdpa_forward_extend, _run_sdpa_forward_decode, forward_extend, forward_decode）：关键前端：为 torch native 注意力后端统一扩展 cross-attention 支持，使 SDPA 可以处理编码器 KV 缓存。
- 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）：关键前端：为 Intel AMX 注意力后端扩展 cross-attention 支持，选择正确的 cache_loc 并向下游传递参数。
- python/sglang/srt/models/mllama.py（模块 模型层；类别 source；类型 data-contract；符号 VisionMllamaForCausalLM）：模型修复：移除硬编码 CUDA 设备并修正属性访问，使模型能正确在 CPU 上初始化。
- python/sglang/srt/layers/linear.py（模块 线性层；类别 source；类型 data-contract；符号 LinearLayer.init）：全局影响：将 bias 从 torch.empty 改为 torch.zeros，解决某些模型缺失 bias 张量时的精度问题。变更经过讨论后限定在 CPU 平台。
- python/sglang/srt/layers/quantization/unquant.py（模块 量化层；类别 source；类型 core-logic；符号 weight_packed_linear）：功能修复：支持 weight_packed_linear 的高维输入（>2D），解决视觉模型中 3D tensor 传递问题。

关键符号：extend_attention_kernel_impl, decode_attention_kernel_impl, decode_attention_grouped_kernel_impl, _run_sdpa_forward_extend, _run_sdpa_forward_decode, IntelAmxBackend.forward_extend, IntelAmxBackend.forward_decode, VisionMllamaForCausalLM.forward, LinearLayer.init

关键源码片段

sgl-kernel/csrc/cpu/extend.cpp

核心变更：为 extend 注意力内核增加 cross-attention 支持，引入 encoder_lens、is_cross_attn 参数，调整 KV 索引偏移和循环范围。

```
// extend.cpp 核心函数签名变化（以模板函数 extend_attention_kernel_impl 为例）
// 新增 encoder_lens、is_cross_attn、has_encoder_lens 参数

template <typename scalar_t, typename index_t, int BLOCK_M, int BLOCK_N>
void extend_attention_kernel_impl(
    // ... 常规参数 ...
    const int64_t* __restrict__ encoder_lens, // 新参数：编码器长度
    const index_t* __restrict__ extend_seq_lens,
    const index_t* __restrict__ extend_start_loc,
    // ...
    bool is_prefix_skipped,
    bool is_cross_attn, // 新参数：是否为跨注意力
    bool has_encoder_lens) { // 新参数：是否有编码器长度值

  // ... 初始化 ...
  for (int i = begin; i < end; ++i) {
    int seq_len = seq_lens[bs];
    int seq_len_extend = extend_seq_lens[bs];
    int seq_len_prefix = seq_len - seq_len_extend;
    int seq_extend_start_loc = extend_start_loc[bs];
    int req_pool_id = req_pool_indices[bs];

    // 计算 kv_offset：非 cross-attn 且存在 encoder_lens 时偏移
    int kv_offset = (has_encoder_lens && (!is_cross_attn)) ? encoder_lens[bs] : 0;

    // ...
    // 设置 KV 循环范围
    int kv_start = 0;
    int kv_end = is_cross_attn ? encoder_lens[bs] : seq_len_prefix;

    // stage 1: 用 prefix 或 encoder 的 KV 计算注意力
    for (int n = kv_start; n < kv_end; n += BLOCK_N) {
      int n_size = std::min(BLOCK_N, kv_end - n);
      // pack_vnni 使用 req_to_token 索引时加上 kv_offset
      pack_vnni<scalar_t, index_t>(
          Btmp,
          k_buffer + head_kv_id * k_strideH,
          req_to_token + req_pool_id * max_context_len + n + kv_offset,
          n_size, head_size, ...);
      // ... 后续注意力计算 ...
    }
    // cross-attention 时跳过 stage2（三角形部分），但后处理仍然执行
    if (!is_cross_attn) {
      // stage 2: 处理 extend 部分的自注意力
    }
    // 后处理：softmax 归一化并写入 output
```

```
  }
}
```

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

关键前端：为 torch native 注意力后端统一扩展 cross-attention 支持，使 SDPA 可以处理编码器 KV 缓存。

```
# torch_native_backend.py 中 _run_sdpa_forward_extend 部分
# 新增 encoder_lens 和 is_cross_attn 参数
def _run_sdpa_forward_extend(
    self,
    query: torch.Tensor,
    k_cache: torch.Tensor,
    v_cache: torch.Tensor,
    req_to_token: torch.Tensor,
    req_pool_indices: torch.Tensor,
    seq_lens: torch.Tensor,
    extend_prefix_lens: torch.Tensor,
    extend_seq_lens: torch.Tensor,
    encoder_lens: Optional[torch.Tensor] = None, # 新参数
    scaling=None,
    enable_gqa=False,
    causal=False,
    is_cross_attn=False, # 新参数
):
    # ...
    for seq_idx in range(seq_lens.shape[0]):
        extend_seq_len_q = extend_seq_lens[seq_idx]
        prefill_seq_len_q = extend_prefix_lens[seq_idx]
        seq_len_kv = seq_lens[seq_idx]
        end_q = start_q + extend_seq_len_q

        # 关键：根据 encoder_lens 和 is_cross_attn 决定 start_kv 和 end_kv
        if encoder_lens is not None:
            if is_cross_attn:
                start_kv = 0
                end_kv = encoder_lens[seq_idx]
            else:
                start_kv = encoder_lens[seq_idx]
                end_kv = start_kv + seq_len_kv
        else:
            start_kv = 0
            end_kv = start_kv + seq_len_kv

        # 获取对应的 KV 缓存索引
        per_req_tokens = req_to_token[req_pool_idx, start_kv:end_kv]
        per_req_key = k_cache[per_req_tokens].movedim(...)
        per_req_value = v_cache[per_req_tokens].movedim(...)
        # ... 后续 SDPA 调用
```

评论区精华

- intel_amx_backend 实现策略：mingfeima 建议不要在该后端内嵌 SDPA 实现，而是直接让 *extend/decode* 内核支持 cross-attention。作者最初使用混合方案，后续改为内核原生支持。
- bias 初始化争议：mingfeima 质疑为何将 bias 从 *empty* 改为 *zeros*。作者解释某些模型（LLaMA 3.2 11B）的权重中缺少 bias 张量，*empty* 可能导致未初始化值影响精度。最终仅在 CPU 上使用 *zeros*。
- *extend* 内核 *cross-attention* 阶段 2：mingfeima 提问 *cross-attention* 是否不需要 *stage2* ，作者确认跳过 *stage2*，但后续后处理仍需执行。
- *cache_loc* 位置讨论：mingfeima 建议将 *cache_loc* 计算移至 forward_metadata 以每批次只算一次。作者回应其依赖 layer.is_cross_attention 运行时值，不能提前确定。
- 代码风格改进：mingfeima 指出应使用 Optional 类型注解、避免命名参数传递等，作者已采纳。
  - intel_amx_backend 中 *cross-attention* 的实现方式 (*design*): 最终采用内核原生支持 *cross-attention*，统一了两种后端的实现路径。
  - bias 初始化 *empty* 改为 *zeros* (*correctness*): 仅在 CPU 上将 bias 初始化为 *zeros*，其他平台保持 *empty*。
  - *extend* 内核中 *cross-attention* 跳过 *stage2* (*correctness*): 跳过 *stage2* 但仍执行后处理，逻辑正确。
  - *cache_loc* 是否应提前到 forward_metadata (*performance*): 保持每迭代计算，因依赖于 layer 运行时属性。
  - 代码风格：使用 Optional 类型与避免命名参数 (style): 作者修改了类型注解和参数传递方式。

风险与影响

- 风险：
  - 回归风险：*extend/decode* 内核函数签名增加了参数，但调用方已同步更新；非 *cross-attention* 场景 is_cross_attn 为 false，行为与原逻辑一致。但 kv_offset 的引入改变了索引计算方式，若请求池索引布局有意外假设可能导致越界访问。
  - 性能风险：*cache_loc* 和 *encoder_lens* 在每轮迭代中重复计算可能引入少量开销，但最终方案无法移到 *metadata*，这部分开销占比很小。
  - 精度风险：bias 初始化改为 *zeros* 可能影响其他依赖特定初始化的模型，但变更已限定 CPU。
  - 安全风险：无。
  - 兼容性：新增 *encoder_lens* 作为可选参数，向后兼容。
- 影响：
  - 用户：CPU 用户现在可以正常使用 LLaMA 3.2 11B 视觉模型进行图像描述、多模态对话等任务。

- 系统：注意力后端统一支持 cross-attention，为未来更多多模态模型的 CPU 部署奠定基础。
- 团队：需维护新增的 cross-attention 内核路径，但 *core* 逻辑与已有 *extend/decode* 共享，维护成本较低。
- 风险标记：核心内核变更，跨注意力逻辑复杂，bias 初始化影响广泛，测试覆盖待完善

关联脉络

- 暂无明显关联 PR