PR #24582 完整报告

sgl-project/sglang

[NPU] Enhance accuracy for model Step3_5 from 0 to 88%

合并时间：2026-05-29 11:29

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24582

执行摘要

- 一句话：NPU 端 Step-3.5 模型精度从 0% 提升至 88%
- 推荐动作：建议 NPU 后端开发者和模型适配者仔细研读此 PR，特别是 FIA sparse_mode 4 与自定义 mask 的组合技巧、激活函数 clamp 的设计、以及 CANN 版本兼容性处理。对于使用 Step-3.5-Flash 的用户，强烈建议升级并按文档设置环境变量以获得最佳精度。未来的工作可以考虑将 swiglustep_and_mul 迁移至 sgl-kernel-npu，并为此路径补充单元测试。

功能与动机

Step-3.5-Flash 模型在 NPU 上使用 Ascend 后端时推理精度仅为 0%，主要原因是原有 FIA 注意力实现不支持滑动窗口（SWA），且 MoE 激活函数缺少 clamp 操作。在 CANN 8.5 环境下，当 query 长度为 1 时 sparse_mode 4 的 pre/next_tokens 参数失效，需引入 SWA mask。同时，MoE 的激活函数需要与 GPU 端的 clamp 行为对齐以提升精度。

实现拆解

1. FIA 注意力重构：在 ascend_backend.py 中添加 get_swa_mask 方法，为滑动窗口注意力生成 mask（left_context 通过参数传入，实际调用时使用 layer.sliding_window_size）。在 forward_extend 中，当层启用 hybrid SWA 且使用 FIA 时，改用 npu_fused_infer_attention_score_v2 并设置 sparse_mode=4、pre_tokens=sliding_window_size、next_tokens=0，利用 CANN 内置的滑动窗口实现。在 forward_decode 中，由于 decoder 每步 query 长度为 1，pre/next_tokens 对 sparse_mode 4 无效，因此改为通过 atten_mask 参数传入自定义的 SWA mask，同时保持 sparse_mode=4 并设 pre_tokens=0、next_tokens=0，让 CANN 忽略窗口参数而完全依靠 mask 遮罩。调整 block_tables 的选择逻辑：当 self.is_hybrid_swa and layer.sliding_window_size != -1 时使用 block_tables_swa。
2. MoE 激活函数重构：在 unquant.py 中添加全局函数 swiglustep_and_mul，对 gate 部分执行 silu -> clamp(max=limit)，对 up 部分执行 clamp(min=-limit, max=limit)，然后相乘，与 GPU 端 _swiglu_silu_clamp_mul 对齐。在 _forward_npu 分支的激活函数选择中，当 activation == "silu" 且 gemm1_clamp_limit != None 时调用此函数，否则回退到原有 npu_swiglu。
3. 启用条件：需要设置环境变量 ASCEND_USE_FIA=1 以启用 FIA 后端路径。对于 AIME 数据集还需设置 SGLANG_NPU_FORWARD_NATIVE_GEMMA_RMS_NORM=1。同时需要

cann>=8.5 以获得 sparse_mode 4 支持。

关键文件：

- python/sglang/srt/hardware_backend/npu/attention/ascend_backend.py（模块 注意力模块；类别 source；类型 core-logic；符号 get_swa_mask, forward_extend, forward_decode）：核心文件，新增 get_swa_mask 方法并重写了 forward_extend 和 forward_decode 中的 FIA 路径，是精度提升的主要贡献者。
- python/sglang/srt/layers/quantization/unquant.py（模块 MoE 模块；类别 source；类型 core-logic；符号 swiglustep_and_mul）：新增 swiglustep_and_mul 激活函数以对齐 GPU 行为，在 MoE 前向时替换原有 npu_swiglu，是精度提升的另一关键。

关键符号：get_swa_mask, swiglustep_and_mul, forward_extend, forward_decode

关键源码片段

python/sglang/srt/hardware_backend/npu/attention/ascend_backend.py

核心文件，新增 get_swa_mask 方法并重写了 forward_extend 和 forward_decode 中的 FIA 路径，是精度提升的主要贡献者。

添加到 AscendAttnMaskBuilder 类中 defget_swa_mask(self, seq_lens: torch.Tensor, s2: int, left_context=512) -> torch.Tensor: """ 生成滑动窗口注意力 mask。 当 sequence 长度 s2 > left_context 时，将 left_context 之前的 token 和 pad token 置为 masked。 实际使用时 left_context 应传入 layer.sliding_window_size（通过调用方覆盖默认 512）。 """ if seq_lens.dim() == 1: seq_lens = seq_lens.unsqueeze(1) # [b] -> [b,1] b = seq_lens.size(0) device = seq_lens.device indices = torch.arange(s2, device=device).unsqueeze(0).expand(b, -1) # [b, s2] start_indices = torch.clamp(seq_lens - left_context, min=0) # [b,1] mask = (indices < start_indices) | (indices >= seq_lens) # [b, s2] return mask.unsqueeze(1).to(self.device, non_blocking=True) # [b,1,s2] 在 forward_decode 中调用该函数生成 mask 并传给 npu_fused_infer_attention_score_v2 的 atten_mask 参数。

python/sglang/srt/layers/quantization/unquant.py

新增 swiglustep_and_mul 激活函数以对齐 GPU 行为，在 MoE 前向时替换原有 npu_swiglu，是精度提升的另一关键。

```
def swiglustep_and_mul(x: torch.Tensor, limit: float = 7.0) -> torch.Tensor:
    """
    Out-variant of swiglustep activation.
    计算: silu(x[:d]).clamp(max=limit) * x[d:].clamp(-limit, limit)
    """
    gate, up = x.chunk(2, dim=-1) # split at hidden//2
    gate = F.silu(gate)
    gate = gate.clamp(max=limit)
    up = up.clamp(min=-limit, max=limit)
    out = gate * up
    return out
```

```
# 在 _forward_npu 中使用:
elif self.moe_runner_config.activation == "silu":
    if self.moe_runner_config.gemm1_clamp_limit is not None:
        hidden_states = swiglustep_and_mul(
            hidden_states, self.moe_runner_config.gemm1_clamp_limit
        )
    else:
        hidden_states = torch.ops.npu.npu_swiglu(hidden_states)
```

评论区精华

- 激活函数位置讨论：Todobe 建议将 swiglustep_and_mul 移入 sgl-kernel-npu，认为公共代码不适合添加 NPU 函数。作者表示采纳但最终未移动（函数仍位于 unquant.py）。
- SWA 窗口大小问题：AndyLi429 指出 get_swa_mask 的 left_context 参数固定 512 会与其他模型窗口大小不一致，应使用 layer.sliding_window_size。作者采纳并在调用时传入窗口大小。
- decode 阶段 mask 必要性：Todobe 询问为什么 prefill 阶段不需要 mask。作者解释当 query 长度为 1 时，pre/next_tokens 对 sparse_mode 4 无效，因此 decode 必须手动传入 mask。
- block_size 硬编码：AndyLi429 指出在前向中部分地方硬编码 128 作为 block size，应使用 self.page_size。作者修复。
- k/v head dim 区分：AndyLi429 指出在 MLA 中 qk_head_dim 与 v_head_dim 通常不等，代码中应区分使用。作者修正。
  - swiglustep_and_mul 函数位置 (design): 作者表示采纳，但最终未移动函数，仍然留在 unquant.py。
  - SWA 窗口大小参数 (correctness): 作者采纳并在调用 get_swa_mask 时传入了 layer.sliding_window_size。
  - decode 阶段 SWA mask 必要性 (question): 作者解释当 query 长度为 1 时 pre/next_tokens 对 sparse_mode 4 无效，所以 decode 需要手动 mask。
  - block_size 硬编码 128 (correctness): 作者修复。
  - qk_head_dim 与 v_head_dim 区分 (correctness): 作者修复。

风险与影响

- 风险：
  1. CANN 版本依赖：FIA sparse_mode=4 的行为依赖 CANN 8.5/9，且 8.5 在 qlen=1 时有上述限制，升级 CANN 9 可能改变行为，需要持续关注兼容性。
  2. 仅验证单一模型：精度提升仅在 Step-3.5-Flash 上验证，其他 NPU 模型（尤其是使用 hybrid SWA 的模型）未经过测试，可能引入回归。
  3. 缺少单元测试：新的 get_swa_mask 和 FIA 分支逻辑没有对应的单元测试，覆盖依赖端到端评测，使得回归发现较晚。
  4. 激活函数默认 clamp 值：swiglustep_and_mul 默认 limit=7.0，若其他模型需要不同值但未通过 gemm1_clamp_limit 指定，可能产生精度偏差。 - 影响：主要影响使用 NPU

（Ascend）后端的 Step-3.5-Flash 模型推理，精度从 0% 提升至 88%+（gms8k）和 93%（AIME25），超出 GPU 基线。用户必须设置环境变量 ASCEND_USE_FIA=1 以启用新的 FIA 路径。对其他模型无直接影响，但修改了 ascend_backend.py 中的前向逻辑（forward_extend 和 forward_decode），可能影响其他使用 hybrid SWA 的模型，需要验证。对团队而言，新增的 FIA+SWA mask 路径增加了代码复杂度，需维护两份注意力实现（sinks 和 mask 分支）。 - 风险标记：CANN 版本依赖，缺少单元测试，仅验证单一模型

关联脉络

- 暂无明显关联 PR