PR #26147 完整报告

sgl-project/sglang

[NPU] Add Gemma4 Sliding Window Attention support on Ascend backend

合并时间：2026-06-12 11:44

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26147

执行摘要

- 一句话：为 Ascend NPU 添加 Gemma4 SWA 支持
- 推荐动作：该 PR 是 Ascend NPU 后端支持混合滑动窗口注意力的重要里程碑，涉及 mask 生成、图模式支持、回退路径增强、语义对齐等多层面改动。代码质量较高，设计决策清晰（如窗口约定转换、图模式预分配策略），评测数据充分。建议关注 SWA 路径在图模式下的性能表现和测试覆盖的持续性。值得精读，尤其是 ascend_backend.py 中 _init_cuda_graph_metadata 的 mask 更新逻辑和 ascend_torch_native_backend.py 的窗口裁剪实现。

功能与动机

Gemma4 uses a hybrid Sliding Window Attention (SWA) architecture where 5/6 of transformer layers use local (windowed) attention and 1/6 use global attention. The Ascend NPU backend lacked SWA support, preventing Gemma4 models from running correctly on Ascend NPU.

实现拆解

1. SWA Mask 与图模式支持：在 AscendAttnMaskBuilder 中新增 get_swa_mask 方法；ForwardMetadata 增加 swa_mask 字段；init_cuda_graph_state 预分配 swa_mask 和 swa_indices 缓冲区，图回放时通过 init_forward_metadata_replay_cuda_graph 动态更新 mask。
2. SWA 页式块表构建：新增 _is_swa_layer、_can_use_tnd 等辅助方法，以及 _build_swa_paged_block_tables、_get_paged_attention_inputs 等核心方法，实现窗口裁剪后的块表构建，用于 FIA、SDPA、PA 路径。
3. SDPA 回退路径增强：在 AscendTorchNativeAttnBackend.run_sdpa_forward_extend 中增加 sliding_window_size 和 full_to_swa_mapping 参数，实现 extend 阶段的 KV 窗口裁剪及 full/SWA pool 索引翻译。
4. 模型层语义对齐：gemma4_causal.py 使用 get_attention_sliding_window_size 将滑动窗口从 HuggingFace 的 inclusive 转为 SGLang 的 exclusive 约定；gemma4_mm.py 将 boolean 索引替换为 torch.where 以兼容 NPU 图模式。
5. 测试与配置：新增 8 个测试文件（VLM 和 LLM 各 4 个），覆盖 Gemma4 全部变体；memory_pool_npu.py 传递 SWA 参数；server_args.py 更新注意力后端选择逻辑以支持

Ascend。

关键文件：

- python/sglang/srt/hardware_backend/npu/attention/ascend_backend.py（模块 NPU 注意力；类别 source；类型 core-logic；符号 _is_swa_layer, _can_use_tnd, SWA_INT_MAX, _build_swa_paged_block_tables）：核心变更文件，新增 SWA mask 生成、图模式支持、页式块表构建等核心逻辑
- python/sglang/srt/hardware_backend/npu/attention/ascend_torch_native_backend.py（模块 NPU 回退；类别 source；类型 dependency-wiring；符号 run_sdpa_forward_extend）：SDPA 回退路径扩展 SWA 支持，窗口裁剪与 full index 翻译
- python/sglang/srt/models/gemma4_causal.py（模块 模型定义；类别 source；类型 data-contract；符号 get_attention_sliding_window_size, sliding_window）：滑动窗口语义对齐，使用 get_attention_sliding_window_size 转换为 exclusive 约定
- python/sglang/srt/models/gemma4_mm.py（模块 多模态模型；类别 source；类型 data-contract；符号 forward, ple_ids）：NPU 图模式兼容性修复，用 torch.where 替换 boolean 索引
- python/sglang/srt/mem_cache/memory_pool_npu.py（模块 内存池；类别 source；类型 configuration）：传递 SWA 参数（swa_head_num 等）至 SWAKVPool
- test/manual/ascend/vlm_models/test_npu_gemma_4_e2b.py（模块 模型测试；类别 test；类型 test-coverage；符号 TestGemma4E2B, test_vlm_mmmu_benchmark）：新增模型级测试，验证 Gemma4-E2B-it 在 MMMU 上的准确率不低于 0.15

关键符号：_is_swa_layer, _can_use_tnd, _build_swa_paged_block_tables, run_sdpa_forward_extend, init_cuda_graph_state, init_forward_metadata_replay_cuda_graph

关键源码片段

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

核心变更文件，新增 SWA mask 生成、图模式支持、页式块表构建等核心逻辑

```
# ascend_backend.py - 滑动窗口注意力核心数据结构和辅助方法

import torch

# 全注意力窗口的最大值（相当于不限制窗口）
FULL_ATTENTION_WINDOW = 2147483647

class ForwardMetadata:
    """前向元数据，包含注意力计算所需的各种张量"""
    swa_mask: Optional[torch.Tensor] = None # 滑动窗口注意力 mask，图模式 decode 使用，True 表示遮盖

class AscendAttentionBackend(AttentionBackend):
```

```
    def _is_swa_layer(self, layer: RadixAttention) -> bool:
        """判断给定层是否为滑动窗口注意力层。
        仅在启用混合 SWA 且 layer.sliding_window_size > -1 时返回 True。
        """
        return (
            self.is_hybrid_swa
            and layer.sliding_window_size is not None
            and layer.sliding_window_size > -1
        )

    @staticmethod
    def _can_use_tnd(layer: RadixAttention) -> bool:
        """检查是否可以使用 TND 布局。
        TND 要求 qk_head_dim 与 v_head_dim 相等且为 128 或 192，
        或者 d=192、v=128 的特殊组合。
        """
        d = layer.qk_head_dim
        v = layer.v_head_dim
        return (d == v and d in (128, 192)) or (d == 192 and v == 128)

    def init_cuda_graph_state(self, max_bs: int, max_num_tokens: int):
        """初始化 CUDA 图状态，预分配 SWA mask 和索引缓冲。"""
        total_context_len = self.max_context_len + 1
        self.graph_metadata['swa_mask'] = torch.ones(
            (max_bs, 1, total_context_len),
            dtype=torch.bool,
            device=self.device,
        )
        self.graph_metadata['swa_indices'] = torch.arange(
            total_context_len, device=self.device, dtype=torch.int32
        )
```

评论区精华

Review 中主要讨论了以下要点：

- AndyLi429 指出 get_swa_mask 是死代码，作者已删除；
- AndyLi429 询问 FIA 路径是否支持 block_table，作者说明实际使用 BSND fallback，TND v2 路径为预留给未来扩展（基于 #24582）；
- liupeng374 对 gemma4_causal.py 中 sliding_window 从 None 改为 -1 表示质疑，作者解释 -1 是 SGLang 中"无窗口"的约定，并通过引用 RadixAttention 和 FlashInferBackend 的 dispatch 逻辑证明对 GPU 无影响，同时指出这是将 HuggingFace 的 inclusive 窗口转换为 exclusive 的 deliberate fix；
- liupeng374 关注 graph replay 中 SWA mask 更新的性能开销，作者给出新增 ~2.3ms（约 9%）的数据；
- Todobe 建议将 SWA_INT_MAX 定义移到函数之前，作者采纳；

- Hexq0210 指出测试文件不应在 test/registered 中使用 register_npu_ci，作者将全部 8 个测试文件移至 test/manual 并移除注册。
- Dead code get_swa_mask (correctness): 作者确认是死代码并移除
- Gemma4 sliding_window None to -1 (correctness): 作者解释 -1 是 SGLang 的独家约定，该改动是 inclusive 转 exclusive 的修正，引用 RadixAttention 和 FlashInferBackend 证明无影响
- SWA mask graph replay performance (performance): 作者给出 ~2.3ms（约 9% forward time）
- Test registration and location (testing): 作者将所有测试文件从 test/registered 移至 test/manual 并移除注册
- NPU graph compatibility in gemma4_mm (correctness): 作者解释功能等价且对 GPU 无影响，仅因 NPU 图模式不支持 boolean 索引

风险与影响

- 风险：
  1. 新 SWA 路径与现有 FIA、SDPA 路径的交互可能引入未预期行为，尤其是图模式下的 swa_mask 更新逻辑依赖 seq_lens 的动态裁剪，若边界处理不完全可能产生错误 mask。
  2. 滑动窗口从 inclusive 转 exclusive 的语义对齐改动，虽经论证对 GPU 无影响，但若其他模型代码也依赖原始 config.sliding_window 直接使用，可能潜在一致性问题。
  3. 测试文件从 test/registered 移至 test/manual，不再纳入常规 CI，仅通过 nightly 运行，存在回归漏检风险。
  4. SWA mask 更新在 decode 图回放中增加约 2.3ms（约 9%），对高吞吐场景可能产生性能影响。 - 影响：对用户：Gemma4 模型可在 Ascend NPU 上正确运行，AIME 准确率提升 9%~17%，MMMU Pro 提升 8%~11%。对系统：增加 SWA 图模式预分配，内存占用略有增加；新增 SWAKVPool 参数传递路径。对团队：完善 NPU 后端混合 SWA 支持，为后续其他模型的 SWA 支持提供参考架构。对 GPU 等其他后端无影响。 - 风险标记：SWA 图模式性能开销，滑动窗口语义对齐风险，测试回归漏检，NPU 图模式兼容性

关联脉络

- PR #24582 Related reference for TND v2 path: Review 中表示 TND v2 路径（npu_fused_infer_attention_score_v2）参考了该 PR，保留供未来使用