PR #25110 完整报告

sgl-project/sglang

[Fix]: BCG support for RadixLinearAttention (Qwen3.5 / linear-attn hybrid models)

合并时间：2026-05-23 04:30

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25110

执行摘要

- 一句话：修复 RadixLinearAttention BCG 支持，消除静默错误
- 推荐动作：推荐合入，该 PR 以极小的改动修复了一个严重影响模型输出正确性的 bug，且提供了充分的基准测试数据。值得精读的地方在于 *eager_on_graph* 装饰器的使用模式，这是 BCG 框架中确保某些操作不在 CUDA Graph 内被捕获的标准做法。

功能与动机

RadixAttention 在 #22218 中已接入 BCG 逻辑，但 RadixLinearAttention 没有。在混合线性注意力模型（如 Qwen3.5、Qwen3-Next、Kimi-Linear、Bailing-MoE-Linear）上开启 *--enable-breakable-cuda-graph* 时，线性注意力内核会被错误地捕获到分段 CUDA Graph 中，导致服务器静默输出错误结果。GSM8K 测试显示，修复前准确率仅 16.5%，修复后达到 86.7%。

实现拆解

1. 新增导入：在 radix_linear_attention.py 顶部新增对 *eager_on_graph* 和 is_in_breakable_cuda_graph 的导入，来自 sglang.srt.model_executor.breakable_cuda_graph 模块。
2. 扩展 forward 分支：在 RadixLinearAttention.forward 方法的 *extend* 分支中，原有的直接调用 unified_linear_attention_with_output 之前插入 if is_in_breakable_cuda_graph(): 判断。若在 BCG 上下文中，则调用 bcg_unified_linear_attention_with_output；否则仍走原路径 unified_linear_attention_with_output。这对其他执行模式（如 *decode*）无影响。
3. 注册 *eager_on_graph* 版本：在文件末尾新增 bcg_unified_linear_attention_with_output = *eager_on_graph*(True)(unified_linear_attention_with_output)，将原函数用 *eager_on_graph*(True) 包装，确保 BCG 模式下该段计算以 *eager* 模式运行，不会被 CUDA Graph 捕获。

关键文件：

- python/sglang/srt/layers/radix_linear_attention.py（模块 注意力层；类别 source；类型 *dependency-wiring*）：唯一修改的文件，修复了 RadixLinearAttention 在 BCG 模式下的静默错误。新增导入、forward 分支和 *eager_on_graph* 包装。

关键符号：未识别

关键源码片段

python/sglang/srt/layers/radix_linear_attention.py

唯一修改的文件，修复了 RadixLinearAttention 在 BCG 模式下的静默错误。新增导入、forward 分支和 *eager_on_graph* 包装。

```
# SPDX-License-Identifier: Apache-2.0

"""Radix linear attention - 支持 Breakable CUDA Graph 的线性注意力"""

from __future__ import annotations
from typing import TYPE_CHECKING, Optional, Tuple, Union
import torch
from torch import nn

# 新增导入：BCG 所需工具
from sglang.srt.compilation.compilation_config import register_split_op
from sglang.srt.compilation.piecewise_context_manager import get_forward_context
from sglang.srt.model_executor.breakable_cuda_graph.breakable_cuda_graph import (
    eager_on_graph,
)
from sglang.srt.model_executor.breakable_cuda_graph.context import (
    is_in_breakable_cuda_graph,
)
from sglang.srt.model_executor.forward_context import get_attn_backend
from sglang.srt.utils.custom_op import register_custom_op

if TYPE_CHECKING:
    from sglang.srt.model_executor.forward_batch_info import ForwardBatch

class RadixLinearAttention(nn.Module):
    """Linear Attention Layer"""

    # ... (__init__ 不变 ) ...

    def forward(self, forward_batch: ForwardBatch, mixed_qkv, a, b):
        if forward_batch.forward_mode.is_extend() and get_forward_context() is not None:
            seq_len = mixed_qkv.shape[0]
            output = torch.empty((1, seq_len, self.num_v_heads, self.head_v_dim),
                                 dtype=mixed_qkv.dtype, device=mixed_qkv.device)
            # 新增 BCG 分支：若当前处于可中断 CUDA Graph 中，
            # 则调用 eager_on_graph 包装的版本以避免被 Graph 捕获
            if is_in_breakable_cuda_graph():
                bcg_unified_linear_attention_with_output(
                    mixed_qkv, a, b, output, self.layer_id)
            else:
                unified_linear_attention_with_output(
                    mixed_qkv, a, b, output, self.layer_id)
            return output
```

```
        else:
            return get_attn_backend().forward(
                layer=self, forward_batch=forward_batch,
                mixed_qkv=mixed_qkv, a=a, b=b)


@register_custom_op(mutates_args=["output"])
@register_split_op()
def unified_linear_attention_with_output(
    mixed_qkv: torch.Tensor, a: torch.Tensor, b: torch.Tensor,
    output: torch.Tensor, layer_id: int,
) -> None:
    """线性注意力计算核心，保持不变"""
    context = get_forward_context()
    forward_batch = context.forward_batch
    attention_layer = context.attention_layers[layer_id]
    real_num_tokens = forward_batch.num_token_non_padded_cpu

    original_out_cache_loc = forward_batch.out_cache_loc
    forward_batch.out_cache_loc = original_out_cache_loc[:real_num_tokens]

    ret = get_attn_backend().forward(
        layer=attention_layer, forward_batch=forward_batch,
        mixed_qkv=mixed_qkv[:real_num_tokens],
        a=a[:real_num_tokens], b=b[:real_num_tokens],
    )
    forward_batch.out_cache_loc = original_out_cache_loc

    output[:, :real_num_tokens].copy_(ret)


# 包装一个 eager 模式版本，确保在 BCG 中此函数不会编译进 Graph
bcg_unified_linear_attention_with_output = eager_on_graph(True)(
    unified_linear_attention_with_output
)
```

评论区精华

无 review 评论。Oasis-Git 和 Qiaolin-Yu 均直接批准，表明该修复逻辑清晰、风险可控。

- 暂无高价值评论线程

风险与影响

- 风险：
  - 回归风险低：本 PR 只修改一个文件的 34 行代码，且逻辑是纯新增分支（在 BCG 激活时走新路径），非 BCG 场景行为完全不变。

    - 性能影响：BCG 模式下，线性注意力计算改为 *eager* 执行，可能略微增加 GPU kernel launch 开销，但 PR 提供的 GSM8K 数据显示吞吐量从 5578 tok/s 下降到 4122 tok/s（-26%），这主要是由于之前错误输出导致更高的“伪吞吐”。修复后准确率大幅提升，性能损失在可接受范围内。
    - 缺少测试覆盖：本次变更未附带单元测试或集成测试，依赖 CI 中的 GSM8K 基准测试验证。
- 影响：
    - 用户影响：所有使用 Qwen3.5 等混合线性注意力模型并启用 BCG 的用户将不再遇到静默错误输出，模型质量恢复至正常水平。
    - 系统影响：仅影响 SRT 推理引擎中的线性注意力层执行路径，无跨模块副作用。
    - 团队影响：提供了一种可复用的 BCG 接入模式（导入 is_in_breakable_cuda_graph + *eager_on_graph* 包装），未来其他注意力变体可参照此模式。
    - 风险标记：核心路径变更，缺少测试覆盖

关联脉络

- PR #22218 feat: implement BCG wiring for RadixAttention: 本 PR 是 #22218 在 RadixLinearAttention 上的镜像实现，提供了相同的 BCG 分支模式。
- PR #26085 drop FutureIndices wrapper class: 同为调度器 / BCG 相关重构，涉及 overlap_utils.py 和 scheduler.py，表明该区域正在进行活跃改造。