PR #28579 完整报告

sgl-project/sglang

[spec decoding] fully overlap spec decoding for hybrid linear attention backend

合并时间：2026-06-19 04:31

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/28579

执行摘要

- 一句话：混合线性注意力后端推测解码完全重叠
- 推荐动作：建议阅读以了解如何在 CUDA Graph 捕获与 replay 时高效传递 padding 信息。hybrid_linear_attn_backend.py 中的 _replay_metadata 函数修改展示了如何通过可选的 num_padding 参数避免冗余计算，这种模式可推广到其他后端。

功能与动机

推测解码中，draft 模型与 target 模型之间存在等待时间；通过使推理计算与通信重叠，可以减少延迟。PR 作者提到已与 @yhyang201 线下确认，表明该方案在混合线性注意力后端上有效。

实现拆解

1. 传递 padding 数量（num_padding）：在 decode_cuda_graph_runner.py 的 build_replay_fb_view 函数中新增 num_padding=bs - raw_bs，将 CUDA Graph 运行时推断出的 padding 请求数直接附加到 ForwardBatch 的代理命名空间中。
2. 修改注意力后端 _replay_metadata 接口：在两个注意力后端（hybrid_linear_attn_backend.py 与 ascend_hybrid_linear_attn_backend.py）的 _replay_metadata 方法中增加可选的 num_padding 参数。当此参数非 None 时，跳过基于 seq_lens_cpu 计算 padding 数量的逻辑，从而避免一次 CPU-GPU 同步。
3. 调用方适配：在 init_forward_metadata_out_graph 中，通过 getattr(forward_batch, "num_padding", None) 将 ForwardBatch 中的 num_padding 字段传入 _replay_metadata，而不是依赖原有的 seq_lens_cpu 推导。
4. 新增 needs_cpu_seq_lens 标记：在 gdn_backend.py 的 GDNAttnBackend 类中添加 needs_cpu_seq_lens: bool = False，表明 GDN 后端不再强制需要 seq_lens_cpu，为后续移除不必要的 CPU 张量提供基础。

关键文件：

- python/sglang/srt/layers/attention/hybrid_linear_attn_backend.py（模块 注意力后端；类别 source；类型 core-logic；符号 init_forward_metadata_out_graph, _replay_metadata, Mamba2AttnBackend）：核心修改：在 init_forward_metadata_out_graph 中传递 num_padding，并修改 _replay_metadata 支持可选参数，避免重复计算 padding。

- python/sglang/srt/model_executor/runner/decode_cuda_graph_runner.py（模块 CUDA Graph 运行器；类别 source；类型 data-contract；符号 build_replay_fb_view）：在 build_replay_fb_view 中新增 num_padding 字段，作为 ForwardBatch 的一部分传递给下游。
- python/sglang/srt/layers/attention/linear/gdn_backend.py（模块 GDN 后端；类别 source；类型 core-logic；符号 GDNAttnBackend）：为 GDNAttnBackend 添加 needs_cpu_seq_lens = False 标记，表明不再强制需要 CPU seq_lens，是 spec decode 完全重叠的配套改造。
- python/sglang/srt/hardware_backend/npu/attention/ascend_hybrid_linear_attn_backend.py（模块 NPU 后端；类别 source；类型 core-logic；符号 _replay_metadata）：同步修改 NPU 专属的 mixed linear attention 后端，使 _replay_metadata 也支持可选的 num_padding 参数，保持与 GPU 后端行为一致。

关键符号：init_forward_metadata_out_graph, _replay_metadata, build_replay_fb_view

关键源码片段

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

核心修改：在 init_forward_metadata_out_graph 中传递 num_padding，并修改 _replay_metadata 支持可选参数，避免重复计算 padding。

```
# python/sglang/srt/layers/attention/hybrid_linear_attn_backend.py

# 在 init_forward_metadata_out_graph 中传入 num_padding 参数
# 这样 replay 时可以直接拿到 padding 数，避免重新计算
def init_forward_metadata_out_graph(self, forward_batch: ForwardBatch, in_capture: bool = False):
    # seq_lens_cpu 在非 target-verify 时无用但保留兼容性
    self.forward_metadata = self._replay_metadata(
        forward_batch.batch_size,
        forward_batch.req_pool_indices,
        forward_batch.forward_mode,
        forward_batch.spec_info,
        forward_batch.seq_lens_cpu if not in_capture else None,
        # 关键：从 forward_batch 中获取提前算好的 padding 数
        num_padding=(0 if in_capture else getattr(forward_batch, "num_padding", None)),
    )

def _replay_metadata(
    self, bs, req_pool_indices, forward_mode, spec_info, seq_lens_cpu,
    num_padding: Optional[int] = None, # 新增可选参数
):
    # 如果外部提供了 num_padding，就直接使用；否则按原有逻辑从 seq_lens_cpu 算
    if num_padding is None:
        if seq_lens_cpu is None:
            num_padding = 0
        else:
```

```
            num_padding = torch.count_nonzero(
                seq_lens_cpu == self.get_cuda_graph_seq_len_fill_value()
            )
        # 用 num_padding 修正 req_pool_indices 和 mamba_indices 末尾
        req_pool_indices[bs - num_padding:] = 0
        mamba_indices = self.req_to_token_pool.get_mamba_indices(req_pool_indices)
        mamba_indices[bs - num_padding:] = -1
        # ... 后续逻辑不变
```

python/sglang/srt/model_executor/runner/decode_cuda_graph_runner.py

在 build_replay_fb_view 中新增 num_padding 字段，作为 ForwardBatch 的一部分传递给下游。

```
# python/sglang/srt/model_executor/runner/decode_cuda_graph_runner.py

# 构造 replay 时使用的 ForwardBatch-like 对象
def build_replay_fb_view(bs, raw_bs, num_tokens, forward_batch, buffers, capture_forward_
mode, seq_len_fill_value, ...):
    return SimpleNamespace(
        batch_size=bs,
        forward_mode=capture_forward_mode,
        actual_forward_mode=forward_batch.forward_mode,
        input_ids=buffers.input_ids[:num_tokens],
        req_pool_indices=buffers.req_pool_indices[:bs],
        seq_lens=buffers.seq_lens[:bs],
        seq_lens_cpu=buffers.seq_lens_cpu[:bs],
        # 新增：直接将 padding 个数传过去，避免后端再次推算
        num_padding=bs - raw_bs,
        encoder_lens=buffers.encoder_lens[:bs] if is_encoder_decoder else None,
        out_cache_loc=getattr(forward_batch, "out_cache_loc", None),
        spec_info=forward_batch.spec_info,
    )
```

python/sglang/srt/layers/attention/linear/gdn_backend.py

为 GDNAttnBackend 添加 needs_cpu_seq_lens = False 标记，表明不再强制需要 CPU seq_lens，是 spec decode 完全重叠的配套改造。

```
# python/sglang/srt/layers/attention/linear/gdn_backend.py

class GDNAttnBackend(MambaAttnBackendBase):
    """Attention backend for GDN (Gated Delta Network) linear attention."""

    # 明确表示该后端不需要 CPU 上的 seq_lens，
    # 从而在 CUDA Graph replay 时可以跳过相关同步
    needs_cpu_seq_lens: bool = False

    def __init__(self, model_runner: ModelRunner):
        super().__init__(model_runner)
        # ... 其余初始化不变
```

评论区精华

无公开 review 讨论；作者在 PR 评论中称已与 @yhyang201 线下确认。

- 线下确认 (other): 方案已获认可并合入。

风险与影响

- 风险：修改了核心 CUDA Graph 路径和注意力后端 replay 元数据构建逻辑：若 num_padding 传递不正确可能导致 padding 请求处理错误或性能退化；needs_cpu_seq_lens 标记的引入可能影响其他依赖此标记的逻辑；NPU 后端的修改需要验证兼容性。
- 影响：直接影响使用混合线性注意力（hybrid linear attention）后端的推测解码场景，如 Qwen 模型或类似架构。在 NPU（Ascend）上也有对应后端修改，可能影响 NPU 用户的推测解码性能。影响范围限定在 spec decode 开启且使用 hybrid linear attention 后端时，对其他注意力后端无影响。
- 风险标记：CUDA Graph 路径变更，padding 计算逻辑变更，NPU 后端需要验证

关联脉络

- PR #28559 fix: speculative draft worker clobbering target attention backend: 同属推测解码领域，修复 draft worker 覆盖 target 注意力后端的问题，与本次重叠优化有潜在的交互。