PR #26128 完整报告

sgl-project/sglang

[core] Make spec_v2 `seq_lens_cpu` optional via backend `needs_cpu_seq_lens`; Triton opts out

合并时间：2026-05-30 04:00

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26128

执行摘要

- 一句话：使 spec_v2 seq_lens_cpu 按后端标志可选，Triton 跳过 CPU 同步
- 推荐动作：值得精读，展示了通过 per-backend 标志优雅控制同步路径的设计方法；关注 FutureMap 的异步模式及多组件协调方式。

功能与动机

Triton 的 cuda-graph replay 从预分配缓冲区重建元数据，从不读取 seq_lens_cpu / seq_lens_sum，因此在全 Triton EAGLE3 路径上该同步是纯粹开销。

实现拆解

1. 后端标志定义：在 BaseAttentionBackend 添加 needs_cpu_seq_lens: bool = True，TritonAttnBackend 和 TritonMultiStepDraftBackend 重写为 False。
2. 决策函数：新增 decide_needs_cpu_seq_lens（overlap_utils.py），对所有后端标志取 OR，并在 TBO / piecewise CG 开启时强制返回 True（当前尚未适配无 CPU 镜像的路径）。
3. FutureMap 条件跳过：FutureMap.__init__ 接受 needs_cpu_seq_lens 参数；resolve_seq_lens_cpu 中当标志为 False 时保留 GPU 的 seq_lens 更新，但跳过 .cpu() D2H 和 sum 计算，将 seq_lens_cpu/seq_lens_sum 置为 None。
4. 下游适配：eagle_info_v2.py、forward_batch_info.py、cuda_graph_runner.py、schedule_batch.py 等多个文件增加 None 守卫，支持在缺少 CPU 镜像时走 GPU-only 路径；metrics_reporter.py 新增 _decode_total_seq_lens 函数，在无 CPU 镜像时退化为 sum(req.seqlen for req in batch.reqs)。

关键文件：

- python/sglang/srt/managers/overlap_utils.py（模块 调度同步；类别 source；类型 core-logic；符号 decide_needs_cpu_seq_lens, FutureMap.init, FutureMap.resolve_seq_lens_cpu）：核心决策与同步控制：新增 decide_needs_cpu_seq_lens 函数，修改 FutureMap 根据标志跳过 CPU D2H。
- python/sglang/srt/speculative/eagle_worker_v2.py（模块 推测解码；类别 source；类型 core-logic；符号 spec_v2_attn_backends, EAGLEWorkerV2.method）：关键消费者：暴露 spec_v2_attn_backends 属性，在 draft 方法中处理 seq_lens_sum 为 None 的 fallback。

- python/sglang/srt/managers/scheduler_components/metrics_reporter.py（模块 指标收集；类别 source；类型 core-logic；符号 _decode_total_seq_lens）：添加同步安全的 seq_lens 求和 fallback，避免在 CPU 镜像缺失时崩溃。
- python/sglang/srt/model_executor/forward_batch_info.py（模块 前向批处理；类别 source；类型 data-contract；符号 ForwardBatchInfo.init_new）：数据契约变更：ForwardBatchInfo.init_new 处理 extend_seq_lens 的两种路径（list 或 Tensor），并条件计算 seq_lens_sum。
- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 dependency-wiring）：依赖注入：在 init_overlap 中收集所有 backends 并调用 decide_needs_cpu_seq_lens，透传给 FutureMap。
- python/sglang/srt/layers/attention/triton_backend.py（模块 注意力后端；类别 source；类型 core-logic）：Triton 后端声明 needs_cpu_seq_lens=False，是本优化的启动点。
- python/sglang/srt/managers/schedule_batch.py（模块 批调度；类别 source；类型 core-logic）：批处理操作中增加 seq_lens_cpu 的 None 守卫。
- python/sglang/srt/speculative/eagle_info_v2.py（模块 推测信息；类别 source；类型 core-logic）：spec_info 中根据 seq_lens_cpu 存在与否选择 list 或 GPU tensor 路径。

关键符号：decide_needs_cpu_seq_lens, FutureMap.init, FutureMap.resolve_seq_lens_cpu, spec_v2_attn_backends, _decode_total_seq_lens, ForwardBatchInfo.init_new

关键源码片段

python/sglang/srt/managers/overlap_utils.py

核心决策与同步控制：新增 decide_needs_cpu_seq_lens 函数，修改 FutureMap 根据标志跳过 CPU D2H。

```
from typing import Sequence
from sglang.srt.layers.attention.base_attn_backend import AttentionBackend
from sglang.srt.server_args import ServerArgs

def decide_needs_cpu_seq_lens(
    server_args: "ServerArgs",
    attn_backends: Sequence["AttentionBackend"],
) -> bool:
    """
    聚合所有 attention 后端的 needs_cpu_seq_lens 标志，
    并强制 TBO / piecewise CG 路径返回 True（暂不支持无 CPU 镜像）。
    """
    if server_args.enable_two_batch_overlap:
        # FIXME: support TBO without seq lens cpu value
        return True
    if not server_args.disable_piecewise_cuda_graph:
        # FIXME: support PCG without seq lens cpu value
        return True
    return any(b.needs_cpu_seq_lens for b in attn_backends)
```

```
class FutureMap:

    def __init__(self, device, spec_algo, req_to_token_pool, needs_cpu_seq_lens=True):
        # ... 原有初始化 ...
        self.needs_cpu_seq_lens = needs_cpu_seq_lens
        # 根据标志决定是否分配 pinned buffer 和 d2h stream
        if _is_cuda and self.needs_cpu_seq_lens:
            self.new_seq_lens_cpu_pinned = torch.empty(...)
            self.fwd_prepare_d2h_stream = ...

    def resolve_seq_lens_cpu(self, batch: ScheduleBatch) -> None:
        # ... 等待 publish_ready，更新 GPU seq_lens ...
        if not self.needs_cpu_seq_lens:
            # 保留 GPU gather，跳过 D2H 和 sum，下游走 GPU-only 路径
            batch.seq_lens_cpu = None
            batch.seq_lens_sum = None
            return
        # 原有 D2H 路径：pinned stream 或 fallback .cpu()
```

python/sglang/srt/speculative/eagle_worker_v2.py

关键消费者：暴露 spec_v2_attn_backends 属性，在 draft 方法中处理 seq_lens_sum 为 None 的 fallback。

```
# eagle_worker_v2.py (partial)

@property
def spec_v2_attn_backends(self) -> tuple:
    """
    本 worker 涉及的所有 attention 后端，供调度器决策是否保留 seq_lens_cpu。
    """
    return (
        self._target_worker.model_runner.attn_backend,
        self._draft_worker.draft_attn_backend,
        self._draft_worker.draft_extend_attn_backend,
    )

# 在 draft 方法中处理 seq_lens_sum 可能为 None 的情况：
seq_lens_sum = batch.seq_lens_sum
if seq_lens_sum is None:
    if tree_mask_buf is None:
        # tree_mask 非预分配，使用最大静态长度作为上界
        max_context_len = self.target_worker.model_runner.attn_backend.max_context_len
        seq_lens_sum = batch.seq_lens.shape[0] * max_context_len
    else:
        # 预分配 buffer，kernel 会忽略 seq_lens_sum
        seq_lens_sum = 0
```

评论区精华

PR 描述中明确 TBO 和 piecewise CG 场景仍需 CPU seq_lens，目前强制同步并标记为 FIXME；Triton 后端选择退出后，下游多处需增加 None 守卫，增加了分支复杂性，但架构清晰。

- TBO / piecewise CG 是否应强制同步 (design): 保留强制同步，待后续优化再移除。

风险与影响

- 风险：若新增的后端未正确设置 needs_cpu_seq_lens，或下游调用未处理 seq_lens_cpu 为 None 的情况，可能导致 AttributeError 或静默错误。TBO/piecewise CG 路径当前强制同步，但未来若移除需大量适配。缺少针对性测试，回归依赖现有 CI。
- 影响：使用纯 Triton 后端的 EAGLE3 用户受益于减少同步延迟；其他后端和配置无功能变化。对开发者：新增 attention 后端时需正确设置标志，否则可能引入不必要的同步或 bug。
- 风险标记：缺少测试覆盖，核心调度同步变更，条件分支复杂化

关联脉络

- PR #23452 prototype no-verify-sync approach: 本 PR 在其上生产化，将实验性方案转为 per-backend 标志驱动的正式路径。