PR #25879 完整报告

sgl-project/sglang

[Spec] Route seq_lens through FutureMap; drop verify_done.wait

合并时间：2026-05-21 16:51

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25879

执行摘要

- 一句话：路由 seq_lens 通过 FutureMap，消除 verify_done 等待
- 推荐动作：值得精读以理解 speculative decoding v2 的同步设计演变。该 PR 展示了如何用 FutureMap 的发布 / stash 模式替代显式跨流事件，是一种值得借鉴的解耦调度流与前向流数据传递的方法。建议关注 future 缓冲区生命周期管理。

功能与动机

PR 描述中明确说明目标："Drop verify_done.wait() cross-stream barrier — route seq_lens through FutureMap so schedule-stream consumers gate on a forward-stream publish_ready event instead"。原有机制在 verify 完成后需要显式事件同步，增加了调度流阻塞时间，通过 FutureMap 栅栏可以让调度流更早地开始下一轮准备，提升 overlap 效率。

实现拆解

1. FutureMap 重构（overlap_utils.py）：将 store_to_map 拆分为 publish（写入 schedule 消费的 new_seq_lens_buf，并记录 publish_ready 事件）和 stash（写入 forward-only 字段，如 topk/bonus/hidden）。新增 resolve_seq_lens_cpu 方法供调度流在栅栏后 D2H 获取 new_seq_lens，并更新 batch.seq_lens_cpu 和 seq_lens_sum。
2. 调度器 run_batch 调整（scheduler.py）：在 _overlap_forward_isolation 之前，对于 spec v2 batch 调用 resolve_seq_lens_cpu 替代原来的 refresh_seq_lens_cpu。通过 functools.partial 将 publish 绑定到 future_indices 作为回调传递给 worker。在 forward 返回后，使用 stash 写入 forward-only 字段，并将 batch.seq_lens 设为 -future_indices.indices 作为占位符。
3. Worker 触发回调（eagle_worker_v2.py, multi_layer_eagle_worker_v2.py）：两个 worker 的 forward_batch_generation 方法新增 on_verify_complete 参数，在 target 前向后（prefill 分支）或 verify 完成后（decode 分支）调用 on_verify_complete(new_seq_lens)，触发 publish 写入缓冲区并记录事件。
4. 清理 ScheduleBatch（schedule_batch.py）：移除了 maybe_wait_verify_done 和 refresh_seq_lens_cpu 方法，相关的调用点内联或替换。filter_batch 和 merge_batch 中不再调用 maybe_wait_verify_done。
5. 简化数据结构（eagle_info_v2.py, eagle_info.py）：从 EagleDraftInput 中移除 verify_done 字段，在 verify 方法中不再创建和记录事件。

prepare_for_extend_to_fill_draft_kvcache 中直接计算 seq_lens_sum 替代调用 refresh_seq_lens_cpu。

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 _lazy_init_buf, _lazy_init_forward_buf, store_to_map, resolve_seq_lens_cpu）：核心数据结构 FutureMap 重构，新增 publish/stash/resolve_seq_lens_cpu 方法，拆分缓冲区写入策略，引入 publish_ready 栅栏事件。是 PR 的核心变更。
- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 dependency-wiring）：调度器 run_batch 中集成新的同步流程：预取 seq_lens_cpu、传递 on_verify_complete 回调、使用 stash 写入 forward-only 字段。体现了新机制的调用入口。
- python/sglang/srt/managers/schedule_batch.py（模块 批次管理；类别 source；类型 core-logic；符号 maybe_wait_verify_done, refresh_seq_lens_cpu）：移除了 maybe_wait_verify_done 和 refresh_seq_lens_cpu 方法，简化批次同步逻辑。
- python/sglang/srt/speculative/eagle_worker_v2.py（模块 推测执行；类别 source；类型 core-logic；符号 forward_batch_generation）：EagleWorkerV2 的 forward_batch_generation 接受 on_verify_complete 回调，在 verify 完成后触发 publish，是新同步机制的关键触发点。
- python/sglang/srt/speculative/multi_layer_eagle_worker_v2.py（模块 推测执行；类别 source；类型 core-logic；符号 forward_batch_generation）：多层 Eagle worker 同样的改动，确保一致性。
- python/sglang/srt/speculative/eagle_info_v2.py（模块 推测执行；类别 source；类型 core-logic）：移除 verify_done 字段引用，简化 prepare_for_extend_to_fill_draft_kvcache 中的 seq_lens_sum 计算。
- python/sglang/srt/speculative/eagle_info.py（模块 推测执行；类别 source；类型 core-logic）：移除验证的基类中 verify_done 字段（减法 1 行）。
- python/sglang/srt/model_executor/forward_batch_info.py（模块 前向信息；类别 source；类型 data-contract）：小调整（1 行变更），适应 seq_lens_sum 计算时机变更。
- python/sglang/srt/disaggregation/decode_schedule_batch_mixin.py（模块 拆分配置；类别 source；类型 core-logic）：适配移除了 maybe_wait_verify_done 的调用。

关键符号：FutureMap.publish, FutureMap.stash, FutureMap.resolve_seq_lens_cpu, ScheduleBatch.maybe_wait_verify_done（已移除）, ScheduleBatch.refresh_seq_lens_cpu（已移除）, EagleWorkerV2.forward_batch_generation, MultiLayerEagleWorkerV2.forward_batch_generation

关键源码片段

python/sglang/srt/managers/overlap_utils.py

核心数据结构 FutureMap 重构，新增 publish/stash/resolve_seq_lens_cpu 方法，拆分缓冲区写入策略，引入 publish_ready 栅栏事件。是 PR 的核心变更。

```
# python/sglang/srt/managers/overlap_utils.py (head 版本关键片段 )
import torch
from typing import Optional

class FutureMap:
    def __init__(self, device, spec_algo, req_to_token_pool):
        self.device = device
        self.spec_algo = spec_algo
        self.req_pool_size = req_to_token_pool.req_to_token.shape[0]
        if self.spec_algo.is_none():
            # 非 speculative 模式仅需 token_ids_buf，直接分配
            self.token_ids_buf = torch.empty(
                (self.req_pool_size,), dtype=torch.int64, device=self.device
            )
        else:
            # Schedule 消费的缓冲区（new_seq_lens_buf）提前分配固定 dtype
            self.new_seq_lens_buf = torch.empty(
                (self.req_pool_size,), dtype=torch.int64, device=self.device
            )
            # Forward-only 字段的缓冲区延迟初始化（依赖 worker 的具体 shape）
            self._forward_buf_initialized = False
        # 栅栏事件：publish 时记录，schedule 流在 resolve_seq_lens_cpu 中等待
        self.publish_ready: Optional[torch.cuda.Event] = None

    def _lazy_init_forward_buf(self, draft_input: EagleDraftInput):
        # 仅在首次前向时初始化 forward-only 缓冲区
        self._forward_buf_initialized = True
        topk_p0 = draft_input.topk_p[0]
        topk_index0 = draft_input.topk_index[0]
        self.topk_p_buf = torch.empty(
            (self.req_pool_size, *topk_p0.shape),
            dtype=topk_p0.dtype, device=self.device,
        )
        self.topk_index_buf = torch.empty(
            (self.req_pool_size, *topk_index0.shape),
            dtype=topk_index0.dtype, device=self.device,
        )
        self.bonus_tokens_buf = torch.empty(
            (self.req_pool_size,), dtype=torch.int64, device=self.device
        )
        # 以下依配置决定是否分配 hidden_states_buf
        if spec_need_hidden_states():
            hidden_states0 = draft_input.hidden_states[0]
            self.hidden_states_buf = torch.empty(
                (self.req_pool_size, *hidden_states0.shape),
                dtype=hidden_states0.dtype, device=self.device,
            )

    def publish(self, future_indices: FutureIndices, new_seq_lens: torch.Tensor):
```

```
        # 由 worker 的 on_verify_complete 回调触发，在 forward 流中执行
        self.new_seq_lens_buf[future_indices.indices] = new_seq_lens
        # 记录当前流事件，作为 schedule 流读取的栅栏
        self.publish_ready = torch.get_device_module(self.device).Event()
        self.publish_ready.record()

    def stash(self, future_indices: FutureIndices, payload):
        # 写入 forward-only 字段，仅被同一流后续 resolve_future 读取
        # 不需要额外栅栏，因为在同一 CUDA 流内自然有序
        indices = future_indices.indices
        self.topk_p_buf[indices] = payload.topk_p
        self.topk_index_buf[indices] = payload.topk_index
        self.bonus_tokens_buf[indices] = payload.bonus_tokens
        if spec_need_hidden_states() and hasattr(payload, 'hidden_states'):
            self.hidden_states_buf[indices] = payload.hidden_states

    def resolve_seq_lens_cpu(self, batch: ScheduleBatch) -> None:
        # 调度流调用：等待 publish_ready 事件，然后 D2H 拷贝 new_seq_lens
        fi = batch.spec_info.future_indices if batch.spec_info is not None else None
        if fi is None:
            return
        if self.publish_ready is not None:
            self.publish_ready.wait() # 流级等待（不阻塞 CPU）
        # 从 GPU 缓冲区拷贝到 CPU 张量（触发隐式同步）
        batch.seq_lens_cpu = self.new_seq_lens_buf[fi.indices.cpu()].cpu()
        batch.seq_lens_sum = int(batch.seq_lens_cpu.sum())
```

python/sglang/srt/managers/scheduler.py

调度器 run_batch 中集成新的同步流程：预取 seq_lens_cpu、传递 on_verify_complete 回调、使用 stash 写入 forward-only 字段。体现了新机制的调用入口。

```
# python/sglang/srt/managers/scheduler.py (head 版本 run_batch 相关片段 )
from functools import partial

# ... 在 run_batch 方法中，当 enable_overlap 且 is_generation 时：
if self.enable_overlap:
    # Spec v2 预取：在 forward isolation 之前从 FutureMap 获取 seq_lens_cpu
    if batch.is_spec_v2:
        self.future_map.resolve_seq_lens_cpu(batch)

    with self._overlap_forward_isolation(batch):
        future_indices = FutureIndices(indices=batch.req_pool_indices)

        # 如果是 spec v2，构造回调：forward 流在 verify 完成后会调用 publish
        fwd_kwargs = (
            {
                "on_verify_complete": partial(
                    self.future_map.publish, future_indices
                )
```

```
            }
            if batch.is_spec_v2
            else {}
        )

        with self.forward_stream_ctx:
            self.forward_stream.wait_stream(self.schedule_stream)
            self.future_map.resolve_future(batch) # 从缓冲区读取上一轮结果
            batch_result = self.model_worker.forward_batch_generation(
                batch, **fwd_kwargs
            )
            # ... ( 处理 extra_keep_alive_refs 等 )

            # 传递 forward-only 字段（非 spec v2 时传递 next_token_ids）
            stash_payload = (
                batch_result.next_draft_input
                if batch.is_spec_v2
                else batch_result.next_token_ids
            )
            self.future_map.stash(future_indices, stash_payload)
            batch_result.copy_to_cpu(...)

        # 设置下一轮的占位符：seq_lens 设为负数的 future_indices
        if batch.is_spec_v2:
            batch.seq_lens = -future_indices.indices # 调度流中作为 sentinel
        # ...
```

python/sglang/srt/managers/schedule_batch.py

移除了 maybe_wait_verify_done 和 refresh_seq_lens_cpu 方法，简化批次同步逻辑。

```
# python/sglang/srt/managers/schedule_batch.py (head 版本，关键删除部分 )
# 原来存在的方法已被删除（patch 中为 - 行）：
# def maybe_wait_verify_done(self):
# if self.is_spec_v2:
# draft_input: EagleDraftInput = self.spec_info
# if draft_input.verify_done is not None:
# draft_input.verify_done.wait()
#
# def refresh_seq_lens_cpu(self, sync: bool = True):
# if sync and self.is_spec_v2:
# self.seq_lens_cpu = self.seq_lens.cpu()
# self.seq_lens_sum = int(self.seq_lens_cpu.sum())

# 在 prepare_for_decode 中：
# 原为 self.seq_lens_sum = None 的注释更新
# 原在 filter_batch 开头的 self.maybe_wait_verify_done() 调用被移除

# 在 merge_batch 中：
# 原调用 self.maybe_wait_verify_done() 被移除
```

评论区精华

PR 未收到人工审查评论，仅有一个自动代码审查机器人（gemini-code-assist）的总结性评论，无实质性讨论。

- 暂无高价值评论线程

风险与影响

- 风险：核心路径变更：speculative decoding v2 的同步机制完全重写，若 publish_ready 事件或缓冲区索引错误可能导致调度流读取失效 seq_lens，引发推理错误或 crash。现有测试可能覆盖不足（本次未新增测试），依赖回归测试套件。另外，new_seq_lens_buf 使用 int64 固定 dtype，若未来 seq_lens 类型变化需同步修改。调度器与 worker 之间的回调耦合紧密，后续扩展可能增加复杂度。
- 影响：影响范围限于启用 speculative decoding 且使用 overlap 模式的用户（即 spec v2 + enable_overlap）。预期减少调度流阻塞，提升吞吐量。不影响非 speculative 模式或非 overlap 模式。代码重构简化了同步逻辑，降低了维护成本。
- 风险标记：核心路径变更，缺少测试覆盖

关联脉络

- PR #25824 [Refactor] Encapsulate SWA loc translation inside SWAKVPool with per-batch cache invalidation: 同作者（hnyls2002）同一 speculative decoding 优化系列，涉及调度同步改进。