PR #33477 完整报告

sgl-project/sglang

[srt] Reuse batched Mamba boundary mask

合并时间：2026-08-09 16:47

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33477

执行摘要

- 一句话：复用 Mamba 边界 mask，decode 结果处理提速约 18%
- 推荐动作：值得精读。核心看点有三：一是如何在不改变前向语义的前提下，用“预计算 + lookahead 快照选择”消除热路径上的重复计算；二是面对 #31369 被回滚的历史，如何用 kv_committed_len 权威化 + 断言 + 模拟 overlap 调度器的单元测试重建正确性信心；三是 ScheduleBatch.copy() 只快照 reqs 列表而 Req 对象共享，导致 decode_batch_idx 被下一轮推进这一隐蔽时序问题的处理方式。建议重点读 _handle_finish_state_updated_req 的 mask 选择分支，以及 prepare_for_decode 中 mamba_track_mask_next_cpu 仅随 enable_overlap 生成的条件。

功能与动机

PR body 明确指出：ScheduleBatch.prepare_for_decode 已经用一次向量化 CPU 操作算出 Mamba track 边界 mask 并拷贝到设备，而 decode 结果处理仍逐请求重算同一边界条件（含函数调用与簿记开销）。早期优化版本 #31369 因确定性 Qwen3-Next decode-cache-hit KL 回归（avg_kl_div=0.0065 > 0.002）被 #31622 回滚，本 PR 直接针对该失败模式修复：快照当前与一步前瞻 mask，按每请求 decode-batch 推进量选择，并把 kv_committed_len 记为权威提交边界。

实现拆解

1. 在 ScheduleBatch 上扩展 host 端 Mamba 元数据（python/sglang/srt/managers/schedule_batch.py）：新增 mamba_track_mask_cpu、mamba_track_mask_next_cpu、mamba_decode_batch_idx_cpu 三个字段，并在 filter_batch、merge_batch 中置 None、在 copy() 中一并传递，保证批次过滤、合并、快照后字段语义不残留。
2. 在 prepare_for_decode 中一次性生成两组 mask 与计数器快照：非 spec 路径下用 seq_lens_cpu % mamba_track_interval 一次向量化求出当前轮 mask 与一步前瞻 mask（前瞻 mask 仅在 enable_overlap=True 时生成），同时按 prepare 时刻快照每个请求的 req.decode_batch_idx；spec decode 路径显式清空三个字段，保持既有流程不变。异步 H2D 的 mamba_track_mask 改为从已算好的 CPU mask 派生，前向语义不变。
3. 在 decode 结果处理中按 lookahead 选择 mask（python/sglang/srt/managers/scheduler_components/batch_result_processor.py）：_handle_finish_state_updated_req 通过 req.decode_batch_idx - batch.mamba_decode_batch_idx_cpu[i] 计算 lookahead（断言

必须为 0 或 1），据此选用 mamba_track_mask_cpu 或 mamba_track_mask_next_cpu；只有确认到达边界的请求才调用 _mamba_prefix_cache_update，并传 known_boundary=True。

4. 权威边界与兜底校验：_mamba_prefix_cache_update 在 known_boundary=True 时直接以 req.kv_committed_len 作为 track 长度并断言其为 interval 整数倍；新增静态方法 _mamba_assert_committed_len_lookahead，校验 kv_committed_len 与 token 序列长度偏差不超过 1，作为对调度器前瞻窗口的诊断保护；回退路径 _mamba_check_track_boundary 也补上了同一断言。
5. 测试与验证：新增 test/registered/unit/managers/test_batch_result_processor_mamba_boundary.py，用 fake Scheduler 驱动 event_loop_overlap()，分别覆盖 lookahead=0 与 lookahead=1 两种时序，断言 mask 选择结果与 known_boundary 传递；test_qwen3_next_models.py 的 KL 回归测试在 CI 上通过（含一次 flaky 失败后的 rerun）。

关键文件：

- python/sglang/srt/managers/scheduler_components/batch_result_processor.py（模块 结果处理；类别 source；类型 core-logic；符号 _handle_finish_state_updated_req, _mamba_prefix_cache_update, _mamba_assert_committed_len_lookahead, _mamba_check_track_boundary）：核心逻辑所在：decode 结果处理按 lookahead 选择复用 mask，并通过 known_boundary 短路逐请求重算，是本次优化与正确性修复的主战场。
- python/sglang/srt/managers/schedule_batch.py（模块 调度批次；类别 source；类型 core-logic；符号 prepare_for_decode, filter_batch, merge_batch, copy）：数据结构与准备阶段：在 prepare_for_decode 中一次性生成 host 端边界 mask 与计数器快照，并维护 filter/merge/copy 生命周期，是复用的数据来源。
- test/registered/unit/managers/test_batch_result_processor_mamba_boundary.py（模块 单元测试；类别 test；类型 test-coverage；符号 _make_batch, _make_processor, _make_result, TestMambaBoundaryMaskReuse）：新增的 CPU 单元测试用 fake Scheduler 驱动 event_loop_overlap，覆盖 lookahead=0 与 lookahead=1 两种时序，是验证 overlap 时序正确性的关键配套。

关键符号：_handle_finish_state_updated_req, _mamba_prefix_cache_update, _mamba_assert_committed_len_lookahead, _mamba_check_track_boundary, prepare_for_decode, test_overlap_scheduler_handles_zero_and_one_batch_lookahead

关键源码片段

python/sglang/srt/managers/scheduler_components/batch_result_processor.py

核心逻辑所在：decode 结果处理按 lookahead 选择复用 mask，并通过 known_boundary 短路逐请求重算，是本次优化与正确性修复的主战场。

```
# ============================================================
# SchedulerBatchResultProcessor decode 结果处理：复用边界 mask
# ============================================================
```

```
# 背景：ScheduleBatch.prepare_for_decode 已用一次向量化取余算好
# Mamba track 边界 mask（设备端 + 主机端），decode 结果处理不必
# 再逐请求重算边界条件，只需按 lookahead 选出正确的 mask。
#
# 关键时序：ScheduleBatch.copy() 只快照 reqs 列表，Req 对象仍然
# 共享；overlap 模式下处理本 batch 结果时，req.decode_batch_idx
# 可能已被下一轮 decode 推进，因此必须用 prepare 时刻的快照
# mamba_decode_batch_idx_cpu 做差得到 lookahead。

def _handle_finish_state_updated_req(
    self, req, batch, result, i, logits_output
):
    known_mamba_boundary = None
    if batch.mamba_track_mask_cpu is not None:
        lookahead = req.decode_batch_idx - batch.mamba_decode_batch_idx_cpu[i]
        # overlap 最多只允许提前一个 decode batch，超出即调度模型失效
        assert lookahead in (0, 1), (
            'mamba result lookahead={lookahead} for req {req.rid}; '
            'overlap advanced more than one decode batch'
        )
        if lookahead == 0:
            # 结果对应本 batch 的 forward：用当前轮 mask
            known_mamba_boundary = bool(batch.mamba_track_mask_cpu[i])
        else:
            # 结果对应已被 overlap 预跑的下一轮：用一步前瞻 mask
            known_mamba_boundary = bool(batch.mamba_track_mask_next_cpu[i])

    # known_mamba_boundary 为 None 表示该路径未启用（spec decode /
    # 未开 extra buffer），走旧的逐请求重算；否则只有真正到达边界
    # 的请求才进入前缀缓存更新，并把 known_boundary 传给更新函数。
    if known_mamba_boundary is None or known_mamba_boundary:
        self._mamba_prefix_cache_update(
            req, batch, result, i,
            known_boundary=known_mamba_boundary is True,
        )
    # ... 后续 finish 状态、KV 释放、统计等处理保持不变 ...


def _mamba_prefix_cache_update(
    self, req, batch, result, i, known_boundary: bool = False
) -> None:
    """更新 Mamba track 的 ping-pong 状态（仅在边界处有实际动作）。"""
    if req.mamba_ping_pong_track_buffer is None:
        return

    lazy = get_server_args().enable_mamba_extra_buffer_lazy()
    if known_boundary:
        # 已知边界：直接用 kv_committed_len 作为权威 track 长度，
        # 避免再次调用 _mamba_check_track_boundary 做逐请求重算。
```

```
            self._mamba_assert_committed_len_lookahead(req)
            track_seqlen = req.kv_committed_len
            # 边界长度必须是 interval 的整数倍（page-aligned）
            assert track_seqlen % get_exec().mamba.mamba_track_interval == 0
            at_boundary = True
        else:
            # 回退路径（spec decode 或 mask 不可用）保持原逻辑
            at_boundary, track_seqlen = self._mamba_check_track_boundary(
                req, batch, result, i
            )

        if lazy and not batch.spec_algorithm.is_none():
            self._mamba_lazy_spec_update(req, batch, i, at_boundary, track_seqlen)
            return

        if not at_boundary:
            return

        req.mamba_last_track_seqlen = track_seqlen
        if lazy:
            self.mamba_lazy_post_decode_at_boundary(req, batch)
        else:
            req.mamba_next_track_idx = (
                batch.req_to_token_pool.get_mamba_ping_pong_other_idx(
                    req.mamba_next_track_idx
                )
            )
```

python/sglang/srt/managers/schedule_batch.py

数据结构与准备阶段：在 prepare_for_decode 中一次性生成 host 端边界 mask 与计数器快照，并维护 filter/merge/copy 生命周期，是复用的数据来源。

```
# ============================================================
# ScheduleBatch.prepare_for_decode：一次性预计算边界 mask 并保留 host 副本
# ============================================================
# 原本只生成 device 端 mamba_track_mask 供前向使用；现在额外保留
# 三个 host 端字段，供结果处理阶段（batch_result_processor）复用，
# 从而避免每个请求在 decode 结果处理时重算一次边界条件。

if server_args.enable_mamba_extra_buffer():
    mamba_track_interval = get_exec().mamba.mamba_track_interval

    if len(self.reqs) == 0:
        self.mamba_track_indices = torch.empty(
            (0,), dtype=torch.int64, device=self.device
        )
    else:
        if server_args.enable_mamba_extra_buffer_lazy():
            self.mamba_lazy_prealloc_at_boundary(mamba_track_interval)
```

```
        set_mamba_track_indices_from_reqs(self)

    # 一次向量化取余得到当前轮 mask，以及下一轮（overlap 前瞻）mask
    track_remainders_cpu = self.seq_lens_cpu % mamba_track_interval
    track_mask_cpu = track_remainders_cpu == 0
    self.mamba_track_mask_cpu = track_mask_cpu.tolist()
    self.mamba_track_mask_next_cpu = (
        (track_remainders_cpu == mamba_track_interval - 1).tolist()
        if self.enable_overlap
        else None
    )
    # ScheduleBatch.copy() 只快照 reqs 列表，Req 对象仍共享；下一轮
    # overlap decode 可能已经推进 req.decode_batch_idx，所以这里按
    # prepare 时刻给每个请求打一个计数器快照，供结果处理区分
    # lookahead = 0（本 batch）还是 lookahead = 1（下一轮）。
    self.mamba_decode_batch_idx_cpu = [
        req.decode_batch_idx for req in self.reqs
    ]
    # 异步 H2D 保持不变，只是从已算好的 CPU mask 派生
    self.mamba_track_mask = track_mask_cpu.pin_memory().to(
        device=self.device, non_blocking=True
    )
```

test/registered/unit/managers/test_batch_result_processor_mamba_boundary.py

新增的 CPU 单元测试用 fake Scheduler 驱动 event_loop_overlap，覆盖 lookahead=0 与 lookahead=1 两种时序，是验证 overlap 时序正确性的关键配套。

```
# ============================================================
# 单元测试：overlap 调度下 Mamba 边界 mask 复用的正确性
# ============================================================
# 通过 fake Scheduler 驱动 event_loop_overlap()，在
# get_next_batch_to_run 中按 plan_count 控制是否额外准备一个
# decode batch，从而分别制造 lookahead = 0 与 lookahead = 1 的时序，
# 并断言结果处理阶段选到了正确的 mask / known_boundary。

def test_overlap_scheduler_handles_zero_and_one_batch_lookahead(self):
    for schedule_next_decode, expected_lookahead in ((False, 0), (True, 1)):
        with self.subTest(schedule_next_decode=schedule_next_decode):
            req, batch = _make_batch()
            processor = _make_processor()
            result = _make_result()
            # ... fake Scheduler 构造（gracefully_exit、request_receiver、
            # run_batch 等 MagicMock）省略 ...
            plan_count = 0

            def get_next_batch_to_run(*, running_batch, last_batch):
                nonlocal plan_count
                del running_batch, last_batch
```

```
    plan_count += 1
    if plan_count == 1:
        batch.prepare_for_decode()
        return SimpleNamespace(
            running_batch=batch, batch_to_run=batch
        )
    if plan_count == 2 and schedule_next_decode:
        # 第二次 prepare_for_decode：制造 lookahead = 1
        batch.prepare_for_decode()
        return SimpleNamespace(
            running_batch=batch, batch_to_run=batch
        )
    return SimpleNamespace(
        running_batch=batch, batch_to_run=None
    )

scheduler.get_next_batch_to_run = get_next_batch_to_run
observed_lookahead = []

def process_batch_result(result_batch, batch_result):
    # 记录真实观察到的 lookahead，校验内部选择逻辑
    observed_lookahead.append(
        req.decode_batch_idx
        - result_batch.mamba_decode_batch_idx_cpu[0]
    )
    processor.process_batch_result_decode(
        result_batch, batch_result
    )

scheduler.process_batch_result = process_batch_result
# ... 依赖 patch（alloc_for_decode、get_server_args、
# get_exec、pin_memory 等）与 event_loop_overlap() 执行省略 ...

self.assertEqual(observed_lookahead, [expected_lookahead])
if expected_lookahead == 0:
    # 未到边界（mask 为 False）：不应触发任何前缀缓存更新
    cache_update.assert_not_called()
else:
    self.assertTrue(
        cache_update.call_args.kwargs['known_boundary']
    )
```

评论区精华

维护者 ispobock 的 review 评论是唯一一条正式技术意见，核心提醒是：复用 seq_lens_cpu mask 切换自 kv_committed_len，这与 #31369 是相同的基础变更，而 #31622 曾因确定性 Qwen3-Next decode-cache-hit KL 回归（avg_kl_div=0.0065 > 0.002）回滚。作者通过 lookahead 快照 + kv_committed_len 权威化 + 断言 + Qwen3-Next KL 测试验证该失败模式

已解决，ispobock 最终 APPROVED（无附加意见）。CI 过程中 test_qwen3_next_models.py 首次运行失败，作者回应"seems flaky, i ran it locally, was able to pass"，ispobock 两次 rerun 后通过，并最终 /rerun-failed-ci 收尾，合入前 cc 了 Mamba 相关维护者 hanming-lu。另有 GitHub 自动评论提示 Gemini Code Assist 服务已停用，不影响本 PR。

- 复用 seq_lens_cpu mask 与 #31369/#31622 历史回归的关系 (correctness): 作者通过 lookahead 快照 + kv_committed_len 权威化 + 断言保护 + Qwen3-Next KL 测试验证解决了该失败模式；ispobock 最终 APPROVED（无附加意见）。
- Qwen3-Next e2e 测试 flaky 与 rerun (testing): 判定为 flaky，rerun 后通过，PR 合入。
- fork PR 的 /rerun-test 权限限制 (other): 由维护者执行 rerun，流程正常。

风险与影响

- 风险：
  1. 核心 decode 热路径变更：改动位于每个 decode 步都会执行的 _handle_finish_state_updated_req，所有非 spec 的 Mamba 混合架构模型（典型 Qwen3-Next）都会经过，回归影响面大。
  2. overlap 时序敏感：mask 来自 seq_lens_cpu，权威边界却是 kv_committed_len，二者在 overlap 下可能差一步，依赖 mamba_track_mask_next_cpu 兜底；req.decode_batch_idx 被下一轮推进的时序一旦超过 1，assert lookahead in (0, 1) 会直接 fail-fast，属于显式保护，但未来扩展 overlap 宽度必须同步修改此处。
  3. 历史回归领域：Qwen3-Next decode-cache-hit KL 回归正是 #31369 被回滚的原因，本 PR 虽然用 kv_committed_len 权威化并新增断言，但 CI 中 test_qwen3_next_models.py 出现过一次 flaky 失败，需持续关注该测试稳定性。
  4. mask 生命周期：filter_batch / merge_batch 后新字段被置 None，结果处理会静默回退到旧的逐请求重算路径（_mamba_check_track_boundary），行为正确但失去优化，若批次在 filter 后仍被结果处理引用且 mask 未恢复，可能掩盖性能退化问题。
  5. 内存开销：每个 decode batch 多保留两个 bool 列表与一个 int 列表（各 [b]），batch 规模大时 CPU 内存略有增加，相对 token 缓存可忽略。 - 影响：影响所有启用 enable_mamba_extra_buffer 的非 spec Mamba 混合架构模型（典型 Qwen3-Next）的 decode 结果处理：1000 请求规模下 process_batch_result_decode 平均耗时降约 10.8%，单步 CPU 基准降约 18%（每步约省 0.35 ms），在 CPU 调度成为瓶颈的高并发场景收益明显。对团队而言，ScheduleBatch 新增的 host 端 mask 与计数器快照为后续 Mamba 边界处理提供了可复用模式，避免再次逐请求重算；同时 spec decode 路径显式置 None，不受影响。 - 风险标记：核心 decode 热路径变更，历史 KL 回归领域，overlap 时序断言 fail-fast，CI 测试曾 flaky

关联脉络

- PR #31369 （上下文未提供标题，早期同类优化）：PR body 说明这是本优化思路的早期版本，因确定性 Qwen3-Next decode-cache-hit KL 回归被回滚；本 PR 是它的修正实现，改用 kv_committed_len 权威边界并处理 overlap lookahead。

- PR #31622 （上下文未提供标题，回滚 #31369）：PR body 说明它回滚了 #31369（avg_kl_div=0.0065 > 0.002 的 KL 回归），本 PR 专门针对该失败模式修复并补充验证。