PR #34043 完整报告

sgl-project/sglang

[srt] Fix sconv state memory corruption on specdec

合并时间：2026-08-09 21:42

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34043

执行摘要

- 一句话：修复投机解码下 sconv 状态内存破坏，缓存复用后位级一致
- 推荐动作：值得精读。核心亮点是 CUDA Graph 捕获与 Python 控制流 specialize 的交互处理：捕获时若 if 分支跳过 kernel，该 kernel 永远不会出现在图中，本 PR 用"全零惰性字段 + 强制在图上保留 scatter"的方法解决，是通用且可迁移的设计。另一个值得借鉴的点是把静态缓冲 padding 清零作为 populate_from_forward_batch 的数据契约写入，从源头保证图回放位级可复现。建议后续补充回归测试，让这类修复可被 CI 捕获。

功能与动机

PR body 明确说明：DFlash/DSpark speculative pools 在启用 radix cache 时破坏 sconv（short-conv）状态，影响如 Inkling、Inkling-Small 一类 hybrid-SWA 模型。根因是 mamba tracking 被 gate 在 prefill-graph capture 上，任何 radix cache 命中的请求都会以全零 sconv cache 张量继续推理，长上下文场景尤其严重。团队找到 70 题 SWEBench（Verified/Pro/Multilingual）子集，DFlash 将 pass rate 从 55% 拉到 35%，修复目标是与无投机模式达成位级一致的 parity。

实现拆解

本变更沿三个层次展开：

1. 解除投机门控：prefill_cuda_graph_runner.py 的 _is_mamba_track_enabled 删除 spec_algorithm.is_none() 条件，使开启 enable_mamba_extra_buffer 且未禁用 radix cache 的 DFlash/DSpark 请求也能进入 Mamba 状态追踪路径。这是修复的入口——此前投机模式下 tracking 被整体关闭，导致 sconv 状态无从恢复。
2. 图捕获惰性化：inkling_sconv_backend.py 在 _alloc_graph_buffers 中新增 _graph_track_inert_mask、_graph_track_inert_indices、_graph_track_inert_seqlens 三个全零缓冲；_refresh_track_conv_indices 在 mamba_track_mask is None 且处于图路径时，用这些惰性缓冲替换 forward_batch 字段，确保捕获 warmup 批次也会 launch track scatter kernel，避免 Python 层 if 分支把 scatter 从 captured graph 中 specialize 掉；非图路径则显式清空 track_conv_indices 防止脏数据残留。
3. 静态缓冲 padding 清零：buffers.py 的 populate_from_forward_batch 重构 mamba_track_* 复制逻辑：只要静态缓冲存在，无论 forward_batch 是否携带字段，都保证 [bs:] padding 行清零；mask 缺失时 [:bs] 也清零。这样 CUDA Graph 回放时 gather

读到的任何行都不会引用越界或陈旧的 token 索引。

4. 配套处理：ispobock 追加 "fix lint" 提交（dcc087ac）满足审阅要求，随后合并 main。本次未附带新增测试文件，验证依赖手工 A/B 位级一致检查和 70 题 SWEBench 基准。

关键文件：

- python/sglang/srt/layers/attention/linear/inkling_sconv_backend.py（模块 卷积后端；类别 source；类型 core-logic；符号 _alloc_graph_buffers, _refresh_track_conv_indices）：核心修复点：图捕获时用惰性缓冲替换 forward_batch 字段，强制 track scatter 留在 captured graph 内；非图路径显式清空元数据。这是解决 "radix cache 命中即全零 sconv 状态 " 的关键逻辑。
- python/sglang/srt/model_executor/runner_utils/buffers.py（模块 静态缓冲；类别 source；类型 data-contract；符号 populate_from_forward_batch）：静态输入缓冲的 padding 清零逻辑被重构为无条件契约，保证 CUDA Graph 回放时 gather 读到的任何行都不会引用脏索引。
- python/sglang/srt/model_executor/runner/prefill_cuda_graph_runner.py（模块 图捕获；类别 source；类型 core-logic；符号 _is_mamba_track_enabled）：修复入口：移除 mamba tracking 对投机算法的门控，使 DFlash/DSpark 下也启用状态追踪。

关键符号：_is_mamba_track_enabled, _alloc_graph_buffers, _refresh_track_conv_indices, populate_from_forward_batch

关键源码片段

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

核心修复点：图捕获时用惰性缓冲替换 forward_batch 字段，强制 track scatter 留在 captured graph 内；非图路径显式清空元数据。这是解决 "radix cache 命中即全零 sconv 状态 " 的关键逻辑。

以下片段来自 _refresh_track_conv_indices 的入口分支（head 版本整理），展示修复的核心处理：

```
# 若当前批次没有 tracking 元数据，则分两条路径处理。
if forward_batch.mamba_track_mask is None:
    if not on_graph_path:
        # 非图路径：显式清空元数据，避免下游读到上一次的陈旧索引。
        self.sconv_metadata.track_conv_indices = None
        return
    # 图路径：CUDA Graph 会把 Python 的 if 分支固化为“恒不执行”，
    # 因此捕获热身批次即使没有 tracking 元数据，也必须用惰性缓冲占位，
    # 强制 track scatter kernel 被捕获进图内。
    # 全零 mask 会屏蔽所有行，scatter 实际执行但无副作用。
    rows = forward_batch.batch_size
    forward_batch.mamba_track_mask = self._graph_track_inert_mask[:rows]
    forward_batch.mamba_track_indices = self._graph_track_inert_indices[:rows]
    forward_batch.mamba_track_seqlens = self._graph_track_inert_seqlens[:rows]

rows = forward_batch.batch_size
```

```
query_start_loc = self.sconv_metadata.query_start_loc
# 只对真实存活的请求计算窗口位置，避免越界读取。
live = min(
    rows,
    forward_batch.mamba_track_seqlens.shape[0],
    forward_batch.extend_prefix_lens.shape[0],
)
# 追踪范围：从最后一个完整 chunk 边界回退一个 conv 核尺寸，得到窗口起点。
lens_to_track = (
    forward_batch.mamba_track_seqlens[:live]
    - forward_batch.extend_prefix_lens[:live]
)
chunk_aligned = (
    lens_to_track // self.mamba_cache_chunk_size
) * self.mamba_cache_chunk_size
start_indices = query_start_loc[:live] + chunk_aligned - self.conv_state_len

# 后续将 start_indices 扩展为 (live, conv_state_len) 的索引矩阵，
# 经 torch.add 与 gather/scatter 写入 _graph_track_conv_indices。
```

python/sglang/srt/model_executor/runner_utils/buffers.py

静态输入缓冲的 padding 清零逻辑被重构为无条件契约，保证 CUDA Graph 回放时 gather 读到的任何行都不会引用脏索引。

以下片段来自 populate_from_forward_batch 中 mamba_track_* 静态缓冲的填充逻辑（head 版本整理）：

```
# 旧逻辑只在 forward_batch 携带字段时才复制；新逻辑把"清零"作为数据契约：
# 只要静态缓冲存在，padding 行 [bs:] 必须清零，mask 缺失时 [:bs] 也清零，
# 保证 CUDA Graph 回放读取任何行都不会命中上一次捕获留下的脏索引。
if self.mamba_track_indices is not None:
    if forward_batch.mamba_track_indices is not None:
        self.mamba_track_indices[:bs].copy_(forward_batch.mamba_track_indices)
    self.mamba_track_indices[bs:].zero_()
if self.mamba_track_mask is not None:
    if forward_batch.mamba_track_mask is not None:
        self.mamba_track_mask[:bs].copy_(forward_batch.mamba_track_mask)
    else:
        self.mamba_track_mask[:bs].zero_()
    self.mamba_track_mask[bs:].zero_()
if self.mamba_track_seqlens is not None:
    if forward_batch.mamba_track_seqlens is not None:
        self.mamba_track_seqlens[:bs].copy_(forward_batch.mamba_track_seqlens)
    self.mamba_track_seqlens[bs:].zero_()
```

评论区精华

PR 的 review comments 为空，直接技术争论较少，但有两条 issue 评论值得记录：

- Qiaolin-Yu 在 issue 评论中要求 “could you fix the lint”，由 ispobock 以 commit dcc087ac（"fix lint"）解决，随后审阅者 APPROVE。
- ekzhang、ispobock 各执行一次 /tag-and-rerun-ci 重跑 CI，说明修复经过多轮 CI 验证。

核心设计权衡（图捕获必须保留 scatter kernel、padding 必须清零）主要由 PR body 与代码注释呈现，没有形成评审争辩。

- 修复 lint (style): 已解决：lint 修复被并入 PR，审阅者 APPROVE。
- CI 重跑 (other): 无技术结论，最终 PR 合并。

风险与影响

- 风险：
  1. 回归面：buffers.py 的控制流从“字段存在才复制”变为“buffer 存在就完成复制 + 清零”，所有消费 mamba_track_* 的路径行为都被加强；若未来其它后端依赖旧语义（缺失字段不清零），可能产生静默差异。
  2. CUDA Graph 行为变更：inkling_sconv_backend.py 依赖惰性缓冲让 scatter 在图捕获时被保留，这要求 inert mask 全零时 scatter 确实无副作用；该假设若在后续 kernel 改动中被破坏，会导致图内状态写坏。
  3. 开销：_is_mamba_track_enabled 放开后，投机解码 + radix cache 组合的 prefill 会多做一次 conv 状态快照追踪，引入少量 kernel launch 与显存占用（3 个 max_bs 量级惰性缓冲）。PR body 显示 token/turn 计数与 nospec 一致，开销可接受。
  4. 测试缺口：本次没有新增自动化测试，位级一致性依赖手工 A/B 与外部 SWEBench 基准，后续修改缺少防御网。 - 影响：用户影响：DFlash/DSpark 投机 + radix cache + Inkling 类混合 SWA 模型的长上下文用户。修复前会出现推理“失忆”、输出卡死、基准分（如 SWEBench pass rate 55% → 35%）骤降；修复后缓存命中与全新前缀位级一致，投机与无投机结果对齐。系统影响：prefill CUDA Graph 捕获路径与 populate_from_forward_batch 数据契约发生变化，属于 srt 调度核心路径，但改动集中在 3 个文件、改动量 36+/20-，风险可控。团队影响：为 sglang 的确定性推理积累了一个可复用的图捕获“惰性输入”模式，后续维护者可直接借用。
- 风险标记：核心路径变更，缺少测试覆盖，图捕获行为变更，位级一致性依赖

关联脉络

- PR #34168 Add deterministic logprob-consistency test for inkling-small nvfp4: 同为 Inkling 模型，且以确定性 logprob 一致性为目标，可与本 PR 的位级一致验证互相印证。
- PR #32402 Switch inkling per-commit test to nvfp4: Inkling per-commit 精度测试基线，本 PR 修复的正是这类模型在投机模式下的输出退化问题。
- PR #34159 Fix deterministic inference all-reduce for tp>1: 与本次修复同属 sglang 确定性推理输出可复现的投入方向，且都涉及投机 / 并行场景。