PR #26510 完整报告

sgl-project/sglang

Fix _GenerationStreamAccumulator logprob_end off-by-one under retract

合并时间：2026-08-21 07:03

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26510

执行摘要

- 一句话：修复 retract 下 logprob 游标 off-by-one 错位
- 推荐动作：值得精读。核心看点：一是修复刻意限定在 retract 场景并保留既有 prefill-only 行为，是“最小回归面”的典型范例；二是 review 中关于“如何干净地区分两个场景”的讨论展示了行为兼容的设计权衡；三是通过 SGLANG_TEST_RETRACT=True 确定性触发时序 bug 的 E2E 写法，值得在类似竞态 / 时序问题中复用。

功能与动机

PR 解决的是一个时序敏感的游标错位问题：overlap scheduling 下请求在 prefill forward 在途时被 retract，process_batch_result_prefill 中 if req.is_retracted: continue 跳过该请求的 token 追加，但请求仍会进入输出流。非流式 return_logprob=True 请求因 0 % DEFAULT_FORCE_STREAM_INTERVAL == 0 触发强制输出，max(len(output_ids_), 1) 将 send_output_token_logprobs_offset 推进到 1 而 send_token_offset 保持 0，之后每次输出都少带一个 logprob，最终 len(output_token_logprobs) == len(output_ids) - 1。PR body 明确报告了复现数据：修复前连续 5 次试验 2/32 响应不匹配，修复后连续 8 次试验 0/32。

实现拆解

1. 根因定位：在 python/sglang/srt/managers/scheduler_components/output_streamer.py 的 _GenerationStreamAccumulator.accept() 中发现 logprob_end = max(len(output_ids_), 1) 对“无输出 token”的请求不区分原因——无论是被 retract 还是合法的 prefill-only，都统一套用下限，导致 retracted 请求的 logprob 游标被错误前置。
2. 核心修复：logprob_end 改为 len(output_ids_) if req.is_retracted else max(len(output_ids_), 1)。retracted 请求保持 token 与 logprob 两个游标对齐在 0，避免后续错位；非 retract 请求行为完全不变。该改动只影响 return_logprob=True 且输出为空的分支，token 流与增量 detokenize 逻辑不受影响，性能开销仅一个布尔分支。
3. CPU 单元测试：新增 test/registered/unit/managers/test_output_streamer_logprobs.py，用 _FakeReq 桩直接驱动 accept()，两条用例分别断言：retracted 空输出请求两个游标均保持 0、非 retract 的 max_new_tokens=0 请求 logprob 游标推进到 1 且保留首个 logprob。注册 CPU CI（base-a-test-cpu）。
4. GPU/AMD 端到端回归：新增 test/registered/scheduler/test_retract_decode_logprob.py，通过 SGLANG_TEST_RETRACT=True 强制每两步 retract 一次来确定性复现时序 bug，以 32 个并发非流式 return_logprob=True 请求断言输出 token 与 logprob 数量 1:1 对

齐；注册 CUDA base-b 与 AMD stage-b-test-1-gpu-small-amd。

5. 验证与 CI：/rerun-test 在 1-gpu-5090 上单独跑通 E2E 测试；pre-commit 与 Python 语法编译均通过；提交历史包含 4 次 main 合并，最终由 Qiaolin-Yu 合入。

关键文件：

- python/sglang/srt/managers/scheduler_components/output_streamer.py（模块 输出流；类别 source；类型 core-logic；符号 _GenerationStreamAccumulator, accept）：核心修复文件：_GenerationStreamAccumulator.accept() 的 logprob_end 计算增加 req.is_retracted 分支，是本次 bug 的唯一源码改动点。
- test/registered/unit/managers/test_output_streamer_logprobs.py（模块 输出流；类别 test；类型 test-coverage；符号 _FakeReq, _make_accumulator, TestOutputStreamerLogprobs, test_retracted_empty_output_does_not_advance_logprob_offset）：新增 CPU 单元测试，用 _FakeReq 桩直连 accept()，两条用例分别锁定 retracted 空输出与 prefill-only 两种游标行为，是防回归的关键保护层。
- test/registered/scheduler/test_retract_decode_logprob.py（模块 抢占调度；类别 test；类型 test-coverage；符号 TestRetractDecodeLogprob, setUpClass, tearDownClass, _one_request）：新增 GPU/AMD 端到端回归测试，通过 SGLANG_TEST_RETRACT=True 确定性复现 retract 时序，验证 32 并发请求下 token 与 logprob 数量 1:1 对齐，注册 CUDA base-b 与 AMD stage-b。

关键符号：_GenerationStreamAccumulator.accept, TestOutputStreamerLogprobs.test_retracted_empty_output_does_not_advance_logprob_offset, TestOutputStreamerLogprobs.test_prefill_only_request_preserves_first_logprob, TestRetractDecodeLogprob.test_output_logprobs_aligned_under_test_retract

关键源码片段

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

核心修复文件：_GenerationStreamAccumulator.accept() 的 logprob_end 计算增加 req.is_retracted 分支，是本次 bug 的唯一源码改动点。

```
# sglang/srt/managers/scheduler_components/output_streamer.py
# _GenerationStreamAccumulator.accept() 中 return_logprob 分支的核心改动

if req.return_logprob:
    # 修复前这里无条件使用 max(len(output_ids_), 1)：
    # 对 retracted 且 output_ids 为空的请求，logprob 游标会被推进到 1，
    # 而 send_token_offset 仍为 0，导致后续每个输出 token 少携带一个 logprob。
    # 对非 retract 的 prefill-only 请求（max_new_tokens=0），该下限保证仍能
    # 返回首个 logprob，这是当初引入 max(..., 1) 的用途，必须保留。
    logprob_end = (
        len(output_ids_) if req.is_retracted else max(len(output_ids_), 1)
    )
    # 六种 logprob 结构（值 / 下标 /top 序列 /token ids 序列）统一按
    # [send_output_token_logprobs_offset:logprob_end] 切片收集，
    # 随后把游标推进到 logprob_end，保证与 send_token_offset 对齐。
```

```
        self.output_token_logprobs_val.append(
            req.logprob.output_token_logprobs_val[
                send_output_token_logprobs_offset:logprob_end
            ]
        )
        self.output_token_logprobs_idx.append(
            req.logprob.output_token_logprobs_idx[
                send_output_token_logprobs_offset:logprob_end
            ]
        )
        self.output_top_logprobs_val.append(
            req.logprob.output_top_logprobs_val[
                send_output_token_logprobs_offset:logprob_end
            ]
        )
        self.output_top_logprobs_idx.append(
            req.logprob.output_top_logprobs_idx[
                send_output_token_logprobs_offset:logprob_end
            ]
        )
        self.output_token_ids_logprobs_val.append(
            req.logprob.output_token_ids_logprobs_val[
                send_output_token_logprobs_offset:logprob_end
            ]
        )
        self.output_token_ids_logprobs_idx.append(
            req.logprob.output_token_ids_logprobs_idx[
                send_output_token_logprobs_offset:logprob_end
            ]
        )
        req.send_output_token_logprobs_offset = logprob_end
```

test/registered/unit/managers/test_output_streamer_logprobs.py

新增 CPU 单元测试，用 _FakeReq 桩直连 accept()，两条用例分别锁定 retracted 空输出与 prefill-only 两种游标行为，是防回归的关键保护层。

```
# test/registered/unit/managers/test_output_streamer_logprobs.py
# 用桩对象直接驱动 _GenerationStreamAccumulator.accept()，在 CPU 上锁死两条分支行为

class TestOutputStreamerLogprobs(unittest.TestCase):
    def test_retracted_empty_output_does_not_advance_logprob_offset(self):
        # retracted 且 output_ids 为空的请求：token 游标与 logprob 游标都必须停在 0，
        # accumulator 产出空 logprob，确保后续输出不再错位。
        req = _FakeReq(is_retracted=True, max_new_tokens=16)
        accumulator = _make_accumulator()

        accumulator.accept(req=req)

        self.assertEqual(req.send_token_offset, 0)
```

```
        self.assertEqual(req.send_output_token_logprobs_offset, 0)
        self.assertEqual(accumulator.output_token_logprobs_val, [[]])

    def test_prefill_only_request_preserves_first_logprob(self):
        # 非 retract 的 prefill-only 请求（max_new_tokens=0）仍需返回首个 logprob，
        # 这是 max(len(output_ids_), 1) 下限存在的意义，修复后行为保持不变。
        req = _FakeReq(is_retracted=False, max_new_tokens=0)
        accumulator = _make_accumulator()

        accumulator.accept(req=req)

        self.assertEqual(req.send_token_offset, 0)
        self.assertEqual(req.send_output_token_logprobs_offset, 1)
        self.assertEqual(accumulator.output_token_logprobs_val, [[-0.5]])
```

评论区精华

核心交锋发生在初版修复（直接改为 `logprob_end = len(output_ids_)`）之后：Qiaolin-Yu 作为 `max(..., 1)` 的原始引入者指出该下限是为非 retract 的 prefill-only（`max_new_tokens=0`，如 recsys 场景）请求返回首个 logprob 而加，担心破坏该行为；shenxiul 先是追问“Prefill-only as in pd disagg?”，确认场景后承认“that sounds wrong... Is there any suggestion of how we can cleanly distinguish?”；Qiaolin-Yu 最终建议“could we check if `req.is_retracted` is True here?”，并据此落地。最终实现既修复了 bug，又通过 `req.is_retracted` 判别把行为变更面收敛到最小，两条单元测试分别锁死两个方向，所有疑虑均已解决。

- 初版修复是否破坏 prefill-only（max_new_tokens=0）请求的首 logprob 行为 (design): 采纳 Qiaolin-Yu 的建议，用 req.is_retracted 区分两种场景：retracted 请求用 len(output_ids_)，其余保留 max(len(output_ids_), 1)；新增两条单元测试分别锁死两分支。
- 回归测试的确定性触发与 CI 注册 (testing): 确定性触发 + 双平台 CI 注册；修复前后对比数据（2/32 → 0/32）验证有效性。

风险与影响

- 风险：
  - is_retracted 标志生命周期：修复依赖该标志在 retract 路径上的可靠性。若未来有路径复用 Req 对象且未重置标志，prefill-only 请求可能丢失首个 logprob（切片结果为空）。当前 retract 请求随后进入重新调度或 abort 流程，风险较低，但值得在后续改动中留意。
  - 游标对齐易回归：accept() 中 output_ids 收集、force-stream 条件、切片终点三处逻辑耦合，任何一处调整都可能再次引入 token/logprob 错位；已有 CPU 单元测试与 E2E 回归形成双重保护，但单元测试依赖 _FakeReq 桩的属性完整性。
  - CI 成本：E2E 测试为 CUDA base-b（约 300 秒）与 AMD stage-b（约 360 秒）新增真实服务启动与 32 并发请求的回归时长，属可接受增量。
  - 性能：核心改动仅一个布尔分支，无性能影响。
- 影响：

- 用户侧：非流式 return_logprob=True 请求在 retract 场景下 logprob 与 token 数量恢复 1:1，下游依赖 token 级打分的管道（如 RLHF 数据采集、逐 token 评估）不再收到错位数据。
- 系统侧：核心改动仅 4 行，风险面收敛；新增 2 个测试文件均为回归性质，不改变线上行为面。
- 团队侧：确立“CPU 单元桩测 + GPU E2E 强制触发”的双层回归模式，对后续 retract 路径的时序类 bug 修复有直接借鉴价值。
- 风险标记：依赖 is_retracted 标志生命周期，游标对齐逻辑易回归，E2E 测试新增 CI 耗时约 300-360 秒

关联脉络

- PR #35412 [Fix] Land the decode mamba checkpoint depth on the tree page under DCP: 与本 PR 同属 retract 边界场景的调度器修复，涉及 batch_result_processor.py（process_batch_result_prefill 所在文件）与 schedule_batch.py 等相同模块，与本 PR 的根因路径直接相关。
- PR #35622 [misc] Trim restating comments and docstrings in srt/managers: 同仓库 srt/managers 模块近期注释清理与稳定化工作，涉及 schedule_batch.py 等与本 PR 输出流相邻的调度器组件，反映该区域正在持续加固。