PR #26230 完整报告

sgl-project/sglang

[Test] test_session_latency: assert streaming tail/head stability

合并时间：2026-05-25 05:06

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26230

执行摘要

- 一句话：Streaming session 延迟测试改用 tail/head 稳定性断言
- 推荐动作：值得参考其测试设计思路：当 baseline 不稳定时，改测内在属性作为断言。变更简单明确，建议直接合并。

功能与动机

在最近的 scheduled 运行中，regular_session 的 tail_avg 持续下降（190ms -> 155ms），而 streaming_session 保持平稳（~130ms），导致加速比跌破 1.4x 阈值并连续 3 次触发 CI 失败。此问题并非 streaming 变慢，而是 baseline 追赶上来。因此需要一种对 baseline 波动不敏感的断言，旨在测试 streaming 会话本身在上下文增长时延迟是否稳定（streaming 跨轮复用 KV）。

实现拆解

1. 在测试文件头部注释中更新测试描述，将原来的三个测试改为：稳定性（streaming only）、正确性（regular vs streaming 输出一致）、随机长度（无延迟断言）。
2. 引入 HEAD_TURNS = 10 常量，用于取前 10 个 turn（跳过 prefill 后的第一个 turn）作为 head 窗口。
3. 修改 _collect_latencies 函数，添加 first_n 参数以支持取前 N turn，与已有的 last_n 对称；调整控制流，确保 last_n 优先、其次 first_n、否则跳过 turn 1。
4. 移除全局摘要打印函数 _print_summary 和 test_regular_session 测试方法，同时删除 TurnResult 中的 prompt_tokens 和 completion_tokens 字段（从未被读取）。
5. 在稳定性测试（原 test_streaming_session_stability 或替代方法）中，分别通过 _collect_latencies 取 head_avg 和 tail_avg，断言比值 <= 1.15。
6. 清理注释和未使用的导入（如 Dict）。

关键文件：

- test/registered/sessions/test_session_latency.py（模块 会话测试；类别 test；类型 test-coverage；符号 _collect_latencies, _print_summary, test_regular_session）：唯一变更文件，包含测试断言重构、函数移除和清理，是 PR 的核心。

关键符号：_collect_latencies, _print_summary, test_regular_session

关键源码片段

test/registered/sessions/test_session_latency.py

唯一变更文件，包含测试断言重构、函数移除和清理，是 PR 的核心。

```
# 从给定的 ModeResult 列表中收集延迟，支持取最后 N turn 或前 N turn。
# 注意：默认会跳过第一个 turn（包含 prefill）。
def _collect_latencies(
    results: List[ModeResult],
    last_n: Optional[int] = None,
    first_n: Optional[int] = None,
) -> List[float]:
    lats = []
    for r in results:
        # 优先使用 last_n（尾部窗口）
        if last_n is not None:
            turns = r.turns[-last_n:]
        elif first_n is not None:
            # 跳过第一个 prefill turn，然后取接下来 first_n 个 turn
            turns = r.turns[1 : 1 + first_n]
        else:
            # 默认跳过第一个 turn
            turns = r.turns[1:]
        lats.extend(t.client_latency_ms for t in turns)
    return lats

# 在稳定性测试中使用（以下为伪代码示意，实际位于 BenchSessionLatency 类中）
def test_streaming_session_stability(self):
    results = self._run_benchmark(mode='streaming')
    tail_lats = _collect_latencies(results, last_n=TAIL_TURNS)
    head_lats = _collect_latencies(results, first_n=HEAD_TURNS)
    tail_avg = sum(tail_lats) / len(tail_lats)
    head_avg = sum(head_lats) / len(head_lats)
    ratio = tail_avg / head_avg
    self.assertLessEqual(ratio, 1.15,
        f"Streaming latency stability exceeded: tail_avg={tail_avg:.2f}, head_avg={head_avg:.2f},
        ratio={ratio:.3f}")
```

评论区精华

仅 gemini-code-assist[bot] 自动评论，建议确保 tail 采样时排除第一个 turn，并添加显式检查是否成功收集延迟数据。该建议已在实现中覆盖。

- 暂无高价值评论线程

风险与影响

- 风险：风险极低，仅涉及测试文件，不改变生产代码。唯一潜在风险是 1.15 阈值在极端负载下仍可能 flaky，但已有 15% 缓冲且 CI 实测 ratio=1.00 验证了合理性。

- 影响：对 CI 测试：从比较两个模式的绝对值改为测试 streaming 自身稳定性，减少了因 baseline 变化引起的假阳性，提高可靠性。对开发者：移除了维护 test_regular_session 和 _print_summary 的负担。对用户：无影响。
- 风险标记：测试逻辑调整，低风险

关联脉络

- 暂无明显关联 PR