PR #1861 完整报告

THUDM/slime

fix: harden retool rollout against multi-turn / retry desync

合并时间：2026-05-11 19:17

原文链接：http://prhub.com.cn/THUDM/slime/pull/1861

执行摘要

- 一句话：加固 retool 多轮 rollout，防止样本状态不同步
- 推荐动作：建议使用 retool 的团队立即合并此 PR，并关注 ABORTED 样本率的监控指标。该 PR 的设计思路（在关键点维护数据不变量并优先安全中止而非静默错误）值得在其他自定义 rollout 生成函数中借鉴。

功能与动机

在多轮工具调用 rollout 中，sample.rollout_log_probs、loss_masks、response_token_ids 和 response 必须保持相同长度；任何不同步都会导致训练端 slice_log_prob_with_cp 出现令人困惑的长度不匹配错误。实际场景中，异步 retool 训练中样本通过重试路径（aborted → re-enqueued）以及工具输出超出上下文限制时，会触发四种不同的脱钩情况。该 PR 针对这四种情况进行修复，以保障 rollout 过程的稳健性。

实现拆解

1. 重置陈旧样本状态：在 generate 函数入口（examples/retool/generate_with_retool.py），将 sample.rollout_log_probs、sample.response、sample.response_length、sample.loss_mask 全部清空。重试样本（之前被 abort 或 partial）会带着旧的 rollout 状态到达，如果不清理直接追加新 token，会导致长度不匹配。
2. 限制每轮 max_new_tokens：将 max_context_length 的计算提前到循环外。在每轮循环中，计算剩余预算 remaining_budget = max_context_length - total_length，然后设置 per_turn_sampling_params['max_new_tokens'] = min(采样参数中的 max_new_tokens, remaining_budget)。这样单轮生成不会超过剩余上下文容量，避免生成过大样本导致训练端 partition/batch 代码崩溃（assert 或 OOM）。
3. 缺失 logprobs 时直接中止：当 sglang 返回文本但没有 output_token_logprobs 时（原代码会尝试重新 tokenize 并构造 logprobs），现在改为直接返回 ABORTED 状态，让 rollout manager 重新排队该样本。这避免了 token_id 和 logprobs 长度不同步导致的训练崩溃。
4. 修剪工具输出溢出：在每个 turn 处理之后，检查 total_length >= max_context_length，如果超限，则按比例截断 response_token_ids、loss_masks、rollout_log_probs（取前 max_context_length - len(prompt_tokens_ids) 个 token），重新解码 response 使文本匹配，并将样本状态设为 TRUNCATED。这确保工具输出（如大 print）不会导致长度不同步。

关键文件：

- examples/retool/generate_with_retool.py（模块 工具调用示例；类别 source；类型 core-logic；符号 generate）：该文件实现了 retool 工具调用滚动的自定义生成函数，是本 PR 唯一修改的文件，包含了全部四个修复逻辑。

关键符号：generate

关键源码片段

examples/retool/generate_with_retool.py

该文件实现了 retool 工具调用滚动的自定义生成函数，是本 PR 唯一修改的文件，包含了全部四个修复逻辑。

```
async def generate(args, sample: Sample, sampling_params) -> Sample:
    '''Custom generation function supporting tool calls'''
    # 修复 1：重置重试样本的陈旧状态
    sample.rollout_log_probs = None
    sample.response = ''
    sample.response_length = 0
    sample.loss_mask = None

    state = GenerateState(args)
    url = f'http://{args.sglang_router_ip}:{args.sglang_router_port}/generate'

    tool_specs = tool_registry.get_tool_specs()
    prompt = format_conversation_with_tools(prompt=sample.prompt, tools=tool_specs)
    prompt_tokens_ids = state.tokenizer(prompt, add_special_tokens=False)['input_ids']
    response = ''
    response_token_ids = []
    loss_masks = []

    # 提前计算 max_context_length（修复 2 的一部分）
    if args.rollout_max_context_len is not None:
        max_context_length = args.rollout_max_context_len
    else:
        max_context_length = args.context_parallel_size * args.max_tokens_per_gpu

    for turn in range(TOOL_CONFIGS['max_turns']):
        total_length = len(prompt_tokens_ids) + len(response_token_ids)
        if total_length >= max_context_length:
            sample.status = Sample.Status.TRUNCATED
            break

        # 修复 2：限制每轮生成的最大 token 数不超过剩余预算
        remaining_budget = max_context_length - total_length
        per_turn_sampling_params = dict(sampling_params)
        per_turn_sampling_params['max_new_tokens'] = min(
            sampling_params.get('max_new_tokens', remaining_budget),
```

```
        remaining_budget,
    )

    current_token_ids = prompt_tokens_ids + response_token_ids
    payload = {
        'input_ids': current_token_ids,
        'sampling_params': per_turn_sampling_params,
        'return_logprob': True,
    }

    output = await post(url, payload)

    if output['meta_info']['finish_reason']['type'] == 'abort':
        sample.status = Sample.Status.ABORTED
        return sample

    if 'output_token_logprobs' in output['meta_info']:
        cur_response_token_ids = [item[1] for item in output['meta_info']['output_token_
        logprobs']]
        cur_response = state.tokenizer.decode(cur_response_token_ids)
        cur_log_probs = [item[0] for item in output['meta_info']['output_token_logprobs']]
        if sample.rollout_log_probs is None:
            sample.rollout_log_probs = []
        sample.rollout_log_probs += cur_log_probs
    else:
        # 修复 3：当 sglang 返回文本但没有 logprobs 时，直接中止而非 fallback
        sample.status = Sample.Status.ABORTED
        return sample
    # 后续处理（循环末尾有修剪逻辑，此处省略）
```

评论区精华

本 PR 没有收到 review 评论。但作者在 PR body 和 commit message 中详细阐述了每个修复的动机和潜在行为变化，并强调了修复 #3（当 sglang 缺少 logprobs 时从 fallback 改为 abort）是值得注意的行为变化，操作员应关注 ABORTED 样本数是否有上升。

- 暂无高价值评论线程

风险与影响

- 风险：该 PR 的风险主要体现在以下方面：
  - 行为变化：修复 #3 将 sglang 缺少 logprobs 时的行为从静默 fallback（会导致训练崩溃但难以定位）改为立即 abort 并重试。这可能导致告警增加，但更易于诊断问题。
  - 影响范围：代码位于 *examples/retool/* 路径，仅当用户通过 *--custom-generate-function-path* 选择该生成函数时才会生效，核心框架不受影响。
  - 测试覆盖：无新增单元测试，*examples/retool/* 目前缺少自动测试，手动测试可复现每个故障模式。

- 影响：影响范围限定于使用 retool 功能并配置自定义生成函数的训练任务。对系统稳定性的影响正面：多轮工具调用中的 desync 崩溃应不再出现。在运营层面，用户应当监控 ABORTED 样本率，若异常升高则检查 sglang 的 return_logprob 配置。该 PR 为后续 examples/retool 的更多优化和测试覆盖奠定了基础。
- 风险标记：缺少测试覆盖，行为变化

关联脉络

- PR #1862 chore: include length context in slice_log_prob_with_cp assert: 该 PR 增强了 slice_log_prob_with_cp 的断言信息，与本 PR 修复的 desync 问题直接相关；两者配合使用可提高调试体验。