

# PR #2272 完整报告

THUDM/slime

feat: support streaming external rollouts

合并时间: 2026-09-03 14:42

原文链接: <http://prhub.com.cn/THUDM/slime/pull/2272>

## 执行摘要

- 一句话: 新增 SGLang 外部 rollout 的流式响应支持与请求级中止机制。
- 推荐动作: 该 PR 值得精读。它不仅仅是添加了一个新功能, 更重要的是重新设计了 rollout 中止的生命周期管理, 将其从单一的服务器级中止细分为请求级和服务级两种模式, 提高了系统的灵活性和资源效率。SGLangStreamAccumulator 的设计清晰地分离了流处理逻辑, 是处理复杂流式数据的好范例。测试的全面性 (26 个参数化用例 + 端到端验证) 也值得学习。建议关注请求级中止的实现模式以及 `replace_call_state` 标志如何处理完整的 top-p 元数据快照。

## 功能与动机

解决非流式 rollout 路径在中断时无法保留已生成前缀的问题。正如 PR body 所述: 'The non-streaming rollout path receives generation state only after /generate completes. If a rollout is interrupted early, it cannot retain the generated prefix without a separate polling or abort-response API. Streaming exposes that prefix as it is generated.' 流式处理允许 Slime 直接应用每个观察到的块到 `Sample`, 并在中途取消请求时保留观察到的前缀。此外, 对于长上下文场景, 增量流 (只处理新后缀) 可以避免重复拼接或重新扫描累积的响应文本, 提升效率。

## 实现拆解

1. 新增流处理核心模块(`slime/rollout/streaming_utils.py`): 引入 `SGLangStreamAccumulator` 类, 负责规范化 SGLang 的两种流格式 (累积 / 增量)。该类验证每个块的 `output_token_logprobs_length` 是否与配置的模式一致, 递增累积 token 和 logprobs, 并处理文本、top-p 元数据及路由专家数据。关键符号包括 `_has_full_response_top_p_metadata`、`SGLangStreamUpdate` 和 `add` 方法。
2. 重构中止生命周期(`slime/rollout/slang_rollout.py`): 将生成调用与中止机制解耦。新增 `_run_request_abortable_generate` 和 `_run_server_abort_generate` 两个包装器函数。`GenerateState` 新增 `cancellable_tasks` 和 `active_server_generations` 字段以分别跟踪请求级和服务级活动任务。`abort` 函数重写为: 先取消所有 `cancellable_tasks`, 然后仅在存在活动服务器生成时调用服务器级中止。
3. 重写流式生成函数(`slime/rollout/slang_streaming_rollout.py`): `generate_streaming` 函数被重写以使用 `SGLangStreamAccumulator` 处理 SSE 块。它初始化一个 `SGLangStreamAccumulator` 实例, 并在每个块上调用其 `add` 方法以获取更新。根据

update.replace\_call\_state 标志决定是否重置 sample 状态到基础值。函数末尾通过 stream.response\_text 方法一次性生成最终响应文本。在函数上设置 abort\_mode = "request" 属性以请求级中止模式。

4. 添加全面测试(tests/test\_streaming\_rollout.py): 新增包含 26 个参数化测试用例的测试套件, 覆盖累积与增量流、不同流间隔、top-p 与路由专家元数据、错误格式拒绝、请求取消与服务器中止的区分、以及混合完成 / 中止兄弟组的场景。
5. 更新配置与 CI: 在 slime/utils/arguments.py 中更新了 --custom-generate-function-path 的帮助文本以说明 abort\_mode 属性。将新的测试文件添加到 .github/workflows/pr-test.yml 和模板 .github/workflows/pr-test.yml.j2 中以纳入 CI。

关键文件:

- slime/rollout/streaming\_utils.py (模块 流式处理; 类别 source; 类型 dependency-wiring; 符号 \_has\_full\_response\_top\_p\_metadata, SGLangStreamUpdate, SGLangStreamAccumulator, add): 新增核心流处理模块, 包含 SGLangStreamAccumulator 类, 负责规范化、验证和累积 SGLang 的 SSE 流数据。
- slime/rollout/sglang\_rollout.py (模块 Rollout 调度; 类别 source; 类型 dependency-wiring; 符号 \_run\_request\_abortable\_generate, \_run\_server\_abort\_generate, abort, generate\_and\_rm): 核心修改, 引入了请求级和服务级中止的分离机制, 并重构了生成调用流程以支持 abort\_mode 属性。
- tests/test\_streaming\_rollout.py (模块 流式测试; 类别 test; 类型 test-coverage; 符号 \_chunk, \_b64\_int32, test\_merge\_stream\_chunks, test\_merge\_stream\_rejects\_inconsistent\_length): 提供极其全面的测试覆盖, 验证了流处理、中止机制和集成场景, 是本次变更质量的关键保障。

关键符号: slime.rollout.streaming\_utils.SGLangStreamAccumulator.add, slime.rollout.streaming\_utils.SGLangStreamAccumulator.response\_text, slime.rollout.sglang\_rollout.\_run\_request\_abortable\_generate, slime.rollout.sglang\_rollout.\_run\_server\_abort\_generate, slime.rollout.sglang\_rollout.abort, slime.rollout.sglang\_streaming\_rollout.generate\_streaming

## 关键源码片段

### slime/rollout/sglang\_rollout.py

核心修改, 引入了请求级和服务级中止的分离机制, 并重构了生成调用流程以支持 abort\_mode 属性。

```
# 文件: slime/rollout/sglang_rollout.py
# 该片段展示了新增的中止包装器和 `generate_and_rm` 中的关键分发逻辑。
```

```
async def _run_request_abortable_generate(
    state: GenerateState,
    sample: Sample,
    generate_call: Awaitable[Sample | list[Sample]],
```

```
) -> Sample | list[Sample]:  
    """包装器：支持请求级中止。
```

将当前任务加入 `cancellable\_tasks` 集合。如果任务被取消且仍在该集合中，重新抛出异常（让调用方处理）。否则，将样本标记为 `ABORTED` 并返回。这确保了只有 Slime 自己发起的取消才会产生 `ABORTED` 状态。

```
    """  
  
    task = asyncio.current_task()  
    assert task is not None  
    state.cancellable_tasks.add(task)  
    try:  
        return await generate_call  
    except asyncio.CancelledError:  
        if task in state.cancellable_tasks:  
            raise  
        sample.status = Sample.Status.ABORTED  
        return sample  
    finally:  
        state.cancellable_tasks.discard(task)
```

```
async def _run_server_abort_generate(  
    state: GenerateState,  
    generate_call: Awaitable[Sample | list[Sample]],  
) -> Sample | list[Sample]:  
    """包装器：传统服务器级中止。
```

仅跟踪活动服务器生成的数量，用于 `abort` 函数决定是否调用服务器中止。

```
    """  
  
    state.active_server_generations += 1  
    try:  
        return await generate_call  
    finally:  
        state.active_server_generations -= 1
```

```
# 在 `generate_and_rm` 中，根据生成函数的 `abort_mode` 属性选择包装器：  
# ...  
if getattr(generate_func, "abort_mode", None) == "request":  
    sample = await _run_request_abortable_generate(state, sample, generate_call)  
else:  
    sample = await _run_server_abort_generate(state, generate_call)  
# ...
```

```
# `abort` 函数被重写，优先取消请求级任务：  
async def abort(args: Namespace, rollout_id: int) -> list[list[Sample]]:  
    # ...  
    state.aborted = True  
    # 取消所有可取消的任务
```

```
cancellable_tasks = list(state.cancellable_tasks)
state.cancellable_tasks.difference_update(cancellable_tasks)
for task in cancellable_tasks:
    task.cancel()
# 仅当存在活动服务器生成时，才调用服务器级中止
if state.active_server_generations:
    response = await get(f"http://{args.sglang_router_ip}:{args.sglang_router_port}/workers")
    urls = [worker["url"] for worker in response["workers"]]
    await abort_servers_until_idle(urls)
await asyncio.gather(*cancellable_tasks, return_exceptions=True)
# ...
```

## 评论区精华

本 PR 没有留下公开的讨论评论，审核者 [zhuzilin](#) 直接给予了 APPROVED 状态。这表明核心设计决策（如请求级中止模式通过函数属性 `abort_mode` 传递、流处理模块的职责划分）在提交前已达成共识。PR body 详细阐述了设计，可能讨论在合并前进行。

- 请求级中止机制的设计与实现 (design): 设计决策被接受并实现。PR body 和代码注释详细解释了设计原理。中止函数 `abort` 被重写以优先处理请求级取消。
- 流式格式规范化与验证 (correctness): 核心逻辑在 `streaming_utils.py` 中实现，通过明确的模式判断和长度验证来确保正确性。测试用例覆盖了各种格式组合和错误情况。
- 部分 Rollout 恢复与元数据快照 (design): 通过保存 `base_*` 状态并在 `replace_call_state` 为真时恢复来实现。测试验证了在不同流间隔下元数据的正确保留。

## 风险与影响

- 风险：
  1. 向后兼容性：新增功能不破坏现有 API，`abort_mode` 属性缺失时回退到服务器级中止，保持向后兼容。
  2. 流处理正确性：SGLangStreamAccumulator 中的长度验证逻辑是关键，如果 SGLang 输出与配置的模式不匹配会抛出明确错误，避免静默错误训练。
  3. 中止机制：请求级中止依赖于 Python 任务取消 (CancelledError)，实现逻辑清晰（检查任务是否在 `cancellable_tasks` 中）。测试覆盖了取消场景。
  4. 依赖：未引入新依赖。对现有模块 (`sglang_rollout.py`) 的修改是添加性的，主要修改了函数结构而非核心算法。
  5. 性能：PR body 提供了增量累加器的基准测试数据，表明性能开销可接受。流式处理本身可能对长上下文有益。
- 影响：
  1. 用户体验：启用 `--custom-generate-function-path` `slime.rollout.sglang_streaming_rollout.generate_streaming` 后，用户可以使用 SGLang 的流式输出，特别是增量模式 (`--sglang-incremental-streaming-output`) 可改善长上下文生成体验。

2. 系统功能：显著增强了外部 rollout 路径的能力，使 Slime 在处理生成中断（如权重更新、部分 rollout 回收）时更加健壮，能够保留部分生成结果。中止机制的精细化减少了不必要的服务器重启。
3. 开发团队：为自定义生成函数作者提供了新的中止模式选项 (`abort_mode = 'request'`)，扩展了插件契约。新增的测试套件为流式处理逻辑提供了可靠的回归保障。
4. 运维：增量流可能减少网络传输数据量（仅传输增量），但需要 SGLang 服务器端配合开启增量模式。 - 风险标记：核心路径变更，新增外部依赖集成点，复杂状态管理

## 关联脉络

- PR #2323 [cleanup] Refactor rollout.py: 本 PR 是对 rollout 子系统功能的扩展，与 PR#2323 对 `slime/ray/rollout.py` 的清理和拆分工作属于同一条功能演进线，共同完善了 rollout 的架构。
- PR #2238 fix: stop the fully-async rollout dropping completed groups: 本 PR 解决了外部 rollout 路径的中断保留问题，而 PR#2238 解决了全异步 rollout 路径丢弃已完成组的问题。两者都旨在提高 rollout 过程在各种中断场景下的鲁棒性。
- PR #2340 fix(agent): abort timed-out SGLang requests via router workers: 本 PR 实现了通用的请求级中止机制。PR#2340 修复了 agent 路径中通过路由器中止 SGLang 请求的失效问题，可以看作是本 PR 所引入的基础设施在特定路径（agent）上的应用或修复。