PR #2238 完整报告

THUDM/slime

fix: stop the fully-async rollout dropping completed groups

合并时间：2026-08-12 14:05

原文链接：http://prhub.com.cn/THUDM/slime/pull/2238

执行摘要

- 一句话：修复全异步 rollout 丢弃已完成组并解除事件循环阻塞
- 推荐动作：值得精读。该 PR 是典型的“两个缺陷相互耦合、需联动修复”的案例，其背压转移设计（将有界队列改为无界、在生产者侧用 qsize 门控）是一个可复用的异步队列模式。建议关注 get_completed_groups 的 limit 语义与 _loop 门控条件，理解为何“无界队列 + 显式门控”优于“有界队列 + 阻塞 put”在事件循环线程中的表现。

功能与动机

PR body 明确指出两个缺陷：

1) _generate_rollout_async 每次轮询排空整个输出队列，却只返回 [:rollout_batch_size] 的切片，超出的 group 被丢弃——这些 group 已完成生成和奖励打分，且 prompt 已从 data_buffer 消费，永远丢失，属于 GPU 算力浪费； 2) 任务完成回调运行在事件循环线程上，向 maxsize=1000 的有界队列做阻塞 put，当在途 group 数（示例配置 512×3=1536）超过上限时，回调会阻塞事件循环，冻结所有在途生成。两个缺陷相互耦合：单独修 bug 1 会让盈余积压成为常态，使 bug 2 从偶发卡顿变成必然冻结。

实现拆解

1. 修改队列拉取契约：slime/rollout/fully_async_rollout.py 中 get_completed_groups 新增 limit: int | None = None 参数，按需弹出最多 limit 个 group（None 时全部弹出）。_generate_rollout_async 的 poll 循环改为每次只取 target - len(collected) 个，盈余继续留在队列中等待下一次 rollout，恢复文档承诺的 "queue stays warm" 行为，同时删除原来 [:target] 的丢弃切片。
2. 移除事件循环阻塞源：AsyncRolloutWorker.__init__ 中将 queue.Queue(maxsize=1000) 改为无界 queue.Queue()，避免 done-callback 在事件循环线程上阻塞。真正的背压转移到 _loop() 的 top-up 门控：仅当 output_queue.qsize() < max_concurrent 时才继续从 data_buffer 拉取新 prompt，内存上界约为 2 × concurrency 个 group，与原先在途池数量级一致。
3. 可配置轮询间隔：_loop 中两处 sleep(1) 改为 sleep(self.poll_interval)（默认 1.0，行为不变），使 CPU 测试可以调小间隔快速驱动循环。
4. 新增 CPU 单测与 CI 注册：新增 tests/test_fully_async_rollout.py，通过 stub sglang_router/transformers 在无 GPU 环境验证四条核心行为：生成返回恰好 target 个

FIFO group 且盈余留队、get_completed_groups 的 limit 语义、done-callback 超过旧 1000 容量上限不阻塞、以及背压门控让队列维持 ≤2×concurrency。在 .github/workflows/pr-test.yml 和 .github/workflows/pr-test.yml.j2 的 cpu-unittest 作业中注册新测试。

关键文件：

- slime/rollout/fully_async_rollout.py（模块 异步队列；类别 source；类型 core-logic；符号 AsyncRolloutWorker.init, AsyncRolloutWorker.get_completed_groups, AsyncRolloutWorker._loop, _generate_rollout_async）：核心修复文件，涉及队列契约、背压机制和轮询间隔三处关键改动。get_completed_groups 新增 limit 参数，output_queue 改为无界，_loop 增加 qsize 门控。
- tests/test_fully_async_rollout.py（模块 异步队列；类别 test；类型 test-coverage；符号 _FakeGenerateState, _FakeDataBuffer, _make_group, _make_worker）：新增 CPU 单测，覆盖队列契约、limit 语义、done-callback 不阻塞和背压门控四条核心行为，是验证本次修复的关键配套。
- .github/workflows/pr-test.yml（模块 CI 配置；类别 infra；类型 infrastructure）：在 cpu-unittest 作业中注册新测试文件，确保 CI 覆盖。
- .github/workflows/pr-test.yml.j2（模块 CI 配置；类别 infra；类型 infrastructure）：pr-test.yml 的模板源文件，同步新增测试条目。

关键符号：AsyncRolloutWorker.init, AsyncRolloutWorker.get_completed_groups, AsyncRolloutWorker._loop, _generate_rollout_async

关键源码片段

slime/rollout/fully_async_rollout.py

核心修复文件，涉及队列契约、背压机制和轮询间隔三处关键改动。get_completed_groups 新增 limit 参数，output_queue 改为无界，_loop 增加 qsize 门控。

```
# slime/rollout/fully_async_rollout.py
class AsyncRolloutWorker:
    def __init__(self, args, data_buffer, concurrency: int = 10):
        self.args = args
        self.data_buffer = data_buffer
        self.concurrency = concurrency
        self.running = True
        # 输出队列改为无界：put() 运行在事件循环线程（task done-callback），
        # 有界队列一旦写满会阻塞循环本身，冻结所有在途生成。
        # 真正的背压转移到 _loop() 的 top-up 门控。
        self.output_queue: queue.Queue[tuple[int, list[Sample]]] = queue.Queue()
        self.poll_interval = 1.0
        self.worker_thread: threading.Thread | None = None
        self.state = GenerateState(args)

    def get_completed_groups(self, limit: int | None = None) -> list[tuple[int, list[Sample]]]:
        """最多弹出 ``limit`` 个已完成 group（``None`` 时全部弹出）。
```

```
    只需固定数量 group 的调用方必须传 ``limit``——超出部分一旦弹出就会
    被丢弃，而这些 group 已完成生成与奖励打分，且其 prompt 已从
    ``data_buffer`` 消费掉，无法重新入队。
    """
    completed: list[tuple[int, list[Sample]]] = []
    while limit is None or len(completed) < limit:
        try:
            completed.append(self.output_queue.get_nowait())
        except queue.Empty:
            break
    return completed

async def _loop(self) -> None:
    active_tasks: set[asyncio.Task] = set()
    max_concurrent = self.concurrency
    gid_counter = 0

    while self.running:
        try:
            # 收割已完成的 task
            if active_tasks:
                done = {t for t in active_tasks if t.done()}
                for t in done:
                    try:
                        t.result() # 结果已在回调中处理
                    except Exception as e: # noqa: BLE001
                        logger.warning("fully-async task crashed: %r", e)
                active_tasks -= done

            # 补给。qsize 门控承担背压：一旦已有一整批（concurrency）已完成
            # group 排队等待消费，就暂停拉取新 prompt，直到训练侧消费掉一些。
            while (
                len(active_tasks) < max_concurrent
                and self.output_queue.qsize() < max_concurrent
                and self.running
            ):
                groups = self.data_buffer.get_samples(1)
                if not groups:
                    break
                for group in groups:
                    gid = gid_counter
                    gid_counter += 1
                    task = asyncio.create_task(
                        generate_and_rm_group(
                            self.args,
                            group,
                            sampling_params=self.state.sampling_params.copy(),
                            evaluation=False,
                        )
```

```
                )
                task.add_done_callback(self._make_done_cb(gid))
                active_tasks.add(task)

            await asyncio.sleep(self.poll_interval)
        except Exception as e: # noqa: BLE001
            logger.exception("fully-async loop iteration error: %s", e)
            await asyncio.sleep(self.poll_interval)
```

tests/test_fully_async_rollout.py

新增 CPU 单测，覆盖队列契约、limit 语义、done-callback 不阻塞和背压门控四条核心行为，是验证本次修复的关键配套。

```
# tests/test_fully_async_rollout.py
def _make_worker(monkeypatch, data_buffer=None, concurrency=4) -> fa.AsyncRolloutWorker:
    monkeypatch.setattr(fa, "GenerateState", _FakeGenerateState)
    args = SimpleNamespace(rollout_global_dataset=True, rollout_batch_size=4)
    return fa.AsyncRolloutWorker(args, data_buffer or _FakeDataBuffer([]), concurrency=
    concurrency)


@pytest.mark.unit
def test_rollout_takes_target_groups_and_leaves_surplus_queued(monkeypatch):
    # 队列预热 10 个已完成 group，目标 batch 为 4。
    worker = _make_worker(monkeypatch)
    for gid in range(10):
        worker.output_queue.put((gid, _make_group(gid)))
    monkeypatch.setattr(fa, "_get_global_worker", lambda args, data_buffer: worker)

    args = SimpleNamespace(rollout_global_dataset=True, rollout_batch_size=4)
    out = asyncio.run(fa._generate_rollout_async(args, rollout_id=0, data_buffer=None))

    assert len(out) == 4
    # FIFO：最老的 4 个先出队。
    assert [group[0].index for group in out] == [0, 1, 2, 3]
    # 其余 6 个仍留在队列中等待下一次 rollout，而不是被丢弃。
    assert worker.queue_size() == 6
    assert [gid for gid, _ in worker.get_completed_groups()] == [4, 5, 6, 7, 8, 9]
```

评论区精华

该 PR 无逐行 review 评论，但 Issue 评论中有外部反馈：shinytang6 表示团队遇到类似问题并请求 zhuzilin 处理；zhuzilin 回复感谢 bugfix 并道歉延迟 review。这说明该缺陷已影响真实用户，修复得到维护者认可。

- fully-async 问题被外部确认 (other): 外部用户确认遇到相同问题，维护者致谢并合并。

风险与影响

- 风险：
  1. 无界队列内存风险：虽然 _loop 的 qsize 门控限制了队列长度约为 concurrency，但若训练侧长时间不调用 generate_rollout 消费队列，且 worker 持续完成生成，队列最多仍会涨到约 concurrency 的量级，内存有界，但依赖门控逻辑正确性。若未来有人修改 _loop 条件，可能破坏此约束。
  2. warm-queue 语义变化：原先每次 generate_rollout 排空队列的隐式行为被替换为“取走目标数量、留下盈余”，任何依赖旧行为的调用方（例如期望每次拿到的 group 是“当前所有已完成”的代码）需要显式传 limit=None 才能保持旧行为。当前库内调用已全部适配，但外部使用者需注意。
  3. 测试覆盖局限：CPU 测试 stub 了 sglang_router 和 transformers，未覆盖真实 sglang 后端下的并发行为；在途任务异常回调路径（_make_done_cb 的异常分支）也没有专门测试。
  4. poll_interval 配置化：默认值保持 1.0，无行为回归；但若用户调小间隔，可能增加 CPU 空转，影响甚微。 - 影响：该修复直接影响使用 fully-async rollout 模式的用户（尤其是 examples/fully_async/run-qwen2.5-0.5B-fully_async.sh 这类高并发配置），消除了训练数据被静默丢弃和事件循环冻结导致训练停摆的问题，提升训练吞吐与稳定性。对团队而言，该 PR 明确了队列契约并附带可回归的 CPU 测试，降低了后续维护风险。改动集中在 slime/rollout/fully_async_rollout.py 一个源文件，影响面可控。 - 风险标记：核心异步路径变更，无界队列内存风险，warm-queue 语义变化，依赖调用方传 limit

关联脉络

- PR #2170 Fix placement group crash for external engines under debug_rollout_only: 同为 rollout 基础设施稳定性修复，涉及 slime/ray 与参数校验，与本 PR 共同保障外部引擎和异步 rollout 路径的可靠性。
- PR #2242 fix: honor every eval.defaults key and restore per-dataset stop / min_new_tokens: 同属 rollout 模块相关修复，修改了 slime/rollout/sglang_rollout.py 与 eval 配置，与本 PR 同属 rollout 稳定化工作线。