PR #2544 完整报告

radixark/miles

Do not kill the run when one sample's collect_samples loses its connection

合并时间：2026-08-15 03:25

原文链接：http://prhub.com.cn/radixark/miles/pull/2544

执行摘要

- 一句话：采集断连不再拖垮整个训练运行
- 推荐动作：值得精读。这是一个小而精准的生产事故修复：22 行源码与测试联动，但覆盖了大规模训练中最常见的一类稳定性问题。重点学习两点：一是异常边界的刻意收窄 —— 只吸收瞬态传输错误、保留服务器级错误的响亮失败；二是复用已有 ABORTED 机制而非新增降级路径，使修复面最小化。对维护 rollout 基础设施的工程师尤其有参考价值。

功能与动机

PR 描述明确记录：在 main 上运行 *examples/experimental/openenv/glm52_tbench2* 于 16 个 GB300 节点时，64 GPU 的 run 完成了 rollout 0 和训练步，却因单个样本的 HTTP 连接失败在 7 秒后死亡。根因是 *generate*() 只捕获 asyncio.TimeoutError，其他传输错误全部逃逸。这不是罕见路径：在 *--pause-generation-mode in_place* 下，引擎会在请求仍在飞行中时暂停，每次权重更新都会自然造成与sessionserver的连接中断，属于常规事件而非偶发故障。修复的对称性依据是同一函数里的 *agent* 循环早已用 *except* Exception 包裹，且 *collect* 失败本就有正确出口 —— 将样本标记 ABORTED 并交给 *check_no_aborted* 丢弃其组。

实现拆解

1. 在 miles/rollout/*generate_hub/agentic_tool_call.py* 中，*generate*() 是 *agentic* 路径的入口，被 *_generate_group* 与 fully-async worker 循环调用。原实现只捕获 asyncio.TimeoutError，其余异常（含 httpx.ReadError）会一路逃逸到 train_async 并终止运行。本次将 *except* 子句改为 (TimeoutError, httpx.TransportError)，并同步移除不再使用的 import asyncio、引入 import httpx。
2. 将 *collect_timed_out* 状态变量重命名为 *collect_failed*，统一表达“采集失败”而不只是超时。失败分支复用已有的 ABORTED 机制：深拷贝输入样本并置为 Sample.Status.ABORTED，由下游 *check_no_aborted* 丢弃该 *group*；输入样本自身保持 PENDING，不会污染原状态。
3. 刻意保留 RuntimeError 的传播路径。非 2xx 响应（例如 session server 返回 500）由 post_bytes_no_retry 以 RuntimeError 抛出，作者认为这是明确故障而非瞬态条件，应当大声失败，不能被静默吸收。
4. 同步更新 miles/rollout/*generate_utils/openai_endpoint_utils.py* 中 *collect_samples* 的注释，从“asyncio.TimeoutError 传播”改为“timeout 与 transport *errors* 传播”，保持调用契约的文档一致性。

5. 测试配套：tests/fast/rollout/generate_hub/test_multi_turn.py 的 TestAgentCollectionFailure 改为 pytest.mark.parametrize，覆盖 asyncio.TimeoutError 与 httpx.ReadError 两种瞬态错误，断言均产出 ABORTED 样本且输入保持 PENDING，并继续断言 RuntimeError 仍然逃逸；测试注册在 stage-b-cpu CPU CI 套件。

关键文件：

- miles/rollout/generate_hub/agentic_tool_call.py（模块 生成入口；类别 source；类型 core-logic；符号 generate）：核心修复文件：generate() 在此扩大 collect_samples 的异常捕获范围，将传输错误转为 ABORTED 样本，避免整轮训练被单个样本杀死。
- tests/fast/rollout/generate_hub/test_multi_turn.py（模块 多轮测试；类别 test；类型 test-coverage；符号 TestAgentCollectionFailure, test_collect_transient_failure_aborts_sample_but_other_errors_propagate）：测试配套：将 TestAgentCollectionFailure 参数化覆盖 timeout 与 transport 两类瞬态错误，并继续锁定 RuntimeError 必须传播的契约。
- miles/rollout/generate_utils/openai_endpoint_utils.py（模块 会话客户端；类别 source；类型 comment-sync；符号 collect_samples）：同步 collect_samples 的注释契约，明确 timeout 与 transport errors 都会向上传播给 generate 处理，保持文档与行为一致。

关键符号：generate, collect_samples, test_collect_transient_failure_aborts_sample_but_other_errors_propagate

关键源码片段

miles/rollout/generate_hub/agentic_tool_call.py

核心修复文件：generate() 在此扩大 collect_samples 的异常捕获范围，将传输错误转为 ABORTED 样本，避免整轮训练被单个样本杀死。

```
# miles/rollout/generate_hub/agentic_tool_call.py
# generate() 是 agentic 路径的入口，被 _generate_group -> fully-async worker loop 调用；
# 任何未捕获异常都会一路逃逸到 train_async，进而终止整个 run。

async def generate(input: GenerateFnInput) -> GenerateFnOutput:
    ...
    log_prefix = f"[session={tracer.session_id}]"
    agent_metadata = None
    collect_failed = False
    t_start = time.monotonic()

    try:
        # agent 循环自身失败只影响这一个 episode，宽泛 except 已经兜住
        agent_metadata = await custom_agent_function(
            base_url=tracer.base_url,
            prompt=input.sample.prompt,
            request_kwargs=build_chat_request_kwargs(input.sampling_params),
            metadata=metadata,
        )
```

```
    except Exception as e:
        logger.warning(f"{log_prefix} Agent function failed: {e}", exc_info=True)

    finally:
        # 即使 agent 失败也仍要收集样本
        collect_kwargs = {"max_seq_len": max_seq_len}
        if use_v2:
            collect_kwargs["agent_metadata"] = agent_metadata
        try:
            result = await tracer.collect_samples(input.sample, **collect_kwargs)
        # 会话服务器断连或超时只应牺牲这一个样本：标记 ABORTED 后，
        # 下游 check_no_aborted 会丢弃该组，其余样本保持继续；
        # 非 2xx 响应仍以 RuntimeError 抛出 —— 500 是明确故障，不应被吞掉。
        except (TimeoutError, httpx.TransportError) as e:
            collect_failed = True
            logger.warning(f"{log_prefix} Failed collecting samples: {e!r}", exc_info=True)
        else:
            logger.debug(f"{log_prefix} collect_samples done ...")

    if collect_failed:
        # 复制原样本并置为 ABORTED，避免污染输入样本（其状态保留 PENDING）
        sample = deepcopy(input.sample)
        sample.status = Sample.Status.ABORTED
        return GenerateFnOutput(samples=[sample] if use_v2 else sample)
```

tests/fast/rollout/generate_hub/test_multi_turn.py

测试配套：将 TestAgentCollectionFailure 参数化覆盖 timeout 与 transport 两类瞬态错误，并继续锁定 RuntimeError 必须传播的契约。

```
# tests/fast/rollout/generate_hub/test_multi_turn.py
# 参数化瞬态错误：超时 与 httpx 传输错误都应产生 ABORTED 样本，
# 而 RuntimeError（非 2xx 响应）仍必须逃逸，防止把服务端故障伪装成偶发断连。

class TestAgentCollectionFailure:
    @pytest.fixture(params=_AGENTIC_VARIANTS)
    def variant(self, request):
        return request.param

    @pytest.mark.parametrize(
        "collect_error",
        [
            pytest.param(asyncio.TimeoutError(), id="timeout"),
            # in-place 权重更新会在请求在途时暂停引擎，断连是常规事件
            pytest.param(httpx.ReadError("connection closed"), id="transport"),
        ],
    )
    def test_collect_transient_failure_aborts_sample_but_other_errors_propagate(
        self, variant, generation_env, monkeypatch, caplog, collect_error
    ):
```

```
async def fail_collect(_tracer, _input_sample, *, max_seq_len, agent_metadata=None):
    raise collect_error

monkeypatch.setattr(
    "miles.rollout.generate_utils.openai_endpoint_utils.OpenAIEndpointTracer.collect_
    samples",
    fail_collect,
)

input_sample = make_sample(prompt=TwoTurnStub.PROMPT)
with caplog.at_level(logging.WARNING):
    result = _run_generate(variant, generation_env, input_sample)

[sample] = listify(result.sample)
assert sample.status == Sample.Status.ABORTED # 瞬态错误只牺牲当前样本
assert input_sample.status == Sample.Status.PENDING # 原输入不被污染
assert "Failed collecting samples" in caplog.text

collect_error = RuntimeError("assembly failed")
with pytest.raises(RuntimeError, match="assembly failed"):
    _run_generate(variant, generation_env, make_sample(prompt=TwoTurnStub.PROMPT))
```

评论区精华

该 PR 未产生开放式 review 评论，两位 reviewer yushengsu-thu 与 guapisolo 均直接 approve，guapisolo 留下“good fix!”。核心设计权衡在 PR 描述中由作者完整论证：为什么只吸收 TimeoutError 和 httpx.TransportError，而不是宽泛地捕获所有异常 —— 非 2xx 响应抛出的 RuntimeError 表示 session server 正在返回 500，属于明确故障，应当继续中止并暴露问题。这一边界设定得到维护者认可。

- 哪些 collect_samples 异常应被吸收 (design): 维持作者的设计：仅瞬态传输错误被吸收并转为 ABORTED 样本，RuntimeError 仍会向上传播以暴露服务端故障。

风险与影响

- 风险：
  1. 异常边界收窄带来的误吸收风险：如果未来某类非瞬态错误被 httpx 包装为 TransportError（例如目标端口持续不可达），样本会被静默 ABORTED 并丢弃，可能掩盖配置或拓扑层面的真实故障；目前靠日志 warning 和 ABORTED 计数兜底。
  2. 依赖 ABORTED 丢弃机制：修复的容错效果取决于 check_no_aborted 能可靠丢弃失败样本所在 group；若该机制被误配或绕过，失败样本可能以 ABORTED 状态流入训练数据，影响数据质量。
  3. 仍存在其他逃逸窗口：except 并未覆盖 ValueError、KeyError 等编程错误类异常，一旦 collect_samples 内部出现这类错误，仍会按原路径终止整个 run。这属于刻意保留的“响亮失败”，但也意味着该修复只针对连接类故障。

4. 生产验证环境单一：验证来自单次 GLM52 tbench2 运行，未覆盖 session server 在压力下反复断连或清理阶段 post 请求本身失败的场景。 - 影响：影响范围集中在使用 session server 的 agentic 训练路径，尤其是 fully-async 模式和 --pause-generation-mode in_place 的权重更新窗口。此前每次权重更新引发的连接中断都可能随机杀死整个 run；修复后这些瞬态断连只会牺牲单个样本及其 group，训练可继续推进。对 v1 与 v2 agentic 路径同时生效，因为 generate() 是两条路径的公共入口。对非 agentic 或未启用 session server 的 rollout 无影响。团队收益是减少大规模训练运行的中断率和人工介入成本。 - 风险标记：核心路径异常处理调整，异常范围刻意收窄，依赖 ABORTED 丢弃机制，生产环境已验证

关联脉络

- PR #2536 Carry the rollout id on both agentic paths: 同样修改 miles/rollout/generate_hub/agentic_tool_call.py，修复 agentic v1/v2 路径的 rollout_id 一致性，属于同一条 agentic 生成链路的稳定性改进。
- PR #2522 Make the class-based rollout the default and convert legacy path to env var gated: 重构了 rollout 与 fully-async worker 循环路径，本文修复的异常正是从 _generate_group 经 fully-async 循环逃逸到 train_async，与其实践高度相关。
- PR #2347 [AMD] Enable amd pr ci: 涉及 agentic_tool_call.py 的改动并启用 ROCm PR CI，说明 agentic 生成路径是跨硬件平台的重点维护区域，本文续写了该路径的稳定性修复。