PR #2588 完整报告

radixark/miles

glm52_tbench2: collapse levers and launch fixes from the GB300 16-node smoke runs

合并时间：2026-08-18 08:36

原文链接：http://prhub.com.cn/radixark/miles/pull/2588

执行摘要

- 一句话：修复 GB300 16 节点 tbench2 训练崩溃，折叠三个关键配置杠杆
- 推荐动作：值得精读。这是「用一次 16 节点 smoke run 的逐 token 取证驱动配置和运行期行为修复」的典型范例：PR body 中关于 --tis-clip-low 反向后果、SDK 非幂等重试导致 zombie decode 的量化分析（~1000 引擎侧请求 vs 128 活跃样本）尤其值得关注。建议重点阅读三个设计决策：①TIS clip 下限回归默认值的理由（不要为了 " 梯度稳定 " 而违背 importance sampling 语义）；②max_retries=0 与 timeout=3600 的配对逻辑（生成请求不幂等，重试代价高于超时等待）；③finish_reason == "length" 终结 episode 的边界语义。同时建议跟进 #2591 与 --max-weight-staleness 的后续修复，它们与本 PR 构成完整的崩溃治理链路。

功能与动机

两个 16 节点 smoke run 在未修改的 main 上以完全相同的方式崩溃：train_rollout_kl 在 0.02 附近走平，随后 30 步内指数级攀升 57 倍，集中在近似确定性的 token 上，接着响应长度爆炸、截断率 ~0.9、reward 跌至 0.03。PR body 明确指出「--tis-clip-low 0.5 clamps 一个坍缩的 importance ratio（exp(-30)）到 0.5——恰好制造出 importance sampling 认为不应存在梯度的位置的梯度」，同时「SDK 默认 600s 超时 + 自动重试在引擎继续解码原请求时重复提交，僵尸请求堆积——实测引擎侧约 1000 个运行中请求对应 128 个活跃样本（约 85-90% 解码能力被浪费），每小时约 500 个重复请求」。目标是把这一诊断结论沉淀为可复现的 recipe 配置与运行期防御。

实现拆解

变更以三类手段落地：收敛训练配置、修正 agent 运行期行为、补强启动与配额错误处理。

1. 配置杠杆折叠（run_glm5_2_744b_a40b_daytona.py） - --tis-clip-low 0.5 → 0.0：恢复 miles 默认的标准截断 IS（w = min(ratio, 2)），避免在坍缩 ratio 上强行 clamp 制造伪梯度。 - rollout_max_response_len 16384 → 8192：正常完成样本从未触及上限，只有退化循环会；减半让被 cap 的循环 token 量不再翻倍。 - 新增 max_seq_len: int = 65536 并替换硬编码的 --max-seq-len 131072；eval_interval 从 int | None 改为 int 且 0 表示关闭 eval，绕过 typer 无法表达 "off" 的限制，__post_init__ 中同步把判断从 is not None 改为 truthy，避免 eval_interval=0 时误填 eval 数据路径。 - 新增 openenv_daytona_create_max_retries（默认 8）并透传为 OPENENV_DAYTONA_CREATE_MAX_RETRIES 环境变量。

2. Agent 运行期行为（openenv_agent_function.py、eval_tbench2_via_api.py） - policy client 以 timeout=3600.0, max_retries=0 构造：3600 匹配 episode 墙钟上限，max_retries=0 因为重发一次生成并非幂等操作，防止重复请求累积成僵尸解码。 - finish_reason == "length" 的轮次立即终结 episode，不再把截断后的命令前缀喂给沙箱执行。
3. 启动与配额错误防御（launch_16node_slurm.sh、openenv_daytona_agent_function.py） - FABRIC_PREFIX 改为必填，并以 : "${head_ip:?...}" 在 Ray 启动前显式失败，避免此前静默选中不可路由地址后悬挂 15 分钟无报错。 - _is_throttle_error 在文本匹配中追加 " limit exceeded"，把 Daytona 组织配额耗尽（CPU/ 内存 / 磁盘三种措辞，实际以 DaytonaValidationError 到达）与 429 限流同等对待，纳入抖动指数退避重试。
4. 测试配套 - test_openenv_agent_function.py 新增 _TruncatingPolicy 与 test_length_capped_turn_ends_the_episode，断言截断轮次后 turns=1、tool_calls=0 且不执行任何命令。 - test_openenv_daytona_agent_function.py 新增 test_is_throttle_error_quota_exhaustion，覆盖 CPU/ 内存配额耗尽文本并排除无关配置错误。 - 一个独立 commit 修正了 _FakePolicy 假完成对象缺少 finish_reason 导致 4 个既有测试崩溃的问题（真实 chat completion 总携带该字段，因此改为在 fake 中补齐而非让生产代码防御）。
5. 文档同步（README.md） - 更新 Episode 场景的 token 预算描述（8k tokens/turn、65k session）、FABRIC_PREFIX 必填说明，以及 --eval-interval 0 可关闭 eval 的说明。

关键文件：

- examples/experimental/openenv/glm52_tbench2/run_glm5_2_744b_a40b_daytona.py（模块 训练脚本；类别 source；类型 core-logic；符号 ScriptArgs, _execute_train, post_init）：崩溃修复的主战场：折叠 tis-clip-low、rollout_max_response_len、max_seq_len、eval-interval 四个核心配置，并把 Daytona 重试次数透传为环境变量；是 smoke run 可复现性的唯一入口。
- tests/fast/examples/experimental/openenv/test_openenv_agent_function.py（模块 Agent 循环；类别 test；类型 test-coverage；符号 _TruncatingPolicy, _create, test_length_capped_turn_ends_the_episode, spying_with_env）：新增 length 截断终结 episode 的核心行为测试，_TruncatingPolicy 通过改写 finish_reason 精确模拟截断场景；Zhichenzzz 还在此文件修复了 fake completion 缺字段导致的测试崩溃。
- examples/experimental/openenv/openenv_daytona_agent_function.py（模块 Daytona 后端；类别 source；类型 core-logic；符号 _is_throttle_error）：将 Daytona 组织配额耗尽错误并入 throttle 判定，使共享后端的抖动退避重试生效，避免 episode 在首次模型调用前被整体丢弃；是崩溃治理中 " 减少失败样本循环 " 的关键一环。
- examples/experimental/openenv/eval_tbench2_via_api.py（模块 评测入口；类别 source；类型 entrypoint；符号 main）：一行改动但意义重大：policy client 从 SDK 默认的 600s 超时 / 重试 2 次改为 timeout=3600/max_retries=0，直接消除每小时约 500 个重复请求的僵尸解码来源。
- tests/fast/examples/experimental/openenv/test_openenv_daytona_agent_function.py（模块 Daytona 后端；类别 test；类型 test-coverage；符号 test_is_throttle_error_quota_exhaustion）：新增 quota 耗尽错误分类测试，覆盖 CPU/

内存配额文本并排除配置类错误，锁定 _is_throttle_error 的扩边行为。

- examples/experimental/openenv/glm52_tbench2/launch_16node_slurm.sh（模块 启动脚本；类别 other；类型 core-logic）：FABRIC_PREFIX 从可选默认 10. 改为必填，并新增 head_ip 空值断言，把 15 分钟静默悬挂变成启动即失败。
- examples/experimental/openenv/glm52_tbench2/README.md（模块 示例文档；类别 docs；类型 documentation）：同步 Episode 场景的 token 预算、FABRIC_PREFIX 必填和 eval 开关语义，保证 16 节点运行可从 main 直接复现。

关键符号：run_episode, _is_throttle_error, _execute_train, ScriptArgs.post_init, test_length_capped_turn_ends_the_episode, test_is_throttle_error_quota_exhaustion, main

关键源码片段

examples/experimental/openenv/glm52_tbench2/run_glm5_2_744b_a40b_daytona.py

崩溃修复的主战场：折叠 tis-clip-low、rollout_max_response_len、max_seq_len、eval-interval 四个核心配置，并把 Daytona 重试次数透传为环境变量；是 smoke run 可复现性的唯一入口。

```
@dataclass
class ScriptArgs(U.ExecuteTrainConfig):
    # ... 省略无关字段 ...

    rollout_max_response_len: int = 8192
    # 16384 -> 8192：正常完成的样本从未触及该上限，只有退化循环会撞上；
    # 16384 只会让被 cap 的循环 token 量翻倍，毫无收益。
    max_seq_len: int = 65536
    # 整段会话预算；CP 会拆分它，因此它也决定每 rank 的激活量。
    # TODO(#2591)：agent loop 看不到这个截断，仍会生成超过它。

    # ... 省略其余字段 ...

    def __post_init__(self):
        # eval_interval 从 int | None 改为 int：0 表示关闭 eval，
        # typer / argparse 无法用 None 表达 "off"。
        if self.eval_interval and not self.eval_prompt_data:
            self.eval_prompt_data = f"{self.data_dir}/tbench2_eval.jsonl"


def _execute_train(args: ScriptArgs):
    # ... 省略 ...

    grpo_args = (
        "--advantage-estimator grpo "
        "--use-tis "
        "--tis-clip-low 0.0 "
```

```
    # 0.5 -> 0.0：0.5 会把坍缩的 importance ratio（exp(-30)）强行 clamp
    # 到 0.5，恰恰在 importance sampling 认为不该有梯度的地方制造梯度；
    # 恢复 miles 默认值即标准截断 IS（w = min(ratio, 2)）。
    "--tis-clip 2.0 "
)

# ... 省略 ...

rollout_args = (
    "--fully-async "
    f"--rollout-max-response-len {args.rollout_max_response_len} "
    f"--max-seq-len {args.max_seq_len} "
    # 131072 硬编码 -> 可配置旗标，并同步更新 CP 拆分注释。
    "--rollout-temperature 0.8 "
)
```

tests/fast/examples/experimental/openenv/test_openenv_agent_function.py

新增 length 截断终结 episode 的核心行为测试，_TruncatingPolicy 通过改写 finish_reason 精确模拟截断场景；Zhichenzzz 还在此文件修复了 fake completion 缺字段导致的测试崩溃。

```
class _TruncatingPolicy(_FakePolicy):
    """Emits a command the model never finished writing."""

    async def _create(self, **kw):
        completion = await super()._create(**kw)
        # 真实 chat completion 总会携带 finish_reason，这里直接改写它：
        # 模拟模型生成本身被 per-turn token 上限截断。
        completion.choices[0].finish_reason = "length"
        return completion


def test_length_capped_turn_ends_the_episode(monkeypatch):
    """token 上限截断的轮次不得执行：命令是不完整的，执行它等于把一段
    任意前缀发进沙箱。"""
    monkeypatch.setattr(oaf, "load_tbench2", lambda: _CLASSES)

    async def spying_with_env(env_cls, env_url, body):
        return await body(env_cls())

    monkeypatch.setattr(oaf, "_with_env", spying_with_env)

    _, metrics = run_async(
        oaf.run_episode(_TruncatingPolicy(), "m", [{"role": "system", "content": "s"}], {}, {"task_id":
        "t1"})
    )

    # 截断轮次应立即终结 episode：不产生 tool call，也不执行任何命令
    assert metrics["turns"] == 1
    assert metrics["tool_calls"] == 0
```

```
    execs = [a for a in _FakeEnv.last_actions if a.action_type == "exec"]
    assert not [a for a in execs if "echo hi" in (a.command or "")], "ran a command the model
    never finished"
```

examples/experimental/openenv/openenv_daytona_agent_function.py

将 Daytona 组织配额耗尽错误并入 throttle 判定，使共享后端的抖动退避重试生效，避免 episode 在首次模型调用前被整体丢弃；是崩溃治理中 " 减少失败样本循环 " 的关键一环。

```
def _is_throttle_error(exc: BaseException) -> bool:
    """当且仅当 sandbox 创建失败只是因为 Daytona 暂时无法安置它时返回 True。

    SDK 会把 HTTP 429 归一化为 DaytonaRateLimitError；文本匹配是为了兼容
    旧版 SDK 以及只以文本形式暴露消息的服务器（如 "ThrottlerException: Too
    Many Requests"）。

    组织配额耗尽（"Total CPU limit exceeded. Maximum allowed: 500." 以及
    对应的内存/磁盘措辞）会以 DaytonaValidationError 形式到达，但在共享
    org 中同样属于瞬时错误：episode 在第一次模型调用前被中止会让
    check_no_aborted 丢弃整个 group，因此等待重试优于立即失败。
    """
    # 刻意做成函数内 import：这是该类的唯一使用点，且只在失败路径上运行
    # （此时 daytona 已在 sys.modules 中，因为 exc 刚由它抛出），
    # 旧版 SDK 则完全不存在这个类。
    try:
        from daytona.common.errors import DaytonaRateLimitError

        if isinstance(exc, DaytonaRateLimitError):
            return True
    except ImportError: # pragma: no cover - only without the daytona SDK
        pass
    # "limit exceeded" 兜住 CPU / 内存 / 磁盘三类配额耗尽消息
    return common.throttle_text(exc, "throttler", "limit exceeded")
```

评论区精华

1. nblintao 在 approval 中提示「You probably need to rebase this on PR 2587?」——PR #2587 恰好也在处理 openenv 截断轮次立即结束 episode 的问题，两 PR 高度同源；作者后续通过 merge origin/main 将 #2587 变更合入本分支。
2. Zhichenzzz 的修复 commit 点出：本 PR 在 run_episode 中读取 completion.choices[0].finish_reason == "length" 后，_FakePolicy 构造的 SimpleNamespace 只带 message 字段，导致 4 个测试（分布在两个文件）以 AttributeError 失败；结论是「fake 才是错的，真实 chat completion 总是携带 finish_reason，直接读取是正确的，让生产代码做防御只是在掩盖 fake 的缺陷」。
3. 代码中保留 TODO(#2591)：max_seq_len 作为整段会话预算，CP 拆分后决定每 rank 激活量，但 agent loop 看不到这个截断，仍会生成超过预算的内容——这是本 PR 主动标注、留待后续 issue 解决的已知缺口。

- 是否需要基于 PR 2587 变基 (question): 作者通过 merge origin/main 将 2587 的变更合入本分支，两 PR 的截断行为在此归一。
- fake completion 缺少 finish_reason 导致测试崩溃 (testing): 修复 fake 而非让生产代码防御：真实 chat completion 总是携带 finish_reason，直接读取是正确的。
- max_seq_len 与 agent loop 的预算缺口 (design): 以 TODO(#2591) 显式标注为已知缺口，留给后续 issue 处理。

风险与影响

- 风险：
  1. Agent 行为变更的隐性影响面：finish_reason == "length" 直接结束 episode 的逻辑位于共享的 openenv agent 函数中，不限于 GLM-5.2 recipe——所有走该 agent 路径的示例（包括 tbench2 及各 sandbox 后端）都会改变截断轮次的处置方式，可能让原本能靠多轮迭代救回的边界场景提前终止。
  2. 配置语义变化：eval_interval 从 int | None 改为 int（0 表示关闭），__post_init__ 判断从 is not None 改为 truthy；若外部调用方传入负数或依赖 None 语义，行为会与预期不符。rollout_max_response_len 降至 8192 依赖「正常样本从不触及上限」的观测，若后续任务分布变化出现长输出，会触发新的截断。
  3. 僵尸请求的残余通道：PR body 明确说明「残余的带内陈旧 group 缓慢 ratchet 正在通过 --max-weight-staleness 另行处理」，即超时 / 重试修复只消除了主要泄漏通道，陈旧 group 问题仍未闭环。
  4. 启动前置条件收紧：FABRIC_PREFIX 必填会让未配置该变量的用户启动即失败，这是有意为之，但属于破坏性变更，需要 README 引导到位。
  5. 重试窗口延长：配额耗尽错误纳入退避重试后，OPENENV_DAYTONA_CREATE_MAX_RETRIES=8（约 30s 封顶）可能延长一次失败创建的检测时间，但换来 group 不被整体丢弃。 - 影响：对用户：修复了 GLM-5.2 744B-A40B 在 GB300 16 节点规模下的训练崩溃，使长程 agentic tbench2 训练从「必崩」变为稳定运行 70+ 步（2.7 倍于未修复寿命），并释放约 85-90% 被僵尸请求浪费的引擎解码容量；同时所有 openenv 用户都会感知到截断轮次终结 episode 这一行为变化。对系统：训练侧的 train_rollout_kl 从指数发散收敛到 0.03，tis_clipfrac ~0.007，reward 健康爬升；工程层面沉淀了可复现的 16 节点 rcp 基线（launch_16node_slurm.sh 参数可复现 jobs 2403/2415）。对团队：PR body 的崩溃根因链（失败样本循环 → 负 advantage 污染共享 scaffolding token → TI clip 制造伪梯度 → 僵尸解码浪费容量）为后续 --max-weight-staleness 等治理提供了可引用的诊断范式。影响范围集中在 examples/experimental/openenv 下的实验性示例，但 agent 行为改动触及共享代码，属于中等偏大的影响面。 - 风险标记：核心 Agent 行为变更，配置语义变更，已知待跟进缺口 #2591，依赖姊妹 PR 同步，启动前置条件收紧

关联脉络

- PR #2587 openenv: stop the episode at a length-truncated turn: 姊妹 PR，同样处理 length 截断轮次立即结束 episode；nblintao 明确提示需 rebase 同步，本 PR 通过 merge main 归一了该行为。

- PR #2545 Reuse one Daytona client per process instead of one per create attempt: 同一 openenv 示例目录的 Daytona 基建修复，修复客户端 fd 泄漏；本 PR 在 Daytona 创建失败重试上继续增强。
- PR #2544 Do not kill the run when one sample's collect_samples loses its connection: rollout 稳定性主题的延续：单个样本故障不应拖垮整个训练，与本 PR 的失败样本循环治理方向一致。
- PR #2581 dashboard: resolve CP/PP-sharded train dumps offline and fix a partition-reader race: PR body 的崩溃诊断依赖 train dumps 的逐 token 取证，dashboard 侧的分区读取修复为这类分析提供工具链支撑。