

PR #7852 完整报告

verl-project/verl

[ci] fix: wait for and diagnose empty training log in qwen3_vl nightly check step

合并时间: 2026-09-16 15:02

原文链接: <http://prhub.com.cn/verl-project/verl/pull/7852>

执行摘要

- 一句话: 夜间 Ascend 检查步骤增加日志等待与诊断
- 推荐动作: 建议 CI 维护者精读这次改动, 它示范了一种成本极低的「先吸收延迟、再落证据」的 flaky 处理策略: 有界轮询加 `-s` 判定、诊断命令挂 `|| true`、路径收敛到变量, 都值得复用到其他 Ascend 夜间作业的同类检查步骤。重点看三处设计决策: (1) 60s 上限与 10s 间隔的取舍; (2) 诊断只打尾部 4096 字节; (3) 把日志路径抽成 LOG 变量。若追求更彻底, 可进一步讨论把检查输入改为 Actions stdout、或在训练脚本里给 tee 加 `stdbuf -oL / 追加后 sync`, 以及把诊断文件作为 artifact 上传。同时务必在团队内同步: 该作业要恢复全绿还需在 verl-project/verl-ascend-recipe 仓库重新生成 `baseline_qwen3_vl_8b_Instruct_fsdp2_npu.txt`, 这不是本仓库能解决的。

功能与动机

PR body 给出明确的失败签名: 9 天内复现 3 次, 最近一次为 `run 34506778944 / job 102970797276`。原文描述「training completes all 15/15 steps and every step:N line is present in the Actions-captured stdout, but the file tee'd by `tests/special_npu/run_qwen3_vl_8b_Instruct_fsdp2_npu.sh` is empty when the checking step reads it ~6 s later」, 于是 `check_npu.py` 抛 **ValueError: 日志中无有效step**, 作业转红, 而训练步骤本身退出码为 0。作者明确认定这是「runner-side file content loss/delay on the k8s custom-container runner, not a verl bug」, 因此选择先吸收延迟、再把下一次复现的证据 (missing / empty / truncated) 固定下来, 而不是盲目重跑。

实现拆解

1. 入口定位: 改动集中在 `.github/workflows/nightly_ascend.yml` 中 Running `nightlyCI_grpo_qwen3_vl_8b_Instruct_fsdp2_vllm_ascend checking script` 这一步 (该作业仅在 `schedule: '0 17 * * *'` 触发, PR 阶段不执行)。步骤开头 `cd /root/.cache/nightly_log/` 保持不变, 新增局部变量 `LOG=run_qwen3_vl_8b_Instruct_fsdp2_npu/run_qwen3_vl_8b_Instruct_fsdp2_npu.log`, 把原先散落硬编码的路径收敛到一处, 避免等待、诊断、校验三处路径漂移。
2. 有界等待吸收延迟: 新增 `for i in $(seq 1 6)` 轮询, 用 `[-s "$LOG"]` 判断日志「存在且非空」即 `break`, 否则 `echo` 提示并 `sleep 10`, 上限 60s。选择 60s 是对「失败路径不能无限阻塞」与「正常路径几乎不付代价」之间的折中: 日志正常时循环一次即退出。

3. 诊断信息前置：等待结束后无条件打印 `ls -l "$LOG"`、`wc -c "$LOG"`、`tail -c 4096 "$LOG"`，三条命令都挂 `|| true`，保证诊断命令自身失败不会把这一步判红。三条命令分别回答「文件是否存在」「字节数是否为 0」「内容是否被截断」，把原本不透明的 `ValueError` 变成可定性的证据；`tail -c 4096` 只取尾部，避免日志巨大时污染 `Actions` 输出。
4. 调用口径统一：把 `python check_npu.py --log run_qwen3_vl_8b_Instruct_fsdp2_npu/run_qwen3_vl_8b_Instruct_fsdp2_npu.log --base baseline.txt` 改为 `--log "$LOG"`，行为等价但后续改动只需改一处；`check_npu.py` 与 `baseline` 仍通过 `curl --fail --location --retry 3` 从 `${ASCEND_RECIPE_BASELINE_URL}` 拉取，本 PR 未触碰。
5. 测试与配套：无自动化测试（调度型 job 不在 PR 上运行），作者本地验证了 YAML 语法与内嵌 shell 的 `bash -n`，并确认新增命令只用容器镜像内的 POSIX 工具。作者同时说明：即使日志能被解析，该作业仍会卡在显存检查上——recipe 仓库基线文件仍是 #7428 之前的 `actor/perf/max_memory_allocated_gb: 45.43`，而 #7428 之后的运行稳定在约 41.47 GB（阈值 ± 1 GB），该基线只能在 `verl-project/verl-ascend-recipe` 侧重新生成，无法在本仓库修复。

关键文件：

- `.github/workflows/nightly_ascend.yml`（模块 夜间流水线；类别 `infra`；类型 `ci-pipeline`；符号 `Running nightlyCI_grpo_qwen3_vl_8b_Instruct_fsdp2_vllm_ascend checking script, LOG`）：唯一改动文件，新增 60s 有界等待与 `ls -l / wc -c / tail -c 4096` 诊断，并把 `check_npu.py` 的 `--log` 参数改为变量引用
- `tests/special_npu/run_qwen3_vl_8b_Instruct_fsdp2_npu.sh`（模块 冒烟脚本；类别 `test`；类型 `context-only`）：未在本 PR 中修改，但它就是产生 `tee` 日志（`"$@" | tee $LOG_DIR/$SCRIPT_NAME.log`）的脚本，是本次空日志现象的写入端，理解风险边界必须读它

关键符号：`nightlyCI_grpo_qwen3_vl_8b_Instruct_fsdp2_vllm_ascend checking script`（有界等待 + 诊断），`LOG` 非空轮询循环（`seq 1 6 / sleep 10`），`check_npu.py` 调用（`--log "$LOG" --base baseline.txt`）

关键源码片段

[.github/workflows/nightly_ascend.yml](https://github.com/verl-project/verl-ascend-recipe/blob/main/.github/workflows/nightly_ascend.yml)

唯一改动文件，新增 60s 有界等待与 `ls -l / wc -c / tail -c 4096` 诊断，并把 `check_npu.py` 的 `--log` 参数改为变量引用

```
# .github/workflows/nightly_ascend.yml —— qwen3_vl 夜间检查步骤（合并后形态）
- name: Running nightlyCI_grpo_qwen3_vl_8b_Instruct_fsdp2_vllm_ascend checking script
  run: |
    cd /root/.cache/nightly_log/

    # 把日志路径提成变量：等待循环、诊断、check_npu.py 三处共用，
    # 避免原先「硬编码路径 + 新增等待」产生两份不一致的字符串
    LOG=run_qwen3_vl_8b_Instruct_fsdp2_npu/run_qwen3_vl_8b_Instruct_fsdp2_npu.log

    # 有界等待：tee 出来的日志可能晚于训练步骤可见（k8s 自定义容器 runner）
    # 最多轮询 6 轮、每轮 10s，即 60s 上限；-s 表示「存在且非空」即刻提前退出，
```

```

# 因此日志正常时这里只付出一次 test 的代价
for i in $(seq 1 6); do
  if [ -s "$LOG" ]; then
    break
  fi
  echo "attempt $i: $LOG missing or empty, waiting 10s..."
  sleep 10
done

# 无论等到与否都打印诊断，用于区分「文件缺失 / 存在但为空 / 内容被截断」；
# 末尾 || true 保证诊断命令自身失败不会把这一步判红
ls -l "$LOG" || true
wc -c "$LOG" || true
tail -c 4096 "$LOG" || true # 只取尾部，避免大日志污染 Actions 输出

# 校验脚本与基线仍从 ascend recipe 仓库拉取，本 PR 未改动
curl --fail --location --retry 3 --output check_npu.py "${ASCEND_RECIPE_BASELINE_URL}/
check_npu.py"
curl --fail --location --retry 3 --output baseline.txt "${ASCEND_RECIPE_BASELINE_URL}/a2/
nightly_log/run_qwen3_vl_8b_Instruct_fsdp2_npu/baseline_qwen3_vl_8b_Instruct_fsdp2_npu.
txt"

# 原先这里是硬编码的同一路径字符串，现在统一走 $LOG，
# 后续再调路径只需改一处；check_npu.py 抛 ValueError 时也就
# 能和上面的 ls -l / wc -c 诊断直接对上同一个文件
python check_npu.py --log "$LOG" --base baseline.txt

```

评论区精华

本次 PR 实质讨论极少：只有 wucong25 给出一个 APPROVED 且 review body 为空的批准，没有行内 review 评论。有价值的信息几乎全部来自 PR 描述本身：(1) 明确「不是 verl bug，而是 runner 侧文件可见性」的定性，避免团队误入训练代码排查；(2) 主动披露基线漂移这一「合并后仍红」的已知后果，并声明超出本仓库修复范围；(3) 披露 AI 辅助（Kimi）并声明提交者已逐行 review；(4) 说明重复工作检查结论——未找到针对 nightly_ascend / check_npu / 「nightly log empty」的开放 PR。由于无人在评审中质疑等待时长、诊断持久化方式或根因归属，这些点仍属于「已披露但未论证」的状态。

- 只有一次无评论批准，关键取舍未经评审论证 (design)：变更以批准合并；这些取舍当前属于「已披露、未论证」，需要靠下一次夜间运行的实际证据来确认。
- PR 描述自述的遗留问题：recipe 仓库显存基线未重生成 (other)：未解决，超出本仓库范围；需要在 recipe 仓库重新生成 baseline_qwen3_vl_8b_Instruct_fsdp2_npu.txt 才能让该作业真正转绿。

风险与影响

- 风险：

1. 等待窗口固定，可能仍不够：60s 上限基于「~6s 后读到空文件」的现象外推，若 runner 落盘更慢，check_npu.py 依旧会以同样的 ValueError 失败，只是多花 60s；反过来在日志确实为空时，这一步会稳定多耗 60s，拉长夜间作业总时长。
2. 根因未消除：tee 内容丢失发生在 runner 侧或训练脚本 "\$@" | tee \$LOG_DIR/\$SCRIPT_NAME.log 的管道上，本 PR 只是吸收延迟并采集证据，没有改用 stdbuf / sync / 直接以 Actions stdout 作为检查输入；同一 runner 上的其他 Ascend 夜间作业（如 qwen3_5_2b 等链路）若有相同写日志模式，仍可能复现。
3. 诊断信息不进制品：ls -l / wc -c / tail -c 4096 只写进当前步骤日志，未上传 artifact；如果该步骤本身因 runner 异常被截断，证据可能一起丢失。
4. 合并后仍会红的预期偏差：由于基线仍为 45.43 GB，而当前实跑约 41.47 GB（阈值 ± 1 GB），即使日志等待逻辑生效，作业仍会在显存检查处失败；这容易被误读为「本 PR 无效」。
5. 缺少自动化验证：改动只在 schedule: '0 17 * * *' 触发，PR 期间无法回归；YAML 缩进或 shell 兼容性问题要等到下一次夜间运行才暴露，属于典型的 CI-only 变更盲区（bash -n 只能覆盖语法，覆盖不了 runner 行为）。- 影响：影响面局限在 CI 层面：不触碰任何训练、rollout、checkpoint 代码路径，对最终用户与训练正确性零影响。对团队而言，收益是双重的——正常路径几乎无额外开销（日志非空时循环一次即退出），失败路径则把「不透明的 ValueError」升级为可定性的文件状态证据，直接降低夜间作业的排障成本，也有助于确认问题到底出在 runner 还是训练脚本。潜在负面影响集中在可观测性预期上：作业变红时，读者需要额外理解「日志检查已加固、显存基线尚未更新」这两件事，否则容易重复排查。对 Ascend 夜间 CI 的稳定性治理来说，这是一次低风险、可回滚的增量改进，但并非该作业恢复全绿的最后一步。- 风险标记：CI-only 变更无自动化验证，等待窗口固定 60s 可能不足，根因未消除（runner 侧日志可见性），跨仓库基线漂移导致作业仍红，诊断信息未上传 artifact

关联脉络

- PR #7799 [ci] fix: nightlyCI_grpo_qwen3_5_2b_fsdp2_vllm_ascend: 同样修改 .github/workflows/nightly_ascend.yml，属于 Ascend 夜间流水线的连续维护，本 PR 的日志等待与诊断思路可直接复用到该作业。
- PR #7827 [megatron, ci] fix: configure Megatron integration in Ascend images and tests: 同期 Ascend CI/ 镜像基础设施改动（Ascend Dockerfile + Ascend 单测工作流），与本 PR 同属 Ascend 流水线可运行性治理链路。
- PR #7428（标题未在本次上下文中提供）：PR 描述直接引用：该 PR 之后 qwen3_vl 夜间作业的显存从 45.43 GB 降到约 41.47 GB，导致 recipe 仓库基线过期、作业持续红，是本 PR 之外的关键依赖。