PR #33685 完整报告

sgl-project/sglang

[NPU CI] Reorganize test output/log directory structure with workflow context

合并时间：2026-08-18 23:46

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33685

执行摘要

- 一句话：重构 NPU CI 日志输出目录，nightly 单节点改 suite 模式
- 推荐动作：值得精读，特别是以下三个设计决策：1）run_start_metadata 单点记录运行元数据并用 fromJson 分发的模式，避免了多参数透传的冗余与不一致；2）partition 并行 + concurrency group 追加 partition id 的组合，既控制 job 时长又防止并行 job 互相取消；3）测试工具对 suite 级 METRICS_DATA_FILE 的回退兼容设计，让同一份工具代码同时服务 PR 与 nightly 两种布局。建议 CI 维护者关注路径解析启发式在后续布局演进时的脆弱性。

功能与动机

PR body 明确指出动机：Restructure NPU CI output/log persistence so runs from different workflows are distinguishable by directory and re-runs never overwrite prior results, and migrate nightly single-node tests from a per-case matrix to per-suite jobs. 旧布局 {test_type}/{date}/{tc_name} 下，PR test、nightly、full test 等不同 workflow 的结果混在同一日期目录，难以追溯来源；同一 workflow 的 re-run 还会覆盖同名目录，导致历史结果丢失。

实现拆解

1. 运行元数据单点生成与传递：
   - .github/workflows/nightly-test-npu.yml 的 set-vars step 中计算 BRANCH_LABEL（优先取 github.event.pull_request.head.label，否则取 github.ref_name，再用 tr -cd 'a-zA-Z0-9._/:' + tr '/:' '--' 清洗成路径安全字符），并通过 python3 打包 run_start_metadata JSON（含 branch_label、workflow_name、create_time），create_time 用 date -u -d '+8 hours' +%Y%m%d-%H%M 生成（UTC+8 分钟精度），在 workflow 启动时记录一次，所有 job 共享同一前缀，跨午夜也不变。
   - 响应 review 中 cherryblo 反复提出的 "delete it"，PR 侧显式的 branch_label / workflow_name / create_date / timestamp inputs 最终收敛为单一 run_start_metadata JSON 传递。
2. 结构化目录落盘：
   - _npu-single-node-test-stage.yml 用 fromJson(inputs.run_start_metadata) 解析元数据，拼接 output/log 路径；github.run_id + github.run_attempt 保证重跑不覆盖；

partition job 追加 -p{id} 后缀防止结果目录冲突。

- concurrency group 追加 auto_partition_id，避免并行 partition 互相 cancel。
- nightly-test-npu-e2e-multi-node.yml 同步适配同一套路径布局。

3. nightly 单节点 suite 化：
   - 删除 nightly-test-npu-e2e-single-node.yml（-257 行），nightly-test-npu.yml 改为 nightly-perf/acc-{2,4,16}-npu-a3 等 6 个 suite jobs 加 a2 mix suite，acc-2 拆 4 个并行 partition 控制 5h 超时。
   - _npu-pr-test-stage.yml 新增 is_nightly_pipeline_job 输入：注入 --nightly --continue-on-error、预构建镜像 job 跳过依赖安装、动态补装 tabulate 等缺失依赖。
   - test/run_suite.py 注册 NIGHTLY_SUITES，并把 71 个仅 nightly 注册的用例迁到 full-{1,2,4,8,16}-npu-a3，使 nightly 与 PR 注册严格镜像。
4. 测试工具与 k8s 模板适配：
   - test_npu_performance_utils.py / test_npu_accuracy_utils.py 的 _setup_per_case_output 支持 suite 级 METRICS_DATA_FILE，per-case 结果落在 {suite}/{tc_name}；_backup_plog 用 plog_base 镜像 output 前缀；_save_metrics_json 写完后清理中间 JSONL。
   - run_npu_e2e_test.py 的 generate_metrics_json 按 output 后剩余段数解析 test_type，run_npu_e2e_test_case 从 metrics_data_file 推导 run_label 注入 k8s pod。
   - run_npu_testcase.sh 用 RUN_LABEL 构建 pod 内 log_path；k8s_multi_pd_separation.yaml.jinja2、k8s_multi_pd_mix.yaml.jinja2、k8s_single.yaml.jinja2 新增 RUN_LABEL / TROUBLE_SHOTTING env。
5. 稳健性配套：
   - install_pkg 多源 pip 重试（pypi、清华镜像、集群内缓存），REQUIRED_PIP_PKGS 按需补装。
   - 显式 set -euo pipefail，对 Ascend 外部 env 脚本用 set +u 包裹避免未定义变量（如 ZSH_VERSION）中断脚本。
   - 修复 transformers 本地 wheel 拷贝时 glob 被引号包裹不展开的问题，以及 PR 场景下 GNU tee 收到空字符串参数导致 exit 1 的 bug。
   - 校准 est_time 阶梯（acc 归一 4800s、length 分级），deepseek_v4_flash 等长时 case 单独缩短到 1200s。

关键文件：

- .github/workflows/nightly-test-npu.yml（模块 夜间流水线；类别 infra；类型 infrastructure）：NPU nightly 流水线核心编排文件：生成 run_start_metadata 单一 JSON、定义 6 个 per-suite jobs 与 a2 mix suite、替换原 per-case matrix，是本次重构的主入口。
- .github/workflows/_npu-single-node-test-stage.yml（模块 单节点模板；类别 infra；类型 infrastructure）：单节点测试复用模板：新增 nightly 模式、partition 矩阵、结构化 output/log 路径拼接与 set -euo pipefail 加固，是目录结构落盘的核心。

- .github/workflows/nightly-test-npu-e2e-single-node.yml（模块 旧单节点模板；类别 infra；类型 deletion）：被 suite 模式取代的旧 per-case 模板，整体删除（-257 行），是本次重构的核心信号。
- python/sglang/test/ascend/e2e/test_npu_performance_utils.py（模块 性能工具；类别 test；类型 test-coverage；符号 _setup_per_case_output, _save_metrics_json, _backup_plog）：性能测试基类适配 suite 级输出目录：_setup_per_case_output 支持 METRICS_DATA_FILE 回退、plog 前缀镜像、JSONL 清理，是测试工具侧的关键改动。
- python/sglang/test/ascend/e2e/test_npu_accuracy_utils.py（模块 精度工具；类别 test；类型 test-coverage；符号 _setup_per_case_output, _save_metrics_json, _backup_plog）：精度测试基类与性能工具做完全对称的目录适配，保证 accuracy suite 结果同样落在结构化布局下。
- python/sglang/test/ascend/e2e/run_npu_e2e_test.py（模块 E2E 驱动；类别 test；类型 test-coverage；符号 generate_metrics_json, run_npu_e2e_test_case）：多节点 e2e 驱动：generate_metrics_json 需按 PR/nightly 两种目录结构解析 test_type，run_npu_e2e_test_case 从输出路径推导 run_label 注入 k8s pod。
- .github/workflows/_npu-pr-test-stage.yml（模块 PR 测试模板；类别 infra；类型 infrastructure）：PR 测试模板新增 is_nightly_pipeline_job 输入，使 nightly 功能 job 可复用同一模板并注入 --nightly --continue-on-error。
- .github/workflows/nightly-test-npu-e2e-multi-node.yml（模块 多节点模板；类别 infra；类型 infrastructure）：多节点模板同步适配结构化目录，从 run_start_metadata 解析元数据并拼接输出路径，与单节点保持一致。
- python/sglang/test/ascend/e2e/run_npu_testcase.sh（模块 用例脚本；类别 test；类型 test-coverage）：pod 内测试入口脚本：用 RUN_LABEL 构建 log_path，使多节点 pod 日志落入结构化目录前缀下。
- test/run_suite.py（模块 套件执行器；类别 test；类型 test-coverage；符号 NIGHTLY_SUITES）：suite 执行器：新增 NIGHTLY_SUITES 注册表，是 nightly per-suite 模式能够按用例收集执行的支撑。
- python/sglang/test/ascend/e2e/k8s_multi_pd_separation.yaml.jinja2（模块 K8s 模板；类别 test；类型 configuration）：PD 分离多节点 k8s 模板：新增 RUN_LABEL 与 TROUBLE_SHOTTING env，保证 pod 内日志路径与 CI 结构化目录一致。
- test/registered/npu/performance/glm5_1/test_npu_glm5_1_w4a8_1p1d_32p_in64k_out1k_50ms.py（模块 性能用例；类别 test；类型 test-coverage）：GLM5.1 多节点性能用例：该 PR 顺带调整了 chunked-prefill-size、NEXTN 投机解码参数与并发度，属于 suite 迁移过程中的用例校准。

关键符号：_setup_per_case_output, _save_metrics_json, _backup_plog, generate_metrics_json, run_npu_e2e_test_case

关键源码片段

python/sglang/test/ascend/e2e/test_npu_performance_utils.py

性能测试基类适配 suite 级输出目录：_setup_per_case_output 支持 METRICS_DATA_FILE 回退、plog 前缀镜像、JSONL 清理，是测试工具侧的关键改动。

```
@classmethod
def _setup_per_case_output(cls):
    """让每个用例在 suite 输出目录下拥有独立子目录。

    当 workflow 把 METRICS_DATA_FILE 指向 suite 级目录（例如
    .../output/{branch_label}-{create_time}-{run_id}-{run_attempt}/
    {workflow_name}/{test_type}/{suite}）时，把当前用例结果写到
    {suite}/{tc_name}/ 下；没有该环境变量时回退到旧 per-case 布局。
    """
    cls.tc_name = cls._get_tc_name()
    suite_output = os.environ.get("METRICS_DATA_FILE")
    if suite_output:
        # 在 suite 输出前缀下追加用例 id，保证结果按用例归位
        cls.metrics_data_file = os.path.join(suite_output, cls.tc_name)
        # 把 output 前缀镜像成 plog 前缀（去掉 test_type/suite 两段尾部）
        suite_plog = suite_output.replace("/output/", "/logs/plog/", 1)
        cls.plog_base = os.path.dirname(os.path.dirname(suite_plog))
    else:
        # 旧布局 {test_type}/{date}/{tc_name}，等价于删除前 workflow 的行为
        current_date = datetime.now().strftime("%Y%m%d")
        test_type = getattr(cls, "test_type", "perf")
        base_output = f"/root/.cache/tests/output/{test_type}/{current_date}"
        cls.metrics_data_file = os.path.join(base_output, cls.tc_name)
        cls.plog_base = "/root/.cache/tests/logs/plog"
    os.makedirs(cls.metrics_data_file, exist_ok=True)
    # 让 evalscope / dump_metric 都写到 per-case 路径
    os.environ["METRICS_DATA_FILE"] = cls.metrics_data_file
    os.environ["SGLANG_TEST_METRICS_OUTPUT"] = os.path.join(
        cls.metrics_data_file, "metrics"
    )
    logger.info(
        "Per-case output: tc_name=%s metrics_data_file=%s",
        cls.tc_name,
        cls.metrics_data_file,
    )
```

python/sglang/test/ascend/e2e/run_npu_e2e_test.py

多节点 e2e 驱动：generate_metrics_json 需按 PR/nightly 两种目录结构解析 test_type，run_npu_e2e_test_case 从输出路径推导 run_label 注入 k8s pod。

```
def generate_metrics_json(metrics_data_file, test_case, status):
    """把测试 stdout 里的 [METRIC] 行汇总成每用例 metrics.json。

    nightly 目录比 PR 目录多出 {branch}-{date}-{run_id}-{run_attempt}/{workflow}
    两层前缀，因此 output 之后剩余段数不同，test_type 的偏移也不同。
    """
```

```
    log_file = os.path.join(metrics_data_file, "test_output.log")
    metrics, baselines = {}, {}
    if os.path.exists(log_file):
        with open(log_file, "r") as f:
            for line in f:
                m = re.match(r"\[METRIC\] (\S+)=(\S+)", line.strip())
                if m:
                    key, value = m.group(1), m.group(2)
                    try:
                        value = float(value)
                    except ValueError:
                        pass
                    # _baseline 后缀的指标单独归类，便于后续对比
                    if key.endswith("_baseline"):
                        baselines[key[:-9]] = value
                    else:
                        metrics[key] = value

    tc_name = test_case.rsplit("/", 1)[-1].rsplit(".", 1)[0]
    test_type = "unknown"
    # nightly: .../output/{branch}-{date}-{run_id}-{run_attempt}/{workflow}/{test_type}/...
    # PR: .../output/{test_type}/{date}/{tc_name}
    parts = metrics_data_file.split("/")
    for i, part in enumerate(parts):
        if part == "output":
            rest = len(parts) - (i + 1)
            # 剩余段数 >= 4 说明是 nightly 结构，test_type 在 output 后第 4 段
            if rest >= 4:
                test_type = parts[i + 3]
            elif rest >= 1:
                test_type = parts[i + 1]
            break
    # 其余逻辑不变：写 metrics.json 并同步一份到 /tmp/metrics.json
```

评论区精华

review 主要由 cherryblo 主导，核心交锋如下：

- 冗余人参删除：针对显式 branch_label / workflow_name / create_date / timestamp inputs，cherryblo 多次要求 "delete it"、"The newly added parameter has no practical usage scenarios, remove it."；最终收敛为 run_start_metadata 单一 JSON，简化模板输入契约。
- 模板统一：cherryblo 要求 "Adopt the new single-node test template." 和 "Use the workflow file .github/workflows/_npu-pr-test-stage.yml."，促使 nightly 功能 job 最终复用 _npu-pr-test-stage.yml，单节点 job 复用 _npu-single-node-test-stage.yml。
- metrics upload 回退：cherryblo 要求 "Revert this change. Metrics upload related code will not be added for now."，最终删除 suite 级 Upload metrics 步骤，依赖 nightly 日志整体收集。

- LOG_TEE_TARGET 与 tee 空参数：cherryblo 质疑 "What is the purpose of the LOG_TEE_TARGET variable? Is the usage of tee /tmp/test_output.log... correct?"；后续 commit 用数组方式避免 PR 场景传空字符串给 tee（GNU tee 空参数会 exit 1 翻转管道退出码）。
- RUN_LABEL / TIMESTAMP 用途质疑：cherryblo 问 "What are the purposes of these two input variables?" 和 "Do we require a timestamp to validate CI job reruns?"；最终保留 RUN_LABEL 用于 pod 日志目录前缀，timestamp 字段被移除（唯一性由 run_id/run_attempt 保证）。
- shell 严格模式与外部脚本冲突：commit 记录 "Fix set -u abort on Ascend env scripts"，通过临时 set +u 包裹 source 外部 env 脚本解决。
- 显式 branch_label/workflow_name/create_date/timestamp inputs 是否冗余 (design): 最终收敛为单一 run_start_metadata JSON（branch_label + workflow_name + create_time），在 workflow 启动时记录一次，经 needs 传递，删除了所有显式透传 inputs。
- nightly job 应复用哪个测试模板 (design): 单节点与 a2 suite 复用 _npu-single-node-test-stage.yml，nightly-{1,2,4,8,16}-npu-a3 等功能 job 复用 _npu-pr-test-stage.yml；两者都通过 is_nightly_pipeline_job 注入 nightly 专属行为。
- suite 级 metrics upload 是否引入 (design): 删除 suite 级 Upload metrics 步骤与 per-case log merge，nightly 日志整体收集已覆盖测试结果，避免重复造轮子。
- LOG_TEE_TARGET 与 tee 空字符串参数 (correctness): 用日志目标数组构建 tee 参数，PR 路径下不传空字符串；同时显式 set -euo pipefail 保证退出码反映 run_suite.py 真实结果。
- RUN_LABEL / TIMESTAMP 环境变量的用途 (question): RUN_LABEL 保留，用于把 pod 内 log_path 与 CI 结构化目录前缀对齐；timestamp 字段最终被移除（最后一次重构 commit 明确：目录唯一性由 run_id/run_attempt 与 tc_name 保证）。
- set -euo pipefail 与外部 Ascend env 脚本冲突 (correctness): 在 source cann / nnal / ascend-toolkit 等外部脚本前临时 set +u，source 完成后恢复 set -u，并加注释说明原因。
- partition 并行与 concurrency group 冲突 (design): concurrency group 追加 auto_partition_id，使并行 partition 可同时运行；partition job 在目录与 artifact 名上加 -p{id} 后缀避免碰撞。
- BRANCH_LABEL 变量被重复赋值 (style): 重构为单一赋值链（explicit input > PR head label > ref name），并补充脚本注入防护注释：PR 元数据经 env 变量传入避免 fork 分支名造成脚本注入。

风险与影响

- 风险：
  1. nightly-test-npu.yml 大规模重构（+289/-216）：suite 定义、runner 对齐、needs 依赖任一配置错误都会导致 nightly 整体漏跑或误跑；历史上已出现 a2 依赖未安装、suite runner 与 PR job 不一致等问题，需关注 runner 标签（a3-800t-N vs a3-N）的匹配。

2. run_npu_e2e_test.py 的路径解析依赖目录段数（rest >= 4 判定 nightly 布局）：该启发式对后续目录结构调整非常敏感，布局再变会静默解析出错误的 test_type 或 run_label，且无显式告警。
3. run_npu_testcase.sh 的 run_label 默认值是 unknown：若 RUN_LABEL env 未通过 k8s 模板注入（例如新增的 pod 类型遗漏修改 jinja2），日志会全部归拢到 unknown 目录，掩盖可追溯性改进。
4. run_suite.py 的 NIGHTLY_SUITES 与 71 个用例迁移：nightly 与 PR 注册要求严格镜像，注册遗漏会造成 nightly 覆盖偏差，且这种偏差无自动校验。
5. set -euo pipefail 在全 shell 开启后，任何外部脚本或测试工具的未定义变量 / 非零退出都会立即失败；已通过 set +u 缓解 Ascend env 脚本问题，但风险面仍较大。
6. acc-2 拆 4 partition 依赖 est_time 阶梯的准确性，划分不均可能导致某个 partition 超过 5h job 超时上限。 - 影响：影响范围集中在 NPU CI 基础设施层：所有 NPU 单节点 / 多节点 nightly 与 PR 测试的 output、log、plog 目录结构全部变更，结果可追溯性显著提升（workflow + run_attempt 均可区分）；nightly 单节点测试从大量 matrix job 收敛为少量 suite job，配合 partition 并行，单 job 时长更可控，GitHub Actions 页面不再被打爆。对使用者（模型开发者、CI 维护者）而言，_npu-single-node-test-stage.yml 与 _npu-pr-test-stage.yml 的输入契约发生破坏性变化，任何外部复用这两个模板的 workflow 都需要同步更新；对普通用户无任何模型行为影响（PR body 明确声明 accuracy/speed 均不受影响）。影响等级中等偏高，但局限在 NPU CI 域。 - 风险标记：CI 全链路重构，shell 严格模式风险，目录段数启发式解析，suite 注册映射易错，RUN_LABEL 注入依赖

关联脉络

- PR #35238 Exclude multimodal-gen NPU jobs from fast-fail cascade: 同为 NPU CI 稳定性治理，改动 _npu-single-node-test-stage.yml / _npu-pr-test-stage.yml / pr-test-npu.yml 等同一批 NPU workflow 文件，与本 PR 的模板改造互相叠加。
- PR #35162 Add deepseek_v4_flash_w8a8_8p_in32k_out1k_50ms: 新增 deepseek_v4_flash NPU 性能基准测试用例；本 PR 的 suite 注册与 est_time 校准提交中恰好调整了 deepseek_v4_flash 的 est_time（缩短至 1200s），两者同属 NPU 性能测试用例治理。
- PR #35196 [Chore] Move version tag helper to release scripts: 同为跨多个 .github/workflows 文件的基础设施重构，体现了 NPU/CI 基础设施持续收敛的趋势，可为该 PR 的大范围 workflow 改动提供参照。