PR #33724 完整报告

sgl-project/sglang

[NPU] Improve the execution efficiency and maintainability of pr-test-npu

合并时间：2026-08-08 16:42

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33724

执行摘要

- 一句话：重构 NPU PR CI 降低并发并快速失败
- 推荐动作：建议 CI 基础设施维护者精读，重点关注健康检查的根因失败过滤逻辑与 per-case 输出下沉设计，这两处体现了避免级联误杀和保证可观测性之间权衡；普通贡献者只需了解 NPU 套件命名规则即可。注意观察合并后 base-c 套件的实际稳定性，尤其是 GLM5 阈值下调是否有 flaky 依据。

功能与动机

PR body 明确指出两个动机：一是减少 pr-test-npu.yml 中的并发任务数，因为 excessive tasks 导致 K8s 调度阻塞，大量 queued tasks 无法运行；二是优化 pr-test-npu.yml 文件结构以提高可维护性。

实现拆解

1. 统一测试套件命名：修改 test/run_suite.py 中 HWBackend.NPU 套件列表，将旧的 stage-a-unit-test-npu、stage-b-test-{1,2,4,8,16}-npu-a3 全部替换为 base-a-test-1-npu-a2、base-b-test-{1,2,4,8,16}-npu-a3，并新增 base-c-test-acc-{2,4,8,16}-npu-a3 与 base-c-test-perf-{2,4,8,16}-npu-a3 共 8 个准确率 / 性能套件。
2. 合并同资源模型用例：把 test/registered/npu/ 下原先每个模型一个独立 job 的准确率与性能用例，按资源规格（2/4/8/16 卡 a3）归入上述 base-c 套件，并将各文件中的 register_npu_ci(...) 调用改为 register_npu_ci(est_time=3600, suite="base-c-test-xxx")，去掉 nightly=True 与 disabled 参数，从而大幅削减单次 PR 触发的并发 job 数。
3. 封装可复用工作流：新增 .github/workflows/_npu-single-node-test-stage.yml（单节点准确率 / 性能测试模板）与 .github/workflows/_npu-pr-test-stage.yml（PR 测试阶段模板），通过 workflow_call 接收 runner、镜像、套件名、超时等输入；.github/workflows/pr-test-npu.yml 由约 400 行缩减到 170 行左右，各 job 改为 uses: 引用上述模板。
4. 引入健康检查快速失败：在两个可复用工作流中增加 Check PR test health 步骤（actions/github-script@v8），检查同 commit 的 lint 状态与当前 run 内其他 job 的根因失败；只有当存在真实失败（已过滤级联失败和 h20 flaky job）时才 setFailed 提前终止，并支持 SKIP_PR_TEST_HEALTH_CHECK 与 bypass-fastfail label 逃生门。

5. per-case 输出下沉到测试基类：在 python/sglang/test/ascend/e2e/test_npu_accuracy_utils.py 与 test_npu_performance_utils.py 中新增 _get_tc_name、_setup_per_case_output、_save_metrics_json、_backup_plog 四个类方法，将 workflow 中原有的固定路径输出、stdout 解析和 plog 备份动作改成按用例名落盘，保证套件合并后每个用例仍有独立可定位的产物。

关键文件：

- python/sglang/test/ascend/e2e/test_npu_accuracy_utils.py（模块 测试基类；类别 test；类型 test-coverage；符号 _get_tc_name, _setup_per_case_output, _save_metrics_json, _backup_plog）：准确率测试基类新增 4 个类方法，把 per-case 输出目录、metrics 汇总、plog 备份从 workflow 下沉到测试框架，是套件合并后用例级可观测性的核心支撑。
- python/sglang/test/ascend/e2e/test_npu_performance_utils.py（模块 测试基类；类别 test；类型 test-coverage；符号 _get_tc_name, _setup_per_case_output, _save_metrics_json, _backup_plog）：性能测试基类与准确率基类同步新增 4 个类方法，保证合并后的 perf 套件也能按用例名产出 metrics 与 plog。
- .github/workflows/_npu-single-node-test-stage.yml（模块 CI 工作流；类别 infra；类型 infrastructure；符号 failedStep, rootCauseFailures, Check PR test health）：新增的单节点 E2E 复用工作流，承载准确率 / 性能套件的依赖安装、健康检查与测试执行，是降低并发和统一配置的核心载体。
- .github/workflows/_npu-pr-test-stage.yml（模块 CI 工作流；类别 infra；类型 infrastructure；符号 failedStep, rootCauseFailures, Check PR test health）：PR 测试阶段的复用工作流，抽取依赖安装、环境变量与健康检查逻辑，让 pr-test-npu.yml 主文件大幅缩减。
- .github/workflows/pr-test-npu.yml（模块 CI 工作流；类别 infra；类型 infrastructure）：主工作流从约 400 行缩减到约 170 行，各 job 改为引用 _npu-pr-test-stage.yml 与 _npu-single-node-test-stage.yml，并统一按 base-a/b/c 命名。
- test/run_suite.py（模块 套件注册；类别 test；类型 configuration）：更新 HWBackend.NPU 套件注册表，用 base-a/base-b/base-c 新命名替换旧的 stage-* 命名，并新增 acc/perf 两类共 8 个套件。
- test/registered/npu/accuracy/glm5_top64_pruned/test_npu_glm5_top64_pruned_bf16_8p_gsm8k.py（模块 模型用例；类别 test；类型 test-coverage；符号 register_npu_ci）：用例从独立 nightly 注册改为并入 base-c-test-acc-16-npu-a3 套件，并将 accuracy 阈值从 0.50 下调到 0.48。
- test/registered/npu/performance/qwen3_6_27b/test_npu_qwen3_6_27b_1p_in1024x1024_30_out1024_50ms.py（模块 模型用例；类别 test；类型 test-coverage；符号 register_npu_ci）：性能用例从 nightly 独立任务并入 base-c-test-perf-2-npu-a3 套件，注册参数简化为单行调用。

关键符号：_get_tc_name, _setup_per_case_output, _save_metrics_json, _backup_plog, rootCauseFailures

关键源码片段

.github/workflows/_npu-single-node-test-stage.yml

新增的单节点 E2E 复用工作流，承载准确率 / 性能套件的依赖安装、健康检查与测试执行，是降低并发和统一配置的核心载体。

```
// 健康检查：在正式执行测试前快速失败，节省 CI 资源。
// 跳过条件：显式 SKIP_PR_TEST_HEALTH_CHECK、定时任务（schedule）、
// 或 PR 携带 bypass-fastfail label。
if (process.env.SKIP_PR_TEST_HEALTH_CHECK === 'true') {
  core.notice('[health-check] SKIP: SKIP_PR_TEST_HEALTH_CHECK=true');
  return;
}
if (context.eventName === 'schedule') {
  core.notice('[health-check] SKIP: scheduled run');
  return;
}

// 通过 checks.listForRef 查询同一 commit 的 lint 状态
//（listJobsForWorkflowRun 只能看到当前 run 内的 job）。
const ref = context.payload.pull_request?.head?.sha || context.sha;
const { data } = await github.rest.checks.listForRef({
  owner: context.repo.owner,
  repo: context.repo.repo,
  ref: ref,
  check_name: 'lint',
});
const lintRun = data.check_runs.find(cr => cr.app?.slug === 'github-actions');
if (lintRun?.status === 'completed' && lintRun?.conclusion === 'failure') {
  core.setFailed('Fast-fail: lint check failed');
  return;
}

// 根因失败判定：过滤掉级联失败（健康检查步骤本身失败）与
// 已知 flaky 的 h20 job，避免误伤其他 stage。
const jobs = await github.paginate(github.rest.actions.listJobsForWorkflowRun, {
  owner: context.repo.owner,
  repo: context.repo.repo,
  run_id: context.runId,
  per_page: 100,
});
const rootCauseFailures = jobs.filter(j => {
  if (j.status !== 'completed' || j.conclusion !== 'failure') return false;
  // 按 job 名拆分出基础 key，兼容 inline 与 reusable 两种形式
  const baseName = j.name.split(/[ /]/)[0];
  if (baseName === 'base-c-test-8-gpu-h20') {
    core.info(`[health-check] Filtered out h20 job: ${j.name}`);
    return false;
  }
  // 如果失败步骤是健康检查本身，说明是级联失败
```

```
    const failedStep = (j.steps || []).find(s => s.conclusion === 'failure');
    if (failedStep && (failedStep.name.includes('check-pr-test-health') ||
        failedStep.name.includes('Check PR test health'))) {
      return false;
    }
    return true;
  });

  if (rootCauseFailures.length > 0) {
    core.setFailed(`Fast-fail: skipping — root cause job(s): ` +
      `${rootCauseFailures.map(j => j.name).join(', ')}`);
  } else {
    core.notice('[health-check] PASS: no root cause failures detected');
  }
```

评论区精华

本 PR 没有有效的 review 讨论线程，comments_count 中仅有的 2 条均为 sglang-npu-bot 自动触发的 /tag-and-rerun-ci 命令；设计权衡主要通过代码内注释和 120 次 commit 的演进体现（例如健康检查中过滤 h20 与级联失败的注释说明）。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 健康检查误判风险：.github/workflows/_npu-single-node-test-stage.yml 与 _npu-pr-test-stage.yml 中的根因失败判定依赖 job 名称过滤（如忽略 base-c-test-8-gpu-h20）和失败步骤名称识别级联失败，一旦 job 命名规则变化或 check run 状态未同步，可能误杀正常测试或漏判真实失败。
  2. 套件合并增大失败定位成本：多个模型用例合并进同一 base-c 套件后，单个用例崩溃可能导致整个 job 失败并重跑整个套件；虽然 _save_metrics_json 提供了 per-case 产物，但 CI 层面仍需依赖 Python 日志区分具体失败用例。
  3. 测试严格性下降：test/registered/npu/accuracy/glm5_top64_pruned/test_npu_glm5_top64_pruned_bf16_8p_gsm8k.py 中 accuracy 阈值从 0.50 下调到 0.48，可能削弱回归防线，需确认是否因 flaky 基线调整。
  4. 注册名变更范围大：60 个文件中的 register_npu_ci 套件名同步变更，若 run_suite.py 注册表与某个用例的 suite 名不一致，该用例会被静默跳过。
  5. 外部可移植性受限：工作流内硬编码了华为云镜像地址（swr.cn-southwest-2.myhuaweicloud.com）与集群内缓存服务（cache-service.nginx-pypi-cache.svc.cluster.local），fork 仓库或非 NPU 集群无法直接复用。 - 影响：影响范围集中在 NPU 硬件上的 PR CI 流程：显著降低单次 PR 触发的并发 job 数，缓解 K8s 调度排队；通过健康检查快速失败减少无效 CI 消耗；对 NPU 相关开发者而言 PR 测试周期更短、失败反馈更早。对运行时、推理性能没有影响，但涉及 CI 基础设施的维护约定（套件命名、注册方式、输出目录），后续新增 NPU 用例需要遵循新的 base-c 套件分类。 - 风险标记：CI 快速失

败可能误判，套件合并增大失败定位成本，阈值下调削弱回归防线，大量注册名变更，依赖内部镜像与缓存服务

关联脉络

- PR #34070 [CI] Trim redundant nightly test registrations: 同一时期对测试注册进行裁剪与合并，目标同为节省 CI 资源，反映测试套件治理的延续方向。
- PR #32505 [UT][NPU] add unit tests for ascend_torch_native_backend and mla_preprocess: NPU 测试体系扩展的一部分，与本 PR 共同完善 NPU CI 覆盖。
- PR #34041 Docker: install DeepEP from release wheels: 同为 CI 基础设施优化（构建与安装流程），与本 PR 的 workflow 调整处于同一演进线上。