PR #25394 完整报告

sgl-project/sglang

[CI] slash handler: lookup `runs_on` from `runner_configs.yml`

合并时间：2026-05-16 04:59

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25394

执行摘要

- 一句话：重跑测试从 YAML 动态解析 runner 标签
- 推荐动作：值得所有涉及 CI 维护的开发者精读，学习如何通过 YAML 配置驱动 workflow、移除硬编码表。重点关注 _extract_runner_config 的实现和 rerun-test.yml 的拆分模式。

功能与动机

Make runner_configs.yml the single source of truth for /rerun-test (matches the main PR test pipeline) and refactor rerun-test.yml to consume it directly. Drop the hardcoded CUDA_SUITE_TO_RUNNER stage table and the bespoke use_deepep / install_diffusion flag soup.

实现拆解

1. 扩展 slash_command_handler.py：新增 _extract_runner_config 函数，从新格式的 register_cuda_ci(stage=..., runner_config=...) 调用中提取 runner_config 名称；detect_suite 改为优先通过 _runner_configs.load() 加载 YAML 配置来查找 runs_on、install_script、install_timeout、rdma_devices。同时移除了硬编码的 CUDA_SUITE_TO_RUNNER、DEEPEP_SUITES、_LEGACY_RDMA_DEVICES 和 _extract_suite 遗留回退逻辑。添加 _ALLOWED_INSTALL_SCRIPT 正则校验，仅允许 scripts/ci/cuda/*.sh 路径。
2. 重写 rerun-test.yml：将原单 job 拆分为三个独立 job（cuda、multimodal_gen、cpu），由 mode 输入控制；CUDA job 直接接收 runs_on、install_script、install_timeout、rdma_devices 参数，不再通过 use_deepep 等标志分支。修复了部分路径错误（例如 dsv4 b200 之前错误地使用了 deepep 脚本）。
3. 迁移测试注册：将 test/registered/dsv4/ 下的 1 个文件从旧的 suite="stage-..." 改为新的 stage=..., runner_config=... 格式；移除 test/manual/ 下 27 个文件的 register_cuda_ci 和 register_amd_ci 调用及对应导入（这些文件已被 run_suite.py 忽略，属于死代码）。调整 test_priority_scheduling_disaggregation.py 的运行阶段（先移至 stage-a-test-cpu，后因传递性 CUDA 导入回退至 stage-b GPU）。
4. 更新 pr-states.yml：移除对 /rerun-failed-ci 的误导性建议。

关键文件：

- scripts/ci/utils/slash_command_handler.py（模块 CI 脚本；类别 infra；类型 infrastructure；符号 _extract_suite, _extract_runner_config, _err, detect_suite）：核心变更文件：新增 _extract_runner_config 函数，重写 detect_suite 以从 runner_configs.yml 动态解析 runner 标签，移除硬编码的 suite→runner 映射表，并添加 install_script 路径白名单校验。
- .github/workflows/rerun-test.yml（模块 CI 工作流；类别 infra；类型 infrastructure）：工作流定义文件：重构为三个独立 job（cuda/multimodal_gen/cpu），直接接收 runs_on、install_script 等参数，消除 use_deepep/install_diffusion/is_cpu 等冗余标志，与 slash 处理解耦。
- test/manual/mla/test_mla_deepseek_v3.py（模块 测试注册；类别 test；类型 test-coverage）：代表性测试文件：展示了从 test/manual/ 中移除已失效的 register_cuda_ci 调用的典型变更，这些调用对实际 CI 运行无影响但造成虚假的注册意图。

关键符号：_extract_runner_config, detect_suite, _extract_suite

关键源码片段

scripts/ci/utils/slash_command_handler.py

核心变更文件：新增 _extract_runner_config 函数，重写 detect_suite 以从 runner_configs.yml 动态解析 runner 标签，移除硬编码的 suite→runner 映射表，并添加 install_script 路径白名单校验。

```
# scripts/ci/utils/slash_command_handler.py

import re
import sys
import os

# 导入 runner_configs 模块（上级目录）
sys.path.insert(0, os.path.join(os.path.dirname(os.path.abspath(__file__)), ".."))
import runner_configs as _runner_configs

# install_script 白名单：仅允许 scripts/ci/cuda/ 下的 .sh 文件
_ALLOWED_INSTALL_SCRIPT = re.compile(r"^scripts/ci/cuda/[\w.-]+\.sh$")

def _extract_runner_config(content):
    """从新样式 register_cuda_ci(stage=..., runner_config=...) 调用中提取 runner_config 值。"""
    # 匹配行首（跳过注释）直到 register_cuda_ci(...) 的括号内容
    args = re.search(r"^[^#\n]*register_cuda_ci\s*\(([^)]*)\)", content, re.MULTILINE)
    if not args:
        return None, None
    args_str = args.group(1)
    # 在参数串中查找 runner_config="..."
    m = re.search(r'runner_config\s*=\s*["\']([^"\']+)["\']', args_str)
    if not m:
        return None, args_str
    return m.group(1), args_str
```

```
def detect_suite(file_path):
    """识别测试文件的 CI 注册配置。"""
    with open(file_path) as f:
        content = f.read()

    # 优先尝试新样式：runner_config="..."
    rc, args_str = _extract_runner_config(content)
    if rc:
        # 加载 YAML 配置（带缓存）
        configs = _runner_configs.load()
        cfg = configs.get(rc)
        if cfg is None:
            known = ", ".join(f"{k}" for k in sorted(configs.keys()))
            _err(f"runner_config '{rc}' not found in runner_configs.yml. Known: {known}")
            return None, None
        # 从 cfg 中提取必需字段
        runs_on = cfg.get("runs_on", "")
        install_script = cfg.get("install_script", "")
        # 校验 install_script 格式
        if install_script and not _ALLOWED_INSTALL_SCRIPT.match(install_script):
            _err(f"install_script '{install_script}' does not match allowed pattern")
            return None, None
        install_timeout = cfg.get("install_timeout", 20)
        rdma = cfg.get("rdma_devices", "")
        return (runs_on, install_script, install_timeout, rdma), args_str

    # 回退到旧样式（已废弃，不会走到）
    return None, None
```

评论区精华

Gemini Code Assist 在 review 中提出两点改进建议：一是加强正则健壮性，允许 register_cuda_ci 调用中存在空白字符；二是优化性能，通过缓存 YAML 配置加载结果并复用已解析的参数字符串，避免在 detect_suite 中重复文件扫描。作者在后续 commit（dfdff0fd）中采纳了这两点，添加了 _runner_configs.load() 缓存和 _extract_runner_config 返回 args_str 的实现。

- 检测函数性能优化与正则健壮性 (design): 作者在 dfdff0fd commit 中落实了两点：在正则中增加 \s* 支持空白，在 _extract_runner_config 中返回值改为 (runner_config, args_str) 元组供调用者复用。同时通过模块级变量缓存 _runner_configs.load() 的结果。

风险与影响

- 风险：核心风险在于配置迁移的完备性：若 runner_configs.yml 中缺失某些注册配置，/rerun-test 可能因找不到运行器而失败，需确保 YAML 与硬编码表完全一致。install_script 的 allow-list 校验可能阻止合法脚本路径，但已覆盖 scripts/ci/cuda/*.sh 模式，且 CI 测试已验证。test/manual 注册调用的移除不会影响实际 CI 运行（因为它们未被扫描），但开发者手动运行这些测试时可能丢失自动注册，需依靠手动指定参数。整体影响

局限于 CI 系统，不涉及运行时。

- 影响：对 CI 维护者：配置中心化，消除重复维护负担，添加新测试套件只需更新 YAML 即可。对开发者：/rerun-test 行为与主流水线一致，不再需要关心底层 runner 映射。对运行时：无影响。对 CI 工作流：rerun-test.yml 重构后更清晰，故障排查更容易。
- 风险标记：配置迁移完备性，install_script 校验限制，测试注册死代码清理

关联脉络

- PR #25264 move runs_on + rdma into runner_configs.yml: 本 PR 的前提基础：25264 创建了 runner_configs.yml 并将 runs_on 和 rdma 移入其中，本 PR 进一步让 slash handler 消费该 YAML。