PR #25138 完整报告

sgl-project/sglang

ci: extract cuda stage actions + runner_config mapping

合并时间：2026-05-14 12:16

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25138

执行摘要

- 一句话：提取 CUDA stage 为可重用工作流，统一 runner_config 映射
- 推荐动作：建议 CI 相关团队阅读，可参考其如何通过可重用工作流和外部映射文件大规模裁剪 CI 配置冗余。整体设计清晰，等价性验证方法值得借鉴。

功能与动机

减少 CI 工作流中大量重复步骤，统一安装脚本选择逻辑，将 runner_config 作为单点配置，避免手动传递 install script 路径。PR body 指出 'pulls every CUDA stage job in pr-test.yml into 3 composite actions and routes install-script selection through a single runner_config lookup key'。

实现拆解

1. 创建 runner_configs.yml：在 scripts/ci/runner_configs.yml 中定义每个 runner_config 的安装脚本路径、artifact 版本和安装超时时长，成为单点配置。
2. 创建 runner_configs.py：在 scripts/ci/runner_configs.py 中实现 CLI 工具，解析 YAML 并输出指定配置的键值对，供 composite action 运行时调用。
3. 创建 _pr-test-stage.yml：新增 .github/workflows/_pr-test-stage.yml，作为可重用工作流，包含 checkout、sgl-kernel 下载、依赖安装、测试运行、coredump 上传、venv 清理等完整步骤，通过 inputs 接收 runner_config、partitions 等参数。
4. 精简 pr-test.yml：将原有 13 个 CUDA stage 的大段 inline 逻辑替换为 uses: ./.github/workflows/_pr-test-stage.yml 的简短调用 stub，每个 stub 约 20 行，通过 runner_config 输入映射到对应的安装配置。

关键文件：

- .github/workflows/_pr-test-stage.yml（模块 CI 工作流；类别 infra；类型 infrastructure）：新增可重用工作流，集中管理所有 CUDA stage 的 setup、run、teardown 步骤，通过 18 个 inputs 参数化，是本次重构的核心抽象层。
- .github/workflows/pr-test.yml（模块 CI 主工作流；类别 infra；类型 infrastructure）：主 CI 工作流，原有的 13 个 CUDA stage 的 inline 定义被大幅删减，替换为对 _pr-test-stage.yml 的引用，每个 stage 的 caller stub 仅约 20 行。
- scripts/ci/runner_configs.py（模块 配置工具；类别 infra；类型 infrastructure；符号 load）：CLI 工具，解析 runner_configs.yml 并输出指定 runner_config 的配置键值对，

是安装脚本路由的关键环节。

- scripts/ci/runner_configs.yml（模块 运行器配置；类别 infra；类型 infrastructure）：配置数据源，定义所有 runner_config 对应的 install 脚本、artifact_version 和 install_timeout，使用 YAML 锚点减少重复。

关键符号：load

关键源码片段

scripts/ci/runner_configs.py

CLI 工具，解析 runner_configs.yml 并输出指定 runner_config 的配置键值对，是安装脚本路由的关键环节。

```
# scripts/ci/runner_configs.py
# 通过 CLI 查询指定 runner_config 的安装配置（install script, artifact version, timeout）
# 被 _pr-test-stage.yml 中的 setup 步骤调用，输出格式为 $GITHUB_OUTPUT 兼容的 key=value

import os
import sys
import yaml

# YAML 数据文件路径（与脚本同目录）
_YAML_PATH = os.path.join(os.path.dirname(__file__), "runner_configs.yml")


def load() -> dict:
    """加载 runner_configs.yml 并返回 runner_configs 字典"""
    with open(_YAML_PATH) as f:
        return yaml.safe_load(f)["runner_configs"]


if __name__ == "__main__":
    if len(sys.argv) != 2:
        sys.exit("usage: runner_configs.py <runner_config>")
    rc = sys.argv[1]
    config = load().get(rc)
    if config is None:
        sys.exit(f"unknown runner_config: {rc!r}")
    # 输出为 GitHub Actions 步骤输出格式
    for key, value in config.items():
        print(f"{key}={value}")
```

scripts/ci/runner_configs.yml

配置数据源，定义所有 runner_config 对应的 install 脚本、artifact_version 和 install_timeout，使用 YAML 锚点减少重复。

```
# scripts/ci/runner_configs.yml
# 每个 runner_config 对应一个 CUDA test stage 的安装配置
# 通过 YAML 锚点复用常见的 install 脚本路径
```

```
_anchors:
  default_install: &default scripts/ci/cuda/ci_install_dependency.sh
  deepep_install: &deepep scripts/ci/cuda/ci_install_deepep.sh
  dsv4_install: &dsv4 scripts/ci/cuda/ci_install_dsv4_dep.sh

runner_configs:
  1-gpu-small: { install: *default, artifact_version: v4, install_timeout: "20" }
  1-gpu-large: { install: *default, artifact_version: v4, install_timeout: "20" }
  2-gpu-large: { install: *default, artifact_version: v4, install_timeout: "20" }
  4-gpu-b200: { install: *default, artifact_version: v6, install_timeout: "20" }
  4-gpu-h100: { install: *default, artifact_version: v4, install_timeout: "20" }
  8-gpu-h200: { install: *default, artifact_version: v4, install_timeout: "20" }
  8-gpu-h20: { install: *deepep, artifact_version: v4, install_timeout: "20" }
  deepep-4-gpu-h100: { install: *deepep, artifact_version: v4, install_timeout: "20" }
  deepep-8-gpu-h200: { install: *deepep, artifact_version: v4, install_timeout: "20" }
  dsv4-4-gpu-b200: { install: *dsv4, artifact_version: v6, install_timeout: "30" }
  dsv4-8-gpu-h200: { install: *dsv4, artifact_version: v4, install_timeout: "30" }
```

评论区精华

无实质 review 讨论，作者独立完成并自行合并。仅有一条自动评论（gemini-code-assist）提示配额已满，以及一条 /tag-and-rerun-ci 命令。

- 无实质讨论 (other): 作者自行合并，无需要关注的争议点。

风险与影响

- 风险：等价性验证通过 Python 脚本自动比对，但可能遗漏 GHA 表达式运行时行为细微差异；未来新增 runner_config 需同步更新映射文件和 YAML 锚点，否则 CI 可能因找不到配置而失败；温足步骤（DeepGEMM / server warmup）暂留 inline，未完全中心化，但影响可控。
- 影响：对用户无直接影响；对 CI 维护者，大幅减少 pr-test.yml 冗余（-826/+227），降低修改出错概率；新增 _pr-test-stage.yml 和 runner_configs.* 成为 CI 新入口，学习成本较低。CUDA stage 行为完全等价，温足步骤和部分特例（B200 条件拆分）保持 inline。
- 风险标记：等价性依赖手动验证，新增 runner_config 需同步映射，温足步骤暂留 inline

关联脉络

- PR #25197 ci: decouple stage and runner for cuda registry: 本 PR 直接基于 #25197（已合并），#25197 解耦了 stage 和 runner 的注册，为本 PR 的 runner_config 映射提供前置条件。
- PR #25203 ci: B200 conditional split + LPT_SLOP removal (stage-c partition 8→3): 本 PR 在开发过程中合并了 #25203 的变更，B200 条件拆分逻辑被纳入 caller stub 的 runs_on 处理中。
- PR #24253 ci: combine H200 8-GPU warmup steps and surface server log on every path: 本 PR 在最终提交中同步了 #24253 的温足更新，确保 warmup 步骤与最新 CI 一致。