PR #29807 完整报告

sgl-project/sglang

Add XPU CI job monitor workflow

合并时间：2026-07-03 08:30

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29807

执行摘要

- 一句话：为 Intel XPU CI 添加每日作业监控工作流
- 推荐动作：值得对工作流模式和速率限制策略进行参考。关注 XPU 后端的团队应了解此监控机制，并在上游 query_job_status.py 变更时及时 cherry-pick。

功能与动机

使 XPU CI 的每个作业的健康状况（通过 / 失败计数、持续时间、运行器利用率）像其他后端一样可见。PR body 指出「遵循现有 job-monitor 对其他后端的工作模式，作用域限定在 XPU」。

实现拆解

1. 监控脚本：scripts/ci/utils/xpu_job_monitor.py 从 query_job_status.py fork 而来，进行 XPU 特定隔离。包含 gh CLI 可用性检查、API 调用封装、速率限制检测、工作流运行数据抓取与统计、运行器利用率报告等功能。
2. 工作流定义：.github/workflows/xpu-ci-job-monitor.yml 定义了三种触发方式：schedule（每日 UTC 0 点）、pull_request（仅带 run-ci 标签的同仓库 PR，20 分钟窗口烟雾测试）、workflow_dispatch（手动指定时间窗口和作业过滤器）。
3. 数据共享：fetch-actions-data 作业只运行一次 API 抓取，结果作为 artifact 上传，下游报告作业（custom-report、job-reports、runner-report）共享同一个 snapshot，避免 GitHub API 速率限制。
4. 动态作业发现：parse-workflows 作业使用 yq 解析 pr-test-xpu.yml 中的作业列表，自动适配新增测试作业。
5. 安全与并发控制：Fork PR 和未打标 PR 被跳过；并发组 xpu-ci-job-monitor-${{ github.ref }} 防止同一 ref 多次运行时 API 浪费。

关键文件：

- scripts/ci/utils/xpu_job_monitor.py（模块 CI 脚本；类别 infra；类型 infrastructure；符号 check_gh_cli_available, run_gh_command, is_rate_limit_error, _new_workflow_fetch_stats）：核心监控脚本，包含所有作业数据抓取和报告逻辑，自包含且与上游 fork 隔离。
- .github/workflows/xpu-ci-job-monitor.yml（模块 CI 流程；类别 infra；类型 infrastructure）：定义工作流触发器、并发控制和作业解析流程，是监控的实际入口。

关键符号：check_gh_cli_available, run_gh_command, is_rate_limit_error, _new_workflow_fetch_stats, _new_fetch_metadata, _record_workflow_fetch_failure, _record_skipped_run, parse_time

关键源码片段

scripts/ci/utils/xpu_job_monitor.py

核心监控脚本，包含所有作业数据抓取和报告逻辑，自包含且与上游 fork 隔离。

```
#!/usr/bin/env python3
"""
本文件从 scripts/ci/utils/query_job_status.py 的 1b3d8da82 提交 fork 而来，
作为 XPU 自有的副本，可安全修改而不影响上游监控。
"""

import subprocess
import json
import sys
from typing import Any

# 检查 gh CLI 是否已安装并认证
def check_gh_cli_available() -> bool:
    """Check if gh CLI is installed and authenticated."""
    try:
        result = subprocess.run(["gh", "--version"], capture_output=True, text=True)
        if result.returncode != 0:
            return False
        auth_result = subprocess.run(["gh", "auth", "status"], capture_output=True, text=True)
        if auth_result.returncode != 0:
            # auth 失败时打印提示，而非静默失败
            print("Error: gh CLI is not authenticated. Please run 'gh auth login' first.", file=sys.
            stderr)
            print(f"Details: {auth_result.stderr}", file=sys.stderr)
            return False
        return True
    except FileNotFoundError:
        print("Error: gh CLI is not installed. Please install it from https://cli.github.com/", file=sys.
        stderr)
        return False

# 通用的 gh API 调用封装，返回 JSON 解析结果
def run_gh_command(args: list[str]) -> dict:
    """Run gh CLI command and return JSON result."""
    try:
        result = subprocess.run(["gh", "api"] + args, capture_output=True, text=True)
    except FileNotFoundError:
        raise Exception("gh CLI not found. Please install from https://cli.github.com/")
    if result.returncode != 0:
        raise Exception(f"gh api failed: {result.stderr}")
```

```
    return json.loads(result.stdout)

# 判断错误是否由 GitHub 速率限制导致
def is_rate_limit_error(error: str) -> bool:
    """Check whether an API error was caused by GitHub rate limiting."""
    # GitHub 的速率限制错误消息包含 "rate limit exceeded"
    return "rate limit exceeded" in error.lower()

# 创建工作流快照的空元数据桶
def _new_workflow_fetch_stats(workflow: str) -> dict[str, Any]:
    return {
        "workflow": workflow,
        "total_runs_seen": 0,
        "runs_without_jobs": 0,
        "runs_with_failed_parse": 0,
        "job_stats": {},
    }
```

评论区精华

mingfeima询问是否需要显式安装 yq 和 jq 依赖。arathi-hlab回应两者已预装在 GitHub 托管的 ubuntu-latest 运行器上，无需额外安装。该讨论已解决，无需更改代码。

- yq/jq 依赖问题 (question): arathi-hlab 回应两者已预装在 ubuntu-24.04 运行器上，无需添加。

风险与影响

- 风险：风险较低，但存在以下方面：
  - API 速率限制：工作流通过单次抓取 + 共享 artifact 降低 API 次数，但若脚本失效可能影响其他依赖 gh CLI 的流程。
  - fork 同步：xpu_job_monitor.py 是 query_job_status.py 的 fork，未来上游修复需要手动 cherry-pick，可能遗漏重要修复。
  - 工作流稳定性：依赖 yq、jq、tabulate 等外部工具，虽然运行器预装但版本变化可能导致行为差异。
  - 影响：影响范围较小，仅涉及 XPU CI 的可观测性。用户无感知，但团队可获得每日 CI 健康报告，便于跟踪稳定性。工作流在 GitHub Actions 环境中运行，对系统无直接负载。
  - 风险标记：API rate limit, fork sync gap

关联脉络

- 暂无明显关联 PR