PR #26509 完整报告

sgl-project/sglang

[CI] runner-utilization: count in-flight queue waits + per-job status/links

合并时间：2026-05-28 07:06

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26509

执行摘要

- 一句话：修复 CI 报告排队时间低估并添加作业状态列
- 推荐动作：该 PR 值得精读，特别是 classify_job 函数的状态处理逻辑和测试设计，是处理 GitHub API 实际陷阱的典型案例。同时 review 中的 Markdown 转义建议可考虑后续跟进。

功能与动机

旧报告仅统计同时包含 started_at、completed_at 且 runner_name 非空的作业，导致排队超 4 小时的作业完全不可见，严重低估最高队列深度。PR 描述提及 GitHub API 的陷阱：排队作业的 started_at 为占位符等于 created_at，旧代码因此跳过。

实现拆解

1. 提取 classify_job 函数：在 runner_utilization_report.py 中新增 classify_job，根据 status 字段分支处理 queued、in_progress、completed 等状态，准确计算排队时间和运行区间，跳过无效作业。
2. 改进并发指标计算：修改 calculate_concurrency_metrics 使其能处理没有运行区间的排队作业，使用 queue_end 作为排队结束时间，准确计算峰值队列深度。
3. 添加状态计数和链接：新增 format_status_counts 函数，将作业按 pass/fail/cancel/running/queued 分组并用 emoji 展示。在报告摘要表格增加“Status”列，并新增“Longest Queue Waits”小节直接链接作业。
4. 添加单元测试：新建 test_runner_utilization_report.py，覆盖排队、运行中、已完成、跳过、无创建时间等场景，测试 classify_job 和 calculate_concurrency_metrics。测试纯逻辑无 API 依赖。
5. 更新 CI 工作流：在 runner-utilization.yml 中增加单元测试执行步骤，在生成报告前运行测试。

关键文件：

- scripts/ci/utils/runner_utilization_report.py（模块 CI 脚本；类别 infra；类型 infrastructure；符号 format_status_counts, classify_job）：核心逻辑修改，新增 classify_job 和 format_status_counts 函数，修复排队时间计算并添加状态展示。
- scripts/ci/utils/test_runner_utilization_report.py（模块 测试；类别 test；类型 test-coverage；符号 _job, _iso, TestClassifyJob, test_queued_job_counts_ongoing_wait）：新增单元测试覆盖各种场景，防止回归。

- .github/workflows/runner-utilization.yml（模块 CI 配置；类别 infra；类型 infrastructure）：在 CI 工作流中增加单元测试执行步骤，确保测试运行。

关键符号：classify_job, format_status_counts

关键源码片段

scripts/ci/utils/runner_utilization_report.py

核心逻辑修改，新增 classify_job 和 format_status_counts 函数，修复排队时间计算并添加状态展示。

```
def classify_job(job: dict, now: datetime):
    """Derive the queue-wait and busy interval for a single job.
    Returns a job_info dict, or None when the job neither waited for nor
    occupied a runner (skipped / cancelled-before-start / missing data).
    """
    status = job.get("status")
    runner_name = job.get("runner_name") or ""
    created_at = parse_time(job.get("created_at"))
    started_at = parse_time(job.get("started_at"))
    completed_at = parse_time(job.get("completed_at"))

    if status == "queued":
        # 排队中：忽略占位符 started_at，排队结束时间为当前时间
        queue_end, start, end = now, None, None
    elif status == "in_progress" and started_at is not None:
        # 运行中：排队结束时间为开始时间，占用持续到当前
        queue_end, start, end = started_at, started_at, now
    elif (status == "completed"
          and started_at is not None
          and completed_at is not None
          and runner_name):
        # 已完成且分配过 runner
        queue_end, start, end = started_at, started_at, completed_at
    else:
        return None # 跳过未等待也未占用 runner 的作业

    queue_time = (queue_end - created_at).total_seconds()
    duration = None
    if start is not None and end is not None:
        duration = (end - start).total_seconds()

    return {
        "queue_time": queue_time,
        "duration": duration,
        "start": start,
        "end": end,
        "labels": [label for label in job.get("labels", [])
                   if label not in DEFAULT_LABELS_TO_IGNORE | GITHUB_HOSTED_LABELS],
```

```
}
```

评论区精华

- 性能优化建议：*gemini-code-assist[bot]* 建议将忽略标签集合预计算为模块级常量 ALL_IGNORED_LABELS，避免在 *classify_job* 中重复计算集合并。该建议未在合并代码中体现，可能作者认为不影响实际运行时性能。
- Markdown 转义建议：建议对作业名称中的 |、[、] 等 Markdown 特殊字符进行转义，防止破坏表格渲染。同样未在最终代码中看到对应更改。
  - 预计算忽略标签集提升性能 (*performance*): 建议被提出，但合并代码中未采用，可能因为运行时开销可忽略。
  - Markdown 特殊字符转义 (*other*): 建议被提出，但合并代码中未体现，可能存在渲染风险。

风险与影响

- 风险：
  - GitHub API 行为依赖：*classify_job* 假设了特定 API 字段模式（如排队作业的 started_at 为占位符），若 GitHub API 行为变更可能导致逻辑失效，但测试覆盖了当前行为。
  - Markdown 渲染风险：未对作业名称进行转义，若作业名包含 | 或 [] 可能破坏报告表格或链接。
  - 时区处理：使用 *datetime.fromisoformat* 解析，依赖输入为 ISO 格式且含时区信息，若 API 返回格式不一致可能导致异常。
- 影响：
  - 用户 /CI 使用者：现在报告包含所有作业的状态（排队 / 运行 / 完成），排队时间更真实，最长排队可链接到作业，便于排查瓶颈。报告格式略有变化，需确认下游工具兼容性。
  - 团队：新增单元测试降低回归风险，CI 工作流增加测试步骤但不影响现有流程。
  - 风险标记：GitHub API 行为依赖，Markdown 渲染风险，缺少 Markdown 转义

关联脉络

- 暂无明显关联 PR