PR #24102 完整报告

sgl-project/sglang

ci: add per-host utilization view to runner-utilization report

合并时间：2026-05-01 01:05

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24102

执行摘要

- 一句话：修复 CI runner 利用率报告低估问题，提升数据完整性
- 推荐动作：值得精读。该 PR 展示了如何系统诊断并修复一个数据管道中多种导致指标失真的问题，尤其是 retry with backoff、filter=all 使用、以及线程池并发控制等模式可复用于其他 CLI 工具。

功能与动机

PR body 指出：原 runner 利用率报告的 Active hours 和 Utilization 列在忙碌天低估约 2-4 倍。原因有三：

1) 同一主机有多个重叠标签，分母重复计算但分子只计一个标签； 2) 默认 filter=latest 隐藏了重试尝试消耗的主机时间； 3) GitHub API 限流导致异常被静默捕获并返回 None，丢弃了大量运行数据。

实现拆解

1. 重写 run_gh_command：从快速失败改为指数退避重试（最多 10 次，上限 60 秒，加入随机抖动），仅对可重试错误（5xx、速率限制、网络重置）重试，4xx 除 429 外直接报错。
2. 修改 get_jobs_for_run：在 API 请求中添加 ?filter=all 参数，确保重试尝试都计入 numerator。
3. 预过滤非 GPU workflow：新增 _likely_no_gpu_jobs 函数，通过 workflow 名称快速跳过文档、lint、发布等不含 GPU 的 workflow，减少 API 调用。
4. 降低并发并添加完整性警告：线程池并发数从 20 降到 4，避免触发速率限制；收集任何失败请求，在报告头部显示“数据完整性”警告。
5. 重构标签分组：使用主机集并集替代独立计数，确保重叠标签（如 4-gpu-b200 和 4-gpu-b200-kernel）的利用率相同。

关键文件：

- scripts/ci/utils/runner_utilization_report.py（模块 CI 脚本；类别 infra；类型 infrastructure；符号 run_gh_command, _likely_no_gpu_jobs, format_report）：唯一变更文件，集中修复 runner 利用率报告的三个低估原因

关键符号：run_gh_command, get_jobs_for_run, _likely_no_gpu_jobs, format_report

关键源码片段

scripts/ci/utils/runner_utilization_report.py

唯一变更文件，集中修复 runner 利用率报告的三个低估原因

```
def run_gh_command(args: list[str], max_retries: int = 10) -> dict:
    """
    Run `gh` CLI command and return JSON result with retry logic.

    重试策略: 指数退避 + 随机抖动，最多 `max_retries` 次。
    只对可重试错误（5xx、速率限制、网络重置等）重试。
    之前的快速失败配合线程池中的 `except Exception: return None`
    导致 API 短暂故障时静默丢数据，严重低估利用率。
    """
    last_err = ""
    for attempt in range(max_retries):
        result = subprocess.run(
            ["gh", "api"] + args,
            capture_output=True,
            text=True,
        )
        if result.returncode == 0:
            return json.loads(result.stdout)
        last_err = result.stderr or "(no stderr)"

        # 可重试错误包括：HTTP 5xx, secondary rate limit, abuse detection, network reset
        retryable = any(
            s in last_err
            for s in (
                "rate limit",
                "abuse",
                "Internal Server Error",
                "502",
                "503",
                "504",
                "Bad Gateway",
                "Gateway Time-out",
                "connection reset",
                "Connection reset",
                "EOF",
                "timeout",
            )
        )
        if not retryable:
            break
        # 指数退避等待，上限 60 秒，加入随机抖动避免惊群
        delay = min(60, (2 ** attempt) + random.uniform(0, 1))
        time.sleep(delay)
```

```
raise Exception(f"gh api failed after {max_retries} attempts: {last_err[:300]}")
```

评论区精华

该 PR 没有收到 Review 评论，但作者在 Issue 评论区提供了验证工作流链接，确认输出正确。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. API 重试机制：如果所有重试都失败，将抛出异常导致整个报告生成失败，可能影响 CI 监控连续性。
  2. 并发降低：报告生成时间可能增加，尤其在 workflow 数量较多时。
  3. filter=all 数据量：包含所有重试可能使返回数据膨胀，分页循环可能增加 API 调用次数，但已设置 50 页安全上限。
  4. 非 GPU workflow 过滤：可能误过滤某些包含 GPU 但名称不匹配的 workflow，需持续维护列表。 - 影响：直接影响 CI 管理员和团队对 runner 资源利用率的认知，提供更准确的负载数据以便决策。间接影响：减少因 API 限流导致的报告缺失，提升报告可信任度。变更范围仅限 CI 报告脚本，不影响模型服务或用户请求。 - 风险标记：API 速率限制依赖，遗漏自托管 runner 统计，数据完整性警告可能被忽略

关联脉络

- 暂无明显关联 PR