PR #25255 完整报告

sgl-project/sglang

ci: read est_time from sglang-ci-stats instead of scraping CI logs

合并时间：2026-05-14 15:36

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25255

执行摘要

- 一句话：CI 脚本改为集中 model.json 数据源
- 推荐动作：建议 CI 相关开发者阅读此 PR，理解集中式数据源的设计思路——将不同工具的统计模型统一从 sglang-ci-stats 获取，消除重复抓取和数据不一致。关注 fetch_model 的容错策略和边界条件的处理方式。

功能与动机

为了集中 CI 日志抓取到 sglang-ci-stats 仓库，让每周工作流直接消费其 model.json 快照，而不是每个脚本自己运行 gh api 拉取并解析日志，从而简化脚本并确保数据一致性，同时为后续 compute_partitions 在 PR 时使用同一模型奠定基础。

实现拆解

1. 新增 fetch_model 函数，通过 curl --fail 从指定 URL 获取 model.json，失败时非零退出（工作流不会生成无操作 PR）。
2. 删除原有的 gh_api、gh_api_raw、get_workflow_id、get_scheduled_runs、get_successful_jobs、job_name_to_suite、determine_backend 等函数，移除对 gh CLI 和 statistics 库的依赖。
3. 修改主流程直接加载 model.json 中的 per-(suite, file) p90 数据，与本地测试注册文件中的 est_time 对比，调用 is_significant 和 write_summary 产出更新。
4. 修复 is_significant 和 write_summary 中当旧 est_time=0 时的除零错误，使新增测试能正常触发更新阈值判断。
5. 调整工作流 weekly-update-est-time.yml，传入 --model-url 参数，并更新 PR summary 的描述文字以反映新的数据来源。

关键文件：

- scripts/ci/update_est_time.py（模块 CI 脚本；类别 infra；类型 infrastructure；符号 gh_api, gh_api_raw, get_workflow_id, get_scheduled_runs）：核心变更文件，数据源获取逻辑由自抓 GitHub Actions 日志改为消费远程 model.json，删除大量函数，新增 fetch_model，精简脚本规模。
- .github/workflows/weekly-update-est-time.yml（模块 CI 工作流；类别 infra；类型 infrastructure）：工作流文件，增加 --model-url 参数，更新描述文字，配合脚本变更。

关键符号：fetch_model, is_significant, write_summary

关键源码片段

scripts/ci/update_est_time.py

核心变更文件，数据源获取逻辑由自抓 GitHub Actions 日志改为消费远程 model.json，删除大量函数，新增 fetch_model，精简脚本规模。

```
"""关键源码片段：scripts/ci/update_est_time.py 中的核心函数。"""
import subprocess
import json

# 默认远程 model.json URL，由 sglang-ci-stats 仓库维护。
DEFAULT_MODEL_URL = (
    "https://raw.githubusercontent.com/sgl-project/sglang-ci-stats/main/model.json"
)

def fetch_model(url):
    """通过 curl 获取 model.json，失败时抛出异常。工作流会因此非零退出，
    避免静默创建无操作 PR。"""
    out = subprocess.run(
        ["curl", "--fail", "--silent", "--show-error", "--max-time", "30", url],
        capture_output=True, text=True, check=True
    )
    return json.loads(out.stdout)

SIGNIFICANT_ABS_DELTA = 30 # 至少 30 秒才算显著
SIGNIFICANT_REL_DELTA = 0.3 # 至少 30% 的变化

def is_significant(old, new):
    """判断新旧 est_time 变化是否显著。
    注意：当 old=0 时（新测试占位），直接比较绝对值阈值，防止除零错误。"""
    delta = abs(new - old)
    if old <= 0:
        return delta >= SIGNIFICANT_ABS_DELTA
    return delta >= SIGNIFICANT_ABS_DELTA and delta / old >= SIGNIFICANT_REL_DELTA
```

评论区精华

gemini-code-assist[bot] 在 review 中指出 is_significant 和 write_summary 中当 old=0 时会抛出 ZeroDivisionError，这可能在处理新测试（est_time=0 占位）时引发崩溃。提出建议在 is_significant 中添加 if old <= 0: return delta >= SIGNIFICANT_ABS_DELTA，在 write_summary 中将百分比计算改为 pct = round(delta / old * 100) if old > 0 else 100。这些修复已在最终 commit edf5243 中被采纳。

- est_time=0 时的除零处理 (correctness): 已在后续 commit edf5243 中修复，采纳了建议。

风险与影响

- 风险：
  1. 依赖外部仓库 sglang-ci-stats：若该仓库不可用或 model.json 格式变化，工作流将失败退出（不会静默创建无操作 PR），需要及时维护。
  2. AMD/NPU 后端未被纳入 sglang-ci-stats 采集范围，脚本通过 BACKENDS 列表跳过，可能导致这些后端的 est_time 无法自动更新。
  3. 除零错误已在后续提交中修复，但类似的边界情况（如负数）需持续关注。 - 影响：对普通用户无影响；对 CI 维护者：est_time 更新流程简化，不再需要 gh CLI 鉴权和日志解析，但需确保 sglang-ci-stats 的数据及时准确。未来 compute_partitions 将复用同一数据源，形成统一的基础设施层。 - 风险标记：外部数据源依赖，AMD/NPU 未覆盖，除零错误（已修复）

关联脉络

- 暂无明显关联 PR