# PR #33329 完整报告

sgl-project/sglang

[CI] Size the CPU stage from the live partition model

合并时间：2026-08-03 13:26

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33329

## 执行摘要

- 一句话：CPU CI 阶段改用实测分区模型动态扩容
- 推荐动作：值得 CI/ 基础设施维护者精读，核心看三点：一是把步骤 timeout-minutes 作为分区驱动信号而非硬性上限的“预算反馈”设计；二是 _INLINE_SUITE_JOBS 的硬失败策略，用显式错误替代静默降级；三是 CPU（托管、弹性）与 GPU（自托管、稀缺）在 max_parallel 节流上的差异化处理。若团队后续继续推进“实时分区模型”这条线，本 PR 的字段规范与读取逻辑是很好的参照。

## 功能与动机

base-a-test-cpu 内联在 pr-test.yml 中，从未经过 _pr-test-stage.yml 分发，所以没有像其他套件那样拉取实时分区模型，run_suite.py 只能回退到 LPT: no live est (None); using in-source est_time。这些写死的 est_time 比实测 p90 低约 1.5 倍，例如 test_cargo_workspace.py 的 est_time=300 对应实测 p90 仅 80，为一个文件独占整个分片；而最差 LPT 分片实测约 600 秒，与 10 分钟预算几乎重合——最近一次运行 47/47 passed 却以 1.45 秒之差触发步骤超时。PR 的诉求正是让分片按实测 p90 而不是手写常量切分，从根上消除这类“测试全部通过但 CI 超时标红”的假阴性。

## 实现拆解

1. 获取实时分区模型（pr-test.yml）：在缓存 restore 之后新增 Fetch live partition model 步骤，仅在 partition_model_sha 非空时执行；用固定 SHA 从 raw.githubusercontent.com/sgl-project/sglang-ci-stats 拉取 model.json 到 /tmp/partition-model.json，并在 Run test 步骤追加 --partition-model-file /tmp/partition-model.json，保证所有分片基于同一份模型快照划分。
2. 内联套件预算接入（compute_partitions.py）：新增 _INLINE_SUITE_JOBS = {"base-a-test-cpu"}，load_run_timeouts 除了扫描 uses: _pr-test-stage.yml 的作业外，还从内联作业中查找唯一的 Run test 步骤并读取 timeout-minutes；找不到或数量不对时抛 RuntimeError，避免静默退回旧行为。这样 compute_partitions 可以把该步骤超时读回为每分片预算——预算驱动扇出，而不是只做硬性兜底。
3. 解除 CPU 套件固定扇出（compute_partitions.py）：把 base-a-test-cpu 从 _BASE_A_OVERRIDES 移除，其扇出改由实时模型计算：{size: 8, max_parallel: 8} 变为 {size: 10, max_parallel: 10}，并随 CPU 测试量增长自动扩容。

4. 节流策略按 runner 类型区分（compute_partitions.py）：max_parallel 改为 unthrottled = full_parallel or all(t.backend == HWBackend.CPU for t in group)；纯 CPU 套件不做 size // 3 节流，避免把托管的 ubuntu-latest 阶段串行化，自托管 GPU runner 仍保持旧节流。
5. 配套预算与回归说明（pr-test.yml）：Run test 步骤 timeout-minutes 从 10 提升到 15，匹配最差分片约 600 秒的实测；PR body 确认其他所有套件的分区输出字节一致，回归面仅限 base-a-test-cpu。本 PR 未新增单元测试，正确性由后续 pr-test 运行本身验证。

关键文件：

- scripts/ci/utils/compute_partitions.py（模块 CI 脚本；类别 infra；类型 infrastructure；符号 load_run_timeouts, compute_partitions, _BASE_A_OVERRIDES, _INLINE_SUITE_JOBS）：核心逻辑变更：移除 base-a-test-cpu 固定扇出、新增 _INLINE_SUITE_JOBS 预算读取、按 runner 类型调整 max_parallel 节流规则。
- .github/workflows/pr-test.yml（模块 工作流；类别 infra；类型 infrastructure；符号 Fetch live partition model, Run test, base-a-test-cpu）：工作流侧配套：拉取实时模型、透传 --partition-model-file、提升 Run test 预算到 15 分钟，是触发此次修复的载体。

关键符号：load_run_timeouts, compute_partitions

## 关键源码片段

### scripts/ci/utils/compute_partitions.py

核心逻辑变更：移除 base-a-test-cpu 固定扇出、新增 _INLINE_SUITE_JOBS 预算读取、按 runner 类型调整 max_parallel 节流规则。

```
# 内联在 pr-test.yml 中的套件不走可复用 stage，因此没有
# `with.run_timeout_minutes` 输入，只能从 `Run test` 步骤兜底读取。
_INLINE_SUITE_JOBS = {"base-a-test-cpu"}

def load_run_timeouts(pr_test_yml_path: str) -> dict:
    """把 pr-test*.yml 中的步骤预算映射为每个套件的分片预算。"""
    with open(pr_test_yml_path) as f:
        wf = yaml.safe_load(f)
    jobs = wf.get("jobs") or {}
    timeouts = {}

    # 常规套件：直接读可复用 stage 的 `run_timeout_minutes` 输入
    for job_id, job in jobs.items():
        if not isinstance(job, dict) or job.get("uses") != _REUSABLE_STAGE_USES:
            continue
        with_ = job.get("with") or {}
        suite = with_.get("self_name", job_id)
        timeouts[suite] = int(with_["run_timeout_minutes"])

    # 内联套件：必须且只能有一个带 timeout-minutes 的 `Run test` 步骤；
    # 找不到就抛错，宁可 CI 显式失败，也不静默退回源码内陈旧 est_time。
    for suite in _INLINE_SUITE_JOBS:
```

```
            budgets = [
                step["timeout-minutes"]
                for step in ((jobs.get(suite) or {}).get("steps") or [])
                if isinstance(step, dict)
                and step.get("name") == "Run test"
                and "timeout-minutes" in step
            ]
            if len(budgets) != 1:
                raise RuntimeError(
                    f"load_run_timeouts: inline suite {suite!r} needs exactly one "
                    f"`Run test` step with `timeout-minutes` in {pr_test_yml_path}."
                )
            timeouts[suite] = int(budgets[0])

    if not timeouts:
        raise RuntimeError(
            f"load_run_timeouts: no jobs matched uses={_REUSABLE_STAGE_USES!r} "
            f"in {pr_test_yml_path}"
        )
    return timeouts


def compute_partitions(...):
    ...
    size = max(1, ideal_size)
    # 节流只用于稀缺的自托管 GPU runner；托管 ubuntu-latest 是弹性的，
    # 纯 CPU 套件若被 `max_parallel` 节流，反而会把整个阶段串行化。
    unthrottled = full_parallel or all(
        t.backend == HWBackend.CPU for t in group
    )
    max_parallel = size if unthrottled else compute_max_parallel(size)
    result[suite] = {
        "size": size,
        "arr": list(range(size)),
        "max_parallel": max_parallel,
        ...
    }
```

## 评论区精华

该 PR 没有任何人工 review 评论（review_comments_count = 0），Issue 评论仅剩 Gemini Code Assist 的停用公告和作者触发的 /tag-and-rerun-ci 重跑命令。核心设计说明都在 PR body 的 Notes for review 中：compute_partitions 会把 Run test 步骤的 timeout-minutes 读回为每分片预算，因此它是“驱动扇出”的信号而不是兜底上限——缩短预算会得到更多分片而不是更短分片；动态套件的 max_parallel = size // 3 节流只适合稀缺的自托管 GPU runner，对托管的 ubuntu-latest 会串行化阶段，所以所有 CPU 套件保持 max_parallel = size。PR 还声明分区输出对其他套件字节一致，无未解决的技术争议。

- PR 评论仅含机器人停用公告与重跑命令 (other)：无未解决的技术争议；作者直接合并。

## 风险与影响

- 风险：
  - load_run_timeouts 从“找不到就跳过”变成“找不到就抛 RuntimeError”：任何未来新增的内联套件若缺 Run test 步骤，会直接让 CI 解析失败而非静默回退。这是有意为之，但改变了失败模式，后续工作流改动需同步维护 _INLINE_SUITE_JOBS。
  - 新步骤依赖 raw.githubusercontent.com/sgl-project/sglang-ci-stats 的可用性：虽然固定 SHA 并带 --retry 3 --retry-delay 2，但外网拉取仍有偶发失败窗口；失败时 run_suite.py 退回源码 est_time，分区质量会回退到修复前水平。
  - 扇出从 8 增至 10 意味着每次多消耗 2 个弹性 CPU runner 的并发与 Actions 分钟数，且随 CPU 测试增多还会继续上涨，CI 成本需要留意。
  - 15 分钟预算比最差实测 600 秒多出约 50% 余量，可能掩盖后续测试耗时回归（不会立刻超时，而是分片缓慢变慢后再靠扩容消化）。
  - 影响：影响面集中在 CI 基础设施的 CPU 测试阶段：base-a-test-cpu 分片从固定 8 变为模型驱动的 10（max_parallel 保持 10），步骤预算从 10 分钟提到 15 分钟，且随测试数量增加会自动扩容。其他所有套件的分区输出保持字节一致，说明该变更不会扰动 GPU/ 分发套件的调度。对日常贡献者的直接收益是 CPU 门禁更稳定，不再出现“全部测试通过却因超时被标红”的假阴性；对团队而言，CPU 测试的扩容路径变为数据驱动，后续新增测试无需手调扇出参数。
  - 风险标记：外部分区模型拉取，预算读回改为硬失败，CPU 分片扩容成本

## 关联脉络

- PR #33277 [CI] Fix stale CPU test fixtures: 同属 base-a-test-cpu/CPU 测试域：该 PR 修复了 CPU 阶段失败的过期夹具，本 PR 则修复 CPU 阶段因分区模型缺失导致的超时，是同一 CI 稳定性演进的连续步骤。
- PR #33294 test: stand up the config tiers two unit tests read from: 同为修复 main 分支 CI 失败的基础设施变更，二者都在让 CI 运行状态回归真实配置 / 真实数据而非陈旧默认值。
- PR #32392 [NPU] Add PR test cases: 同期扩大 PR CI 测试矩阵的工作流调整，共享 .github/workflows 下跑批与资源调度演进的脉络。