PR #25193 完整报告

sgl-project/sglang

ci: compute matrix partition counts from `est_time`

合并时间：2026-05-14 07:18

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25193

执行摘要

- 一句话：基于 est_time 动态计算 CI 矩阵分区数
- 推荐动作：建议 CI 维护者精读此 PR，特别是 compute_partitions.py 的实现和公式。值得关注的设计决策包括：
  - 使用 est_time 作为分区依据而非测试数量，更贴近实际耗时。
  - stage-a 硬编码作为安全网，体现对关键路径的保守策略。
  - JSON 输出结构和 GitHub Actions 的动态矩阵消费模式。
  - 在 _pr-test-check-changes.yml 中如何通过脚本判断 full-parallel 模式。

建议在后续 PR 中考虑监控分区偏差告警机制。

功能与动机

手动维护 14 个硬编码分区数组，每次新增测试或调整 suite 时需同步更新配置，易出错且维护成本高。通过基于历史执行时间的动态计算，可自动适应测试变化，降低维护工作量。

实现拆解

实现步骤如下：

1. 新增计算脚本scripts/ci/utils/compute_partitions.py，通过 discover_files 查找 test 目录下所有 .py 文件，调用 CIRegistry collect_tests 获取每个 suite 的测试列表，遍历测试的 est_time 字段，累加后按公式 size = ceil(sum(est_time) * 1.15 / 20min) 计算分区数，再根据运行模式（正常 / 高优先级）决定 max_parallel。输出 JSON 格式的 partitions 字典，每个 suite 包含 size、arr（分区索引列表）和 max_parallel。stage-a 使用硬编码覆盖（cpu=4, 1-gpu-small=1）。
2. 修改 check-changes 工作流.github/workflows/_pr-test-check-changes.yml，替换原有的 Set max-parallel 步骤为 Determine full-parallel mode，输出从多个 max_parallel 变量简化为单个 partitions JSON 字符串。同时移除 check-changes 作业输出的 max_parallel、max_parallel_small、max_parallel_2gpu 等旧变量。
3. 更新主 CI 工作流.github/workflows/pr-test.yml，在 stage-a、stage-b 和 stage-c 的 job 中，将硬编码的 expected_count 和 matrix.partition 数组替换为 fromJson(needs.check-changes.outputs.partitions)['suite-name'] 的动态引用。同时添加 max-parallel 策略限制，避免过多并行导致资源竞争。stage-a 的 wait 步骤也动态计算

预期 job 数量。

4. 测试与调整：提交历史显示作者经过多轮迭代，包括保留高优先级 full-parallel 逻辑、合并 main、调整目标时间（20min，最大 30min）、添加分区表输出到 step summary 便于调试。

关键文件：

- scripts/ci/utils/compute_partitions.py（模块 CI 工具；类别 infra；类型 infrastructure；符号 discover_files, compute_max_parallel, compute_partitions, main）：新增核心脚本，实现基于 est_time 的动态分区计算，是整个变更的数据引擎。
- .github/workflows/_pr-test-check-changes.yml（模块 CI 工作流；类别 infra；类型 infrastructure）：重构 check-changes 作业输出，用 partitions JSON 替代多个硬编码变量，并重构了判断 full-parallel 模式的步骤。
- .github/workflows/pr-test.yml（模块 CI 工作流；类别 infra；类型 infrastructure）：所有 stage job 消费 partitions JSON 实现动态矩阵分区，是动态分区生效的核心消费端。

关键符号：compute_partitions, compute_max_parallel, discover_files, main

评论区精华

无外部 review 评论，PR 作者在 body 中详细说明了设计决策：

- 分区大小公式 size = ceil(sum(est_time) * 1.15 / 20min)，LPT 最坏情况上限控制在 30min 内。
- Stage-a 保持硬编码（cpu=4, 1-gpu-small=1）作为关键入口保护，避免因数据问题阻塞 CI。
- 高优先级 PR / 调度运行使用 full-parallel（max_parallel = size），加速反馈。
- 暂无高价值评论线程

风险与影响

- 风险：主要风险包括：
  - 依赖数据准确性：如果 CIRegistry 中的 est_time 不准确或缺失，可能导致分区数不合理，拖长 CI 总耗时或碎片化。虽然公式包含 15% 缓冲，但极端偏差可能超出 30min 上限。
  - 脚本稳定性：新增的 compute_partitions.py 在 bare ubuntu-latest 上运行，需确保不依赖 torch 等重型依赖（PR 已通过直接加载 ci_register.py 避免此问题），但若 ci_register.py 发生不兼容变更，可能导致脚本失败。
  - 配置耦合：check-changes 输出变更影响所有下游 stage job，若 partitions JSON 结构在传递过程中被截断或格式错误，会导致矩阵展开失败。
  - stage-a 硬编码覆盖：stage-a-test-1-gpu-small 硬编码为 1（不分区），若该 suite 测试数量增加，可能该 job 超时，但 PR 作者认为这是有意保护入口。
- 影响：影响范围：
  - CI 系统：所有使用 pr-test.yml 的 CUDA/CPU CI 运行（包括 PR、调度、手动触发）都会采用动态分区，不再依赖人工维护的硬编码数组。

- 开发者体验：新增 suite 或调整测试时无需修改 CI 配置文件，只需在 CIRegistry 中更新 est_time，降低 CI 维护门槛。
- 分区效果：预计大部分 suite 分区数与之前一致（如 stage-b 套件），但部分 stage-c 套件可能增加 1-2 个分区，或之前单 job 的套件自动获得矩阵分发。
- 资源利用：动态 max_parallel 可避免不必要的并行限制，在资源充足时加速 CI；在高优先级时完全放开并行。
- 团队：仅影响 CI 维护者，Rust 相关工作流（如 pr-test-rust.yml）未修改。
- 风险标记：依赖 est_time 准确性，脚本稳定性风险，配置耦合，stage-a 硬编码单点

关联脉络

- 暂无明显关联 PR