

PR #39411 完整报告

sgl-project/sglang

[NPU] [CI] Fix multimodal_gen filter leakage and add daily scheduled run

合并时间: 2026-09-15 19:03

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/39411>

执行摘要

- 一句话: 修复 NPU CI 过滤泄漏并新增每日定时全量跑
- 推荐动作: 值得精读程度中等: 改动只有 14 行, 但里面有两个容易踩坑的 CI 语义点, 建议 CI/ 发布相关同学花几分钟读完。

重点关注两处设计决策: 一是用「正向通配 + 显式负向排除」替代 extglob 的取反写法, 这是修掉作业归属泄漏的关键, 也说明在 paths-filter 里写复杂 extglob 是高风险行为; 二是把 schedule 与 workflow_call 统一到 run_all_tests=true 这条兜底路径上, 与 pr-test.yml 的既有约定保持一致, 属于可复用的模式。

建议的后续动作: (a) 在 filters 块加一行注释, 明确负向规则必须排在正向规则之后, 防止回归; (b) 给 multimodal_gen 的 diffusion 规则补上 !(.mdl.ipynb) 的排除写法, 避免文档改动触发 NPU 测试; (c) 评估每日全量的机时占用, 必要时改成 nightly 或加失败通知; (d) 第一次定时触发后回看一次 Actions 结果, 确认 schedule 分支确实走了全量而不是空跑。

功能与动机

PR body 明确写了两点动机: "Fixes the job filters in pr-test-npu.yml and adds a daily schedule trigger.", 以及 "python/sglang/kernels/ops/diffusion/no longer trigger main_package jobs — now route to multimodal_gen tests only."。根因是旧规则 python/sglang/!(multimodal_gen)/!(*.md) 里的 extglob 取反只作用于 python/sglang/ 之后的第一段路径, kernels/ops/diffusion/... 不属于 multimodal_gen 这一层, 因此被误判进 main_package; 而 multimodal_gen 侧只列了 python/sglang/kernels/ops/diffusion/triton/npu_fallback.py 单个文件, diffusion 内核的其余文件既落在错误的作业里、也缺少对应覆盖。定时部分 body 说明: "Scheduled runs set run_all_tests=true (same convention as pr-test.yml), since paths-filter can't diff on schedule events.", 即 NPU 流水线需要每天自动跑一次全量做兜底回归, 而不是只依赖人工 workflow_dispatch。

实现拆解

1. 触发入口扩展: 在 .github/workflows/pr-test-npu.yml 的 on: 块中新增 schedule: - cron: '0 12 * * *', 使该流水线除 push、pull_request、workflow_dispatch、workflow_call 之外每天 12:00 UTC 自动运行一次。这直接影响 NPU runner 的排队与机时预算, 因为定时运行会以 main 分支代码为对象执行全套 NPU 测试。

2. 运行模式判定改造：Determine run mode 步骤 (id: run-mode) 原先只根据 `inputs.run_all_tests` 判断是否全量，现在条件改为 `github.event_name == 'schedule' || inputs.run_all_tests == 'true'`。原因是 `schedule` 事件没有 `base/head` 变更集，`dorny/paths-filter` 无法产出 `diff`，若不强制全量就会被过滤成空跑。同时把日志从 `Run mode: ALL TESTS (run_all_tests=...)` 扩写为同时打印 `schedule` 与 `run_all_tests` 两个来源，便于排查为何走了全量。原有注释 "`github.event_name` is inherited from caller" 说明 `workflow_call` 的判定语义保持不变。
3. `main_package` 过滤语义修复：把 `python/sglang/!(multimodal_gen)/!(*.md)` 换成正向规则 `python/sglang/!(.md)` 加两条显式排除规则 `!python/sglang/multimodal_gen/` 与 `!python/sglang/kernels/ops/diffusion/*`。这里依赖 `dorny/paths-filter` 「后匹配规则生效」的顺序语义，所以排除项必须排在正向项之后；修复后 `diffusion` 内核改动不再触发 `main_package` 作业。副作用是覆盖范围较旧规则略有扩大（例如 `python/sglang/` 根目录下的非 `markdown` 文件现在也算命中）。
4. `multimodal_gen` 过滤扩容：由单文件 `python/sglang/kernels/ops/diffusion/triton/npu_fallback.py` 改为目录通配 `python/sglang/kernels/ops/diffusion/**`，与第 3 步互补，保证 `diffusion` 内核下非 `triton` 子目录的改动也能拉起 `multimodal_gen` 测试。该规则未排除 `.md` / `.ipynb`，`diffusion` 目录下的纯文档改动会带来少量无效触发。
5. 测试与部署配套：本 PR 没有源码与单测改动，验证完全依赖 CI 自身；`main_package` 中既有的 `python/pyproject_npu.toml`、`scripts/ci/npu/npu_ci_install_dependency.sh`、`test/registered/npu/`、`test/registered/unit/npu/`、`.github/workflows/pr-test-npu.yml` 等触发项保持原样，`multimodal_gen` 一侧的 `python/sglang/srt/**` 等条目也未调整，说明改动范围被刻意限制在过滤矩阵本身。

关键文件：

- `.github/workflows/pr-test-npu.yml`（模块 持续集成；类别 `infra`；类型 `configuration`；符号 `on.schedule`, `Determine run mode (id: run-mode)`, `filters.main_package`, `filters.multimodal_gen`）：本次唯一改动文件，也是理解 NPU 流水线触发矩阵的唯一入口：既重写了 `main_package / multimodal_gen` 的路径过滤（修正 `diffusion` 内核改动的作业归属），又新增每日定时触发并把 `schedule` 纳入 `run_all_tests` 判定。

关键符号：未识别

评论区精华

评审过程极简：唯一的技术讨论通道是一条 `APPROVED` 记录（reviewer: ping1jing2，无正文），以及一条 issue 评论 `/tag-and-rerun-ci`（同样来自 ping1jing2），没有任何 review comment 或行级讨论。

从 PR body 的 CI States 看，Latest PR Test (Base) 为 🟢 (Run #34844768972)，Latest PR Test (Extra) 为 🟢 (Run #34844768738)，AMD ROCm 10 未找到对应运行记录，最终由 `sglang-npu-bot` 合并。

也就是说：过滤矩阵的语义修复只经过一次人工批准，没有围绕「负向规则顺序」「覆盖范围是否被意外放大」「每日全量对 NPU 机时的挤占」展开讨论，这些点在评论区内属于未讨论状态（详见风险分析）。

- 过滤规则泄漏修复方案未经逐行评审 (design): 方案按作者写法合入; 负向规则必须排在正向规则之后这一隐式约束没有被文档化, 属于已合入但未固化的设计风险。
- CI 重跑验证与合并依据 (testing): 以 Base PR Test 绿灯加一次 APPROVED 完成合并; schedule 分支与新的过滤矩阵在合并前未做端到端验证, 需在 main 上首次定时触发后回看。

风险与影响

- 风险:

1. 过滤规则顺序敏感、缺少固化手段: main_package 的正确性依赖!

python/sglang/multimodal_gen/与 !python/sglang/kernels/ops/diffusion/出现在正向规则 python/sglang/*/(.md) 之后。该约束目前只存在于 YAML 书写顺序里, 没有注释、也没有任何校验; 后续若有人在负向规则之前 append 新正向规则, 泄漏会静默回归, 且因为 CI 仍能变绿而很难被发现。

1. 触发覆盖面被放大: python/sglang/*/(.md) 比旧规则 python/sglang/!(multimodal_gen)/*/(.md) 更宽, 旧规则要求 sglang 与文件名之间至少有一层目录, 新规则可以命中更浅层的文件; 这会提高 main_package 作业的触发频率, 在 NPU 机时紧张时表现为排队时间变长。
2. 定时全量的资源占用与失败可见性: 每日 12:00 UTC 的 schedule 会把 NPU 全套测试跑一遍, 且只会作用于默认分支 (GitHub 约定), 实际等价于「NPU 主线每日回归」。如果 NPU 侧在 main 上长期红, 本 PR 没有引入任何失败通知或结果汇总机制, 需要人工去看 Actions 页面; 同时它与仓库内其他定时流水线 (如 pr-test.yml 的同类约定) 在时间窗上是否冲突, 材料中无法确认。
3. 验证深度有限: PR 自述的 CI States 显示 Extra 运行未通过 (🔴), 合并依据主要是 Base PR Test 绿灯 + 一次 APPROVED。对纯 CI 配置改动而言可以接受, 但也意味着 schedule 分支与新的过滤矩阵在合并前并未被端到端验证, 真正的效果要在 main 上第一次定时触发后才能确认。
4. 语义噪声: multimodal_gen 的 python/sglang/kernels/ops/diffusion/**未排除 .md / .ipynb, diffusion 目录下纯文档改动会触发 NPU 多模态测试, 属于低危害的机时浪费。- 影响: 用户侧: 无任何运行时行为变化, 已部署的 NPU 推理服务不受影响。

开发者侧: NPU 相关改动的 CI 归属被修正——改 python/sglang/kernels/ops/diffusion/**的开发者在 PR 中只会看到 multimodal_gen 作业, 而不再看到一个与之无关的 main_package 作业被拉起, 反馈更准确, 也减少了重复的 NPU 机时消耗。

团队与流程侧: NPU 流水线从「PR / 人工触发」扩展到「每天自动全量」, 等于把 NPU 主线回归的责任固化到流水线。收益是回归暴露更及时, 代价是每天固定吃掉一份全量 NPU 机时预算; 对依赖共享 NPU 集群的其他流水线存在轻度排队挤压风险。

可观测性侧: Determine run mode 的日志现在会打印 schedule 与 run_all_tests 两个来源, 排查「为什么这次跑了全量」时的可诊断性提升。

- 风险标记: CI 触发矩阵变更, 过滤规则顺序敏感未固化, 每日全量占用 NPU 机时, 触发覆盖面被放大, 合并前未端到端验证, 无定时失败通知

关联脉络

- PR #39553 [CI] Run test_unified_radix_cache_kl_dcp on cutedsl_mla with bf16 KV cache: 同属「调整 CI 测试路由 / 后端选型」的同类改动，都是通过修改测试触发与执行范围来对齐资源与覆盖需求，可对照理解本仓库 CI 配置维护的节奏。
- PR #39544 [misc] Trim redundant variants from the 8-gpu-h20 disaggregation test suite: 同样以控制 CI 机时为目标（精简冗余测试变体），与本 PR 修过滤泄漏、避免重复拉起 main_package 的动机一脉相承。
- PR #39545 [CI] Wait for a killed test server's GPU memory before the next launch: 同属 CI 基础设施与测试稳定性维护线，涉及 python/sglang/test 侧公共设施；本 PR 的 schedule 每日全量会对这类测试稳定性改造形成持续压力测试。
- PR #39423 [NPU][Bugfix] Disable pinned memory to fix DeepSeek-V2 DP-attention hang: 同属 NPU 平台线改动。该 PR 改的是 NPU 运行时行为，正是本 PR 定时全量回归要持续覆盖的目标对象，两者在 NPU 侧回归链路上互补。