

PR #38842 完整报告

sgl-project/sglang

Revert "[CI] Temporarily disable GB300 tests"

合并时间: 2026-09-10 17:41

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/38842>

执行摘要

- 一句话: 回滚 #38770, 恢复 GB300 相关的 CI 与发布 workflow 任务
- 推荐动作: 值得快速过一遍而非精读: 变更本身是机械取消注释, 逻辑直白, 核心价值在于理解两点设计决策。第一, #38770 采用「整段注释 + 保留 `run_suite.py` 注册与 `runner_configs.yml` 条目 + 保留 `rust-ext-build-aarch64` 缓存」的临时禁用范式, 使恢复成为零冲突 revert, 这值得作为团队处理「runner 故障需临时摘除 CI 阶段」的标准做法。第二, 本次回滚暴露了基础设施侧的未闭环问题: 建议跟进 runner 自愈 / 健康检查 (例如把 runner 安装到持久层、或在 label 无 runner 时让 stage 快速跳过而不是无限排队), 并澄清同批 4-gpu-b200 上 `test_kda_prefill.py`、`test_numa_utils.py` 的失败是否与本改动相关。

功能与动机

38770 的正文明确说明了禁用原因: No GB300 runners are online, so every GB300 job sits queued indefinitely and holds up pr-test-finish on every PR.——GB300 runner 全部离线导致相关 job 无限排队, 进而阻塞每个 PR 的 pr-test-finish 汇总门禁。该 PR 同时声明 The tests stay registered in `run_suite.py` and `runner_configs.yml` keeps its 4-gpu-gb300 entry, so re-enabling is a straight revert of this commit, 即临时禁用被设计成可原样回滚。本 PR 正是在 runner 恢复后执行这一回滚, 把 GB300 (Blackwell Ultra / SM100) 与 aarch64 构建路径重新纳入 PR 与 nightly 的回归覆盖。

实现拆解

1. 恢复 PR 侧 GB300 测试阶段 (`.github/workflows/pr-test.yml`): 取消 `base-c-test-4-gpu-gb300` 整段注释, 恢复 `needs: [check-changes, call-gate, wait-for-base-b, sgl-kernel-build-wheels, rust-ext-build-aarch64]`、`runner_config:`

4-gpu-gb300、run_timeout_minutes: '30'、timeout_per_file: '1800'，并保留 rust_ext_artifact: `${{ needs.rust-ext-build-aarch64.outputs.artifact_name }}`——它是全流程唯一的 aarch64 阶段，因此必须取 aarch64 产物而非默认的 rust-ext-build。

2. 恢复合入门禁依赖 (.github/workflows/pr-test.yml)：pr-test-finish 的 needs 列表重新加入 base-c-test-4-gpu-gb300（位于 base-c-test-4-gpu-b200 与 base-c-test-8-gpu-b300 之间）。这一步决定了 GB300 结果重新成为 PR 合入约束，也意味着 GB300 队列一旦阻塞会再次拖住所有 PR。
3. 恢复夜间回归 (.github/workflows/nightly-test-nvidia.yml)：取消 nightly-4-gpu-gb300 job 注释，恢复其 if 中的 runner_filter 匹配 (' / 'all' / '4-gpu-gb300')、runner_config: 4-gpu-gb300、scheduled: true、run_timeout_minutes: '360' / job_timeout_minutes: '420'；同时在两个夜间汇总 job 的 needs 列表（约第 265、312 行）中重新加入 - nightly-4-gpu-gb300。
4. 恢复发布产物矩阵 (.github/workflows/release-whl-deepgemm.yml)：把 sm100-aarch64 矩阵项 (runner: 4-gpu-gb300、wheel_arch: aarch64) 从注释中恢复，使 aarch64 DeepGEMM wheel 重新进入发布构建。
5. 配套项未改动，属于有意为之：#38770 保留了 run_suite.py 中的测试注册与 runner_configs.yml 的 4-gpu-gb300 条目，rust-ext-build-aarch64 也被刻意留在原地以维持 aarch64 缓存，因此本 PR 无需任何测试或配置同步改动，回滚面收敛为 3 个 workflow 文件。验证侧没有新增测试，而是通过 /rerun-test 触发 6 个 GB300 相关用例（如 registered/ep/test_flashinfer_a2a.py、registered/e2e/models_large/test_deepseek_v3_cutedsl_4gpu.py、registered/utils/test_numa_utils.py）确认 runner 可用。

关键文件：

- .github/workflows/pr-test.yml（模块 CI 工作流；类别 infra；类型 infrastructure；符号 base-c-test-4-gpu-gb300, pr-test-finish, rust-ext-build-aarch64）：回滚的主战场：恢复 base-c-test-4-gpu-gb300 阶段并把它重新挂回 pr-test-finish 的 needs，直接决定 GB300 是否重回 PR 合入门禁路径。
- .github/workflows/nightly-test-nvidia.yml（模块 夜间测试；类别 infra；类型 infrastructure；符号 nightly-4-gpu-gb300, check-changes）：恢复 nightly-4-gpu-gb300 夜间回归任务及其在两个夜间汇总 job 中的 needs，使 GB300 重新进入每日回归覆盖。
- .github/workflows/release-whl-deepgemm.yml（模块 发布构建；类别 infra；类型 configuration；符号 sm100-aarch64）：恢复 sm100-aarch64 wheel 矩阵项，让 aarch64 DeepGEMM wheel 重新进入发布构建，同时把发布链路重新绑定到 4-gpu-gb300。

关键符号：base-c-test-4-gpu-gb300, pr-test-finish.needs, nightly-4-gpu-gb300, release-whl-deepgemm matrix: sm100-aarch64

关键源码片段

[.github/workflows/pr-test.yml](#)

回滚的主战场：恢复 base-c-test-4-gpu-gb300 阶段并把它重新挂回 pr-test-finish 的 needs，直接决定 GB300 是否重回 PR 合入门禁路径。

```

# .github/workflows/pr-test.yml (回滚后恢复的 PR 侧 GB300 阶段)
# GB300 (Blackwell Ultra) 唯一的 aarch64 测试阶段: #38770 曾把这一段整体注释, 本 PR
原样还原
base-c-test-4-gpu-gb300:
  # 依赖链与其它 base-c 阶段保持一致: 门禁、B 组预热、内核 wheel、aarch64 rust 扩展
  needs: [check-changes, call-gate, wait-for-base-b, sgl-kernel-build-wheels, rust-ext-build-
aarch64]
  # 任一上游失败或取消时不再调度, 避免占用稀缺的 GB300 runner
  if: ${{ !failure() && !cancelled() }}
  uses: ../github/workflows/_pr-test-stage.yml
  with:
    self_name: base-c-test-4-gpu-gb300
    # runner_config 直接对应 runner_configs.yml 中保留的 4-gpu-gb300 条目 (#38770
    特意未删除)
    runner_config: 4-gpu-gb300
    check_changes: ${{ toJson(needs.check-changes.outputs) }}
    caller_inputs: ${{ toJson(inputs) }}
    partitions: ${{ needs.check-changes.outputs.partitions }}
    run_timeout_minutes: '30'
    timeout_per_file: '1800'
    # 全流程唯一的 aarch64 阶段, 因此取 aarch64 构建产物, 而不是默认 rust-ext-build 的产物
    rust_ext_artifact: ${{ needs.rust-ext-build-aarch64.outputs.artifact_name }}
    secrets: inherit

# 汇总门禁: 恢复该依赖后, GB300 结果重新计入 PR 合入条件;
# 代价是 gb300 队列一旦阻塞, pr-test-finish 会被拖住 (这正是 #38770 临时禁用的原因)
pr-test-finish:
  needs: [
    # ... 其余 base-c 阶段省略 ...
    base-c-test-4-gpu-b200,
    base-c-test-4-gpu-gb300, # 本次恢复的依赖项
    base-c-test-8-gpu-b300,
  ]
  if: always()

```

[.github/workflows/nightly-test-nvidia.yml](#)

恢复 nightly-4-gpu-gb300 夜间回归任务及其在两个夜间汇总 job 中的 needs, 使 GB300 重新进入每日回归覆盖。

```

# .github/workflows/nightly-test-nvidia.yml (回滚后恢复的夜间 GB300 任务)
# 与 PR 侧同名阶段复用同一个 _pr-test-stage.yml, 只是触发方式变成定时
nightly-4-gpu-gb300:
  needs: check-changes
  # 仅在主仓库、且 runner_filter 未把 gb300 过滤掉时才调度
  if: github.repository == 'sgl-project/sglang' && (inputs.runner_filter == '' || inputs.runner_
  filter == 'all' || inputs.runner_filter == '4-gpu-gb300')
  uses: ../github/workflows/_pr-test-stage.yml
  with:
    self_name: nightly-test-4-gpu-gb300

```

```
runner_config: 4-gpu-gb300
check_changes: ${{ toJson(needs.check-changes.outputs) }}
caller_inputs: ${{ toJson(inputs) }}
partitions: ${{ toJson(needs.check-changes.outputs.partitions) }}
# 夜间任务比 PR 侧给更宽的预算：整轮 360 分钟、单 job 420 分钟
run_timeout_minutes: '360'
job_timeout_minutes: '420'
scheduled: true
secrets: inherit
```

两处夜间汇总 job 的 needs 列表（文件中约第 265、312 行）也同步恢复该任务，
例如 nightly-8-gpu-h200 之前的注释项重新变为 "- nightly-4-gpu-gb300"

评论区精华

本 PR 没有行级 review 评论，只有 Fridge003 的空正文 APPROVED；实质讨论发生在关联 issue 上：

```
mmangkad: /rerun-test test/registered/ep/test_flashinfer_a2a.py
test/registered/e2e/models_large/test_deepseek_v3_cutedsl_4gpu.py
test/registered/disaggregation/test_disaggregation_aarch64.py
test/registered/unit/multimodal/test_cuda_vmm_transport.py
test/registered/kernels/ops/attention/test_kda_prefill.py
test/registered/utils/test_numa_utils.py
```

```
github-actions[bot]: 4-gpu-gb300 (6 tests) : 🟢; 4-gpu-b200 (2 tests) : 🟡
```

即作者用一次针对性的重跑作为「runner 已恢复、可以回滚」的证据：GB300 上 6 个用例全部通过，这是合并的直接依据。作为对照，PR body 中 PR Test (Base) 与 PR Test (Extra) 均标记为 🟡 但 PR 仍被合入，说明那些失败大概率来自其它 job（细节在材料中未给出，存在不确定性）。未被讨论的疑点是同批触发的 4-gpu-b200 失败（test_kda_prefill.py、test_numa_utils.py），材料中未说明其与本次回滚是否相关。

- 用 /rerun-test 在 GB300 上验证 runner 恢复 (testing): GB300 runner 已可正常调度，构成本次 revert 的合并依据；Fridge003 空正文 APPROVED。
- 同批重跑中 4-gpu-b200 失败未澄清 (question): 未解决：无法判断是环境性 flake 还是真实回归，也未确认与本次 GB300 回滚是否相关。
- 临时禁用→原样回滚的 CI 摘除范式 (design): 该范式被验证有效：回滚零冲突、无遗漏，值得作为团队临时摘除 CI 阶段的参考做法。

风险与影响

- 风险：
 1. 单点 runner 依赖回归（高）：base-c-test-4-gpu-gb300 重新挂进 pr-test-finish 的 needs，#38770 记录的失效场景（job 无限排队、拖住所有 PR 的汇总门禁）会随之回归。更关键的是根因未在本 PR 中处理：#38770 指出两个 runner 各自独立掉线，其中一个的 runner 只安装在容器的可写层，pod 被删除后无法自愈；本 PR 只恢复调度，没

有加入 runner 健康检查或超时降级。

2. 发布链路耦合（中）：release-whl-deepgemm.yml 的 sm100-aarch64 矩阵项重新指向 4-gpu-gb300，若该 label 再次无可用 runner，aarch64 DeepGEMM wheel 构建会同样卡住而非失败。
3. 验证证据不完整（中）：合并依据是 /rerun-test 的 6 个用例，属于点状抽样而非全量 stage 跑通；同批 4-gpu-b200 的 2 个失败未澄清来源，若为真实回归，回滚后不会被 GB300 路径覆盖到。
4. 性能 / 成本：GB300 阶段 run_timeout_minutes: '30'、timeout_per_file: '1800'，恢复后 PR 流水线重新引入一段依赖稀缺 GPU 的串行等待，PR 端到端时延会上升。
5. 安全：本次改动只涉及 workflow 调度与注释，未触及权限、secrets 或镜像内容，无新增安全面。- 影响：对系统：GB300/Blackwell (SM100) 与 aarch64 rust 扩展路径的 PR 与 nightly 回归保护重新生效，run_suite.py 中已注册但长期未被调度的 GB300 测试重新获得执行机会；同时 aarch64 DeepGEMM wheel 恢复发布。对团队：所有 PR 的 pr-test-finish 再次依赖 4-gpu-gb300，若 runner 再次掉线，影响会从「GB300 无覆盖」放大为「全仓库 PR 排队」。对用户：无直接运行时行为变化，属 CI 基础设施层的可观测性与发布保障恢复。- 风险标记：CI 门禁单点 runner，队列阻塞复发风险，发布链路依赖 GB300，b200 重跑失败未澄清，缺少 runner 健康检查

关联脉络

- PR #38770 [CI] Temporarily disable GB300 tests: 本 PR 的直接回滚对象；其正文记录了两个 GB300 runner 独立掉线的根因、注释掉的三处位置，并声明重新启用等价于一次 revert，本 PR 逐条对应恢复。
- PR #37495 [AMD] ci: move the miles nightlies from rocm700 to rocm10: 同属 runner 标签生命周期管理类改动，反映仓库在 nightly/runner 迁移与开关上的固定套路，可与本次「禁用→恢复」的节奏对照。
- PR #38694 [AMD][CI] Drop the dead miles ROCm 7.0 nightly image build: 同样是针对失效 runner/ 镜像的 workflow 维护：该 PR 选择删除死工作流，而 #38770 选择注释保留以支持原样回滚，两种取舍可对比参考。