PR #28190 完整报告

sgl-project/sglang

fix(precision): do not promote failed runs to the comparison baseline

合并时间：2026-07-02 09:19

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/28190

执行摘要

- 一句话：修复失败基线晋升为比较基准的 bug
- 推荐动作：值得精读，尤其是对于需要维护自动化回归测试的团队。该 PR 展示了一个微妙的 bug（基线晋升掩盖回归）及其简洁修复，以及测试如何精准覆盖边界。设计决策关注点：用两行代码解决根本问题，保持向后兼容，测试覆盖所有场景。

功能与动机

夜间精度回归测试的核心功能是持续检测回归。然而，_select_latest_run 只根据 push_index 选择最新 run，忽略了 pass_label。当某次运行回归失败后，会上传失败张量带 pass_label="failed"，但下次运行会选中最新的失败 run 作为基准，导致即使回归仍然存在，测试也会通过（因为与已回归的基准比较）。这导致持续回归在一天后变成绿色，完全违背了测试设计初衷。PR body 详细描述了 N→N+1→N+2 的掩盖过程。

实现拆解

1. 在 python/sglang/test/precision_baseline_store.py 的 _select_latest_run 函数中，在排序候选列表后，首先过滤出 pass_label != "failed" 的可用行列表，然后优先选择可用行中的最新一条；只有在没有可用行时才回退到全部候选列表。
2. 核心改动仅两行：usable = [c for c in candidates if c[1].get("pass_label") != "failed"] 和 chosen = usable or candidates。
3. 在 test/registered/unit/test_precision_baseline_store.py 的 TestSelectLatestRun 类中新增 4 个测试方法：
   - test_prefers_older_passed_over_newer_failed：验证旧但通过的 run 优于新失败的 run。
   - test_prefers_baseline_established_over_newer_failed：验证 baseline_established 标签的 run 优于新失败的 run。
   - test_falls_back_to_failed_when_only_failed：验证当所有候选都失败时，仍然返回最新失败 run。
   - test_missing_pass_label_treated_as_usable：验证没有 pass_label 的旧行被视为可用（向后兼容）。

关键文件：

- python/sglang/test/precision_baseline_store.py（模块 测试工具；类别 test；类型 test-coverage）：核心修复文件，修改 _select_latest_run 函数，在排序后过滤掉失败的 run，仅在所有候选都失败时回退。
- test/registered/unit/test_precision_baseline_store.py（模块 测试工具；类别 test；类型 test-coverage；符号 test_prefers_older_passed_over_newer_failed, test_prefers_baseline_established_over_newer_failed, test_falls_back_to_failed_when_only_failed, test_missing_pass_label_treated_as_usable）：新增 4 个单元测试，全面覆盖新逻辑的各种场景：优先选择老的通过 run、优先选择 baseline_established run、仅失败时回退、缺少 pass_label 的向后兼容。

关键符号：_select_latest_run, test_prefers_older_passed_over_newer_failed, test_prefers_baseline_established_over_newer_failed, test_falls_back_to_failed_when_only_failed, test_missing_pass_label_treated_as_usable

关键源码片段

python/sglang/test/precision_baseline_store.py

核心修复文件，修改 _select_latest_run 函数，在排序后过滤掉失败的 run，仅在所有候选都失败时回退。

```
def _select_latest_run(
    rows: list[dict[str, Any]],
    *,
    model: str,
    capture_signature: Optional[str] = None,
) -> Optional[str]:
    # ... 前面的过滤逻辑不变 ...
    candidates.sort(key=lambda kv: kv[0])

    # 过滤掉失败的 run，仅在无可用基线时才回退到失败 run
    # 这防止了一次回归失败后，失败张量成为下次比较的基准
    usable = [c for c in candidates if c[1].get("pass_label") != "failed"]
    chosen = usable or candidates
    return chosen[-1][1]["run_path"]
```

test/registered/unit/test_precision_baseline_store.py

新增 4 个单元测试，全面覆盖新逻辑的各种场景：优先选择老的通过 run、优先选择 baseline_established run、仅失败时回退、缺少 pass_label 的向后兼容。

```
def test_prefers_older_passed_over_newer_failed(self):
    # 先有一个老的通过 run，后有一个新的失败 run
    # 应返回老的通过 run，而不是新的失败 run
    rows = [
        {"model": "org/m", "run_path": "good", "pass_label": "passed", "push_index": 1},
        {"model": "org/m", "run_path": "bad", "pass_label": "failed", "push_index": 2},
    ]
    self.assertEqual(hfs._select_latest_run(rows, model="org/m"), "good")
```

```
def test_prefers_baseline_established_over_newer_failed(self):
    rows = [
        {"model": "org/m", "run_path": "seed", "pass_label": "baseline_established", "push_index": 1},
        {"model": "org/m", "run_path": "bad", "pass_label": "failed", "push_index": 2},
    ]
    self.assertEqual(hfs._select_latest_run(rows, model="org/m"), "seed")

def test_falls_back_to_failed_when_only_failed(self):
    rows = [
        {"model": "org/m", "run_path": "bad1", "pass_label": "failed", "push_index": 1},
        {"model": "org/m", "run_path": "bad2", "pass_label": "failed", "push_index": 2},
    ]
    self.assertEqual(hfs._select_latest_run(rows, model="org/m"), "bad2")

def test_missing_pass_label_treated_as_usable(self):
    rows = [
        {"model": "org/m", "run_path": "legacy", "push_index": 1},
        {"model": "org/m", "run_path": "bad", "pass_label": "failed", "push_index": 2},
    ]
    self.assertEqual(hfs._select_latest_run(rows, model="org/m"), "legacy")
```

评论区精华

gemini-code-assist[bot] 提出性能优化建议：与其通过列表推导创建新列表，不如反向遍历已排序的 candidates，在找到第一个非失败 run 时立即返回，避免分配中间列表。该建议未在最终代码中采纳（代码仍使用了列表过滤方式）。JaredforReal 直接批准，没有进一步讨论。

- 通过反向遍历优化性能 (performance): 未采纳。最终代码使用了列表过滤方式，可能出于可读性考虑。

风险与影响

- 风险：风险极低。核心改动只有两行，且完全在测试框架的私有函数内，不影响任何运行时逻辑。测试覆盖了所有关键场景（包括向后兼容性）。唯一的边界情况是当所有候选都失败时，行为与原来一致（返回最新失败 run），不会恶化。
- 影响：直接影响夜间精度回归测试的正确性：修复后，持续回归将每晚被捕获，直到问题修复。仅影响测试框架，不影响任何用户可见功能。对系统性能无影响。
- 风险标记：测试框架变更，向后兼容保障

关联脉络

- PR #26902 nightly precision regression test: 本 PR 是对 PR #26902 引入的夜间精度回归测试的后续修复，解决了基线选择的关键 bug。
- PR #28925 fix(nightly-precision): pin flashinfer allreduce-fusion backend for TP-partial capture contract: 同为夜间精度测试相关的修复 PR，涉及同一测试框架的不同问题。