

# PR #3197 完整报告

radixark/miles

ci(lark): collapse jobs that failed identically into one card row

合并时间: 2026-09-11 03:14

原文链接: <http://prhub.com.cn/radixark/miles/pull/3197>

## 执行摘要

- 一句话: Lark CI 卡片按相同失败分析合并 job 行, 消除重复刷屏
- 推荐动作: 值得快速精读: 源码改动不到 30 行, 但“分组键 + 非空护栏 + 保留每 job 链接”是一套通用的告警去重取舍, 可直接迁移到其他通知渠道。阅读时建议重点确认三点: 分组键选择 (整段分析元组是否过严 / 过松)、空分析必须独立成行的护栏、以及截断 (`jobs[:limit]`) 在分组前后顺序上的语义。后续可考虑补一个“多个 job 逐条 UNAVAILABLE\_REASON”的用例, 或在分组键中显式过滤哨兵值, 以对齐 PR 描述中的承诺。

## 功能与动机

PR body 开门见山描述了触发场景: 2026-09-10 的 nightly 卡片把同一条失败打印了五次 (stage-b-cpu / stage-a-cpu 各 shard 都报 “[rollout] tests/e2e/agentice/test\_harbor\_e2b\_rollout.py Test collection failed because register\_cuda\_ci() rejected the file's hardware list”)。作者强调 “Not five failures — one.”: `collect_tests(sanity_check=True)` 会在每次 `run_suite.py` 启动时对仓库内每个注册文件做 AST 检查, 任何一个文件缺 `hardware=` 参数都会让所有 CPU shard 在 “Resolve suite plan” 阶段直接抛错, 测试一个都没跑。而卡片是 job 粒度, 于是一个缺陷按 shard 数复制。作者进一步指出这五行不是无关痛痒的噪音: “they were the whole card. Anything else that broke would have been pushed out of view.” —— 真正的第二处故障会被挤出可视区域, 所以要按“同一缺陷只占一行”来重排。

## 实现拆解

1. 引入分组函数 ([.github/workflows/scripts/lark\\_notify.py](https://github.com/workflows/scripts/lark_notify.py)): 新增 `group_identical_failures(jobs, reasons)`, 为每个 job 计算分组键 `analyses = tuple((reasons or {}).get(job.get("id")) or ())`, 即该 job 的完整分析元组; 键已存在则把 job 追加进组, 否则新建组。分组只改变展示聚合, 不改变 job 顺序与失败判定。
2. 设置“非空”护栏: 只有 `analyses` 为真值时才把组写入 `members_by_analyses` 索引, 因此无分析、分析为空列表、`reasons=None` (分析功能关闭) 三种情况都会退化为“一 job 一行”。这是防止“整张卡片被折叠成一行”的关键条件, 作者也把理由写进了函数的 docstring。
3. 重写行渲染 (`list_jobs_md`): 从“逐 job 追加一行 + 逐 job 渲染分析”改为“逐组追加一行”, 行内用 “, ” 连接组内所有 job 的 Markdown 链接, 随后只调用一次 `analysis_md(analysis, repo)` 渲染共享分析块。 `jobs[:limit]` 截断与 ... and N more 尾部逻辑

辑保持不变，保证长列表仍受 MAX\_LISTED\_JOBS 约束。

4. 测试配套 ([tests/ci/test/test\\_lark\\_notify.py](#))：新增 shard(index) 工厂构造与真实 nightly 同形的作业字典 (name 形如 stage-a-cpu (i) / run-cpu)，并用三个用例锁定行为：五个 shard 共享同一分析折叠为一行 (断言 "🔗" 恰好 2 次、stage-a-cpu 仍出现 5 次、无第二个 " - ")；两个相同 + 一个不同的分析产生两行且不同的那行内容完整；reasons=None 时三行保持不变。
5. 无配置 /schema/ workflow 改动：本次未触碰 .github/workflows/ci-lark-notify.yml、policies 或失败分析 prompt，变更被严格限制在渲染函数与其单测，配套验证为 pytest tests/ci/test/ (907 passed, 1 skipped) 与 pre-commit run --all-files。

关键文件：

- .github/workflows/scripts/lark\_notify.py (模块 CI 通知；类别 infra；类型 core-logic；符号 group\_identical\_failures, list\_jobs\_md, analysis\_md)：本 PR 唯一的源码改动：新增 group\_identical\_failures 重写 list\_jobs\_md 的行输出，决定同一缺陷的多个 shard 如何折叠为一行并保留每个 job 的独立链接。
- tests/ci/test/test\_lark\_notify.py (模块 CI 测试；类别 test；类型 test-coverage；符号 shard, test\_one\_defect\_across\_every\_shard\_collapses\_to\_a\_single\_row, test\_a\_different\_failure\_keeps\_its\_own\_row, test\_rows\_stay\_separate\_when\_no\_analysis\_is\_available)：新增 shard 工厂与三个用例，锁定折叠行为、不同故障保留独立行、以及 reasons=None 时不折叠这三条边界。

关键符号：group\_identical\_failures, list\_jobs\_md, shard,  
test\_one\_defect\_across\_every\_shard\_collapses\_to\_a\_single\_row,  
test\_a\_different\_failure\_keeps\_its\_own\_row,  
test\_rows\_stay\_separate\_when\_no\_analysis\_is\_available

## 关键源码片段

### [.github/workflows/scripts/lark\\_notify.py](#)

本 PR 唯一的源码改动：新增 group\_identical\_failures 重写 list\_jobs\_md 的行输出，决定同一缺陷的多个 shard 如何折叠为一行并保留每个 job 的独立链接。

```
# 新增的分组函数：把「分析结果非空且完全一致」的 job 归入同一行
# 返回 [(analyses, [job, ...]), ...], analyses 为空元组表示该行只含一个无可比对的 job
def group_identical_failures(jobs: list, reasons: dict | None) -> list[tuple[tuple, list]]:
    """一个缺陷可能让每个 shard 以完全相同的方式失败，逐 job 重复展示会把其他故障挤出卡片。
```

```
    仅在分析结果非空且完全一致时才合并：分析缺失时没有可比对的内容，
    这些行必须保持独立，否则「分析功能已关闭」的卡片会把所有失败 job 挤成一行。
    """
```

```
    groups: list[tuple[tuple, list]] = []
    members_by_analyses: dict[tuple, list] = {}
    for job in jobs:
        # 分组键 = 该 job 的全部分析构成的元组；无分析 (None 或空列表) 时退化为空元组
        analyses = tuple((reasons or {}).get(job.get("id")) or ())
        if analyses and analyses in members_by_analyses:
```

```

        # 命中已有分组：只把 job 追加为组员，不再新开一行
        members_by_analyses[analyses].append(job)
        continue
    members = [job]
    groups.append((analyses, members))
    if analyses:
        # 空分析不进入索引，保证「无分析」的 job 各自成行
        members_by_analyses[analyses] = members
    return groups

# 渲染侧改造：由「逐 job 一行」变为「逐分组一行」
def list_jobs_md(
    jobs: list,
    limit: int = MAX_LISTED_JOBS,
    reasons: dict | None = None,
    repo: str = DEFAULT_REPO,
) -> str:
    lines = []
    # 先按 limit 截断再分组：超出 limit 的组员不会出现在行内，由尾部 "and N more" 兜底
    for analyses, members in group_identical_failures(jobs[:limit], reasons):
        # 同一行内用 ", " 连接组内所有 job 链接，每个 job 仍是可单独点开的入口
        lines.append("- " + ", ".join(f"{{job['name']}}({job['html_url']})" for job in members))
        # 共享分析块只渲染一次（标签、test_name、原因、关联 PR 都由 analysis_md 统一拼装）
        for analysis in analyses:
            lines.extend(analysis_md(analysis, repo))
    if len(jobs) > limit:
        lines.append(f"- ... and {{len(jobs) - limit}} more")
    return "\n".join(lines)

```

## tests/ci/test/test\_lark\_notify.py

新增 shard 工厂与三个用例，锁定折叠行为、不同故障保留独立行、以及 reasons=None 时不折叠这三条边界。

```

def shard(index):
    # 构造与真实 nightly 同形的 shard 作业：名字带 stage 与序号，conclusion 固定为 failure
    return {
        "id": index,
        "name": f"stage-a-cpu ({{index}}) / run-cpu",
        "html_url": f"https://example/jobs/{{index}}",
        "conclusion": "failure",
    }

def test_one_defect_across_every_shard_collapses_to_a_single_row():
    # 五个 shard 共享同一条分析：期望只剩 1 个列表项，"🔍" 出现 2 次（标签行 + 原因行）
    same = ANALYSIS(reason="Collection failed because the hardware list was rejected.", tags=
("rollout",))
    content = HANDLER.list_jobs_md([shard(i) for i in range(5)], reasons={i: [same] for i in

```

```

range(5))
assert content.count("❸") == 2, content
assert content.count("stage-a-cpu") == 5 # 五个 job 链接都还在，只是合并到同一行
assert content.count("
- ") == 0 # 仅一条列表项，且位于字符串开头故不匹配 "
- "

def test_a_different_failure_keeps_its_own_row():
    # 两个相同 + 一个不同：不同的分析必须完整保留在自己的行里
    same = ANALYSIS(reason="Collection failed because the hardware list was rejected.")
    other = ANALYSIS(reason="The deterministic test exited with code 1.")
    content = HANDLER.list_jobs_md([shard(0), shard(1), shard(2)], reasons={0: [same], 1:
    [same], 2: [other]})
    assert content.count("
- ") == 1, content
    assert "The deterministic test exited with code 1." in content

def test_rows_stay_separate_when_no_analysis_is_available():
    # reasons=None（分析功能关闭）必须退化为「一 job 一行」，否则卡片会误导为单一故障
    content = HANDLER.list_jobs_md([shard(0), shard(1), shard(2)], reasons=None)
    assert content.count("
- ") == 2, content

```

## 评论区精华

本 PR 没有任何实质性的评讨论： [claude\[bot\]](#) 只留下仓库“需手动触发 review”的配置提示（[Comment @claude review ...](#)），并非技术评审；[guapisolo](#) 直接 APPROVED 且正文为空。因此设计权衡全部由作者在 PR body 与函数 docstring 中预先说明，最值得记录的是两条自我设限：一是“分组要求分析非空且完全相同”，作者明确写出“Without that condition a card with the analysis switched off — where no job has an analysis and there is nothing to compare — would collapse every failed job into a single row.”；二是“分析不可用的 job 保持独立”，理由是无可比对的内容。这两点没有 reviewer 追问或反驳，也没有留下未解决的公开疑虑。

- 为什么合并必须要求「分析非空且完全相同」（design）：以「非空 + 完全一致」作为分组键的准入条件，无分析的行行为与改动前完全一致；由 `test_rows_stay_separate_when_no_analysis_is_available` 用例锁定。
- 「分析不可用」的 job 是否真的保持独立（correctness）：未在 PR 内解决，也没有对应测试覆盖（仅覆盖整体 `unavailable=True` 与 `reasons=None`）；属于描述与实现的潜在偏差。
- 评审只做了形式化通过，缺实质技术讨论（other）：变更以作者自测（`pytest tests/ci/test/907 passed, 1 skipped`）与 pre-commit 通过作为唯一验证手段。

## 风险与影响

- 风险：

1. 合并语义依赖 LLM 生成文本的“完全相同”：分组键是整段分析元组，若两个根因不同的失败恰好被分析模型产出同样（或同样笼统）的文字，就会被折进同一行，读者可能误以为只有一处故障。缓解点在于每个 job 链接都被保留，可点开逐个确认，且真实分析通常带 test\_name 等具体信息，完全相同的概率较低。
2. “分析不可用”路径与描述存在潜在偏差：PR 描述称分析不可用的 job 会保持独立，但实现的分组键并未剔除哨兵值 UNAVAILABLE\_REASON。若多个 job 逐条写入同一个 unavailable 常量文本（test\_per\_job\_missing\_log\_reason\_and\_omitted\_footer\_render\_compactly 展示的正是这种单 job 形态），它们的元组会完全相同而被合并，与描述不符。本次测试只覆盖了整卡 unavailable=True 与 reasons=None，没有覆盖“多个 job 逐条 unavailable”的组合。
3. 截断与分组的交互：group\_identical\_failures(jobs[:limit], reasons) 先截断再分组，若一个故障横跨的 shard 数超过 MAX\_LISTED\_JOBS，行内只列出前 limit 个 job 链接，其余由 ... and N more 概括；信息不丢失，但行内链接数与计数提示的对应关系会变模糊。
4. 测试与渲染格式强耦合：用例通过统计 "❓" 出现次数和 "-" 计数来断言结果，任何对前缀符号或行首格式的调整都会连带让这些断言失败，属于可维护性成本而非功能风险。
5. 无回归外溢：改动只发生在 Lark 通知脚本的渲染函数，不参与失败采集、分析调用、CI 调度或训练 /rollout 主路径；即使渲染异常，也只影响通知卡片可读性。- 影响：对团队而言，收益直接体现在 nightly 值班体验：一个跨 shard 的收集期缺陷从占满整张卡片变成一行，其他独立故障重新回到视野；每个 job 仍保留独立链接，排障路径不变。对系统而言影响面极窄——只改变 .github/workflows/scripts/lark\_notify.py 生成的 Markdown 行结构，job 抓取、失败分析调用、policies 校验与 FastFail 行为均未改动，因此不存在对训练、rollout、权重同步等主链路的运行时影响。对协作者而言，新增的 group\_identical\_failures 成为一个可复用的“通知去重”抽象，后续若要按 root cause 或标签聚合失败，可直接在此基础上扩展分组键。- 风险标记：分组键未过滤 unavailable 哨兵，同文本不同根因可能被合并，unavailable 组合缺少测试覆盖，仅影响 CI 通知渲染层

## 关联脉络

- PR #3152 ci(lark): explain each failed job on the CI status card: 引入 Lark 卡片中每个失败 job 下方的 AI 分析块（analysis\_md 及 AnalysisOutcome 体系），本 PR 折叠的正是这套按 job 展开的分析行。
- PR #3168 ci(lark): report every failing test in a job: 同一脚本 .github/workflows/scripts/lark\_notify.py 的上一轮演进，把失败展示从 job 粒度细化到测试粒度；本 PR 从另一个方向（合并重复）继续优化同一张卡片的信噪比。
- PR #2701 feat(ci): declare supported GPU generations on every CUDA test: 建立 register\_cuda\_ci(hardware=...) 契约并要求每个 CUDA 测试声明硬件；本次 nightly 刷屏的直接诱因就是某个文件缺 hardware= 参数，使每个 shard 在收集阶段抛错。
- PR #3184 ci: declare hardware on the two CUDA tests that landed after #2701: 补上 #2701 之后遗漏的硬件声明，正是它修掉了让五个 CPU shard 同时失败的同一类缺陷；与本 PR 属于同一晚 nightly 问题的上下游。

- PR #2876 ci(e2e): Harbor trials on real e2b sandboxes through the full rollout path (manual for now): 新增了 tests/e2e/agentica/test\_harbor\_e2b\_rollout.py——PR body 中报错文件正是它，其 hardware= 声明缺失导致跨 shard 的重复失败。