

PR #3212 完整报告

radixark/miles

ci(lark): give each failure block room in the evidence packet

合并时间: 2026-09-12 07:39

原文链接: <http://prhub.com.cn/radixark/miles/pull/3212>

执行摘要

- 一句话: 修复 CI 证据包丢失败块, 按失败数分配预算
- 推荐动作: 值得精读。该 PR 展示了日志证据提取管线中的一个典型容量陷阱: 下载窗口足够大但逐级预算不匹配导致信息在加工环节丢失。作者用真实数据定位根因, 并采用“按失败实体分配预算 + 套件挑选优先”的对策, 设计思路可迁移到其他日志抽取、摘要生成等场景。同时 PR 中“已知缺口明确不处理”的取舍也值得学习: 避免在取证层无限扩容, 把真正的根因修复留给正确的归属层。

功能与动机

PR body 明确描述了问题: The evidence names this test but contains no failure output for it, so the decisive cause is missing. That sentence was literally true, and it was our fault, not the model's. 通过表格展示, 每个失败块大小为 9,273-13,410 字符, 而 job 日志预算只有 8,400 字符, 导致 `extract_log_evidence` 保留了零个完整块, 回退到只落在套件 roster 和 `##[error]` 上的 marker 窗口。作者认为让套件已挑选的失败块与通用启发式竞争单个槽位才是 bug。

实现拆解

实现拆解如下:

1. 新增失败块识别与整块保留逻辑: 在 `.github/workflows/scripts/ci_failure_analysis.py` 中新增 `FAILURE_BLOCK_RE` 正则匹配 `Last output of <test>: 行`, 并新增 `_failure_block_ranges()` 函数, 将每个失败块映射为 (起始行, 下一个块起始行) 区间。`extract_log_evidence()` 在原有 marker-window 逻辑之前先尝试提取所有失败块, 若合并后的 excerpt 长度不超过 `char_limit` 则直接返回, 彻底避免套件已挑好的块被后续启发式丢弃; 预算不足时才回退到原有 marker 窗口逻辑。同时把证据构造逻辑抽取到独立 `_log_evidence()` helper, 避免重复代码。
2. 按失败数分配日志预算: `_collect_evidence()` 先固定下载 `HARD_MAX_LOG_SCAN_BYTES` (256KB) 的日志窗口, 通过 `extract_failed_tests()` 从 suite summary 中读出失败测试数量 `failing_tests`, 再计算 `wanted = policy.max_log_chars_per_job * max(1, len(failing_tests))`。预算计算同时引入 `MIN_LOG_CHARS_PER_JOB` (4000 字符) 作为每个后续 job 的下限, 防止早期 job 因失败数过多而耗尽总预算 (公式: `log_limit = max(min(wanted, remaining - floor_for_others), min(wanted, job_budget))`)。

3. 剥离每行日志时间戳：新增 `LOG_TIMESTAMP_RE` 正则（匹配行首 `YYYY-MM-DDTHH:MM:SS.ffffffZ`），在 `redact_and_normalize()` 中统一替换为空。这省去约 9% 的摘录体积，且时间信息已由 `evidence` 的 `run` 块携带。
4. 调整策略与硬上限：`.github/workflows/policies/ci-failure-analysis.json` 中 `max_log_chars_per_job` 从 12000 提升到 14000（现在按“每个失败”计），`max_total_evidence_chars` 从 60000 提升到 160000；脚本内 `HARD_MAX_LOG_CHARS` 从 20000 提升到 60000，`HARD_MAX_TOTAL_EVIDENCE_CHARS` 从 80000 提升到 240000 以留出头寸。
5. 测试配套：`tests/ci/test/test_ci_failure_analysis.py` 新增 `failure_block()` 和 `two_block_log()` 构造带时间戳的失败块日志，并新增三个测试：时间戳被剥离、预算足够时两个失败块完整保留、预算不足时回退到 `marker` 窗口且输出有界。

整体改动使证据提取从“按 `job` 固定预算、启发式竞争”转变为“按失败实体分配预算、套件挑选优先”，在不破坏原有回退路径的前提下补齐了失败块缺失问题。

关键文件：

- `.github/workflows/scripts/ci_failure_analysis.py`（模块 失败分析；类别 `infra`；类型 `infrastructure`；符号 `_failure_block_ranges`, `_log_evidence`, `extract_log_evidence`, `redact_and_normalize`）：核心变更文件：新增失败块范围识别、整块优先保留逻辑、按失败数分配日志预算，并在日志归一化时剥离时间戳。
- `tests/ci/test/test_ci_failure_analysis.py`（模块 失败分析；类别 `test`；类型 `test-coverage`；符号 `failure_block`, `two_block_log`, `test_log_evidence_drops_the_timestamp_every_line_carries`, `test_every_failure_block_survives_whole_when_the_budget_allows`）：新增 `failure_block` 与 `two_block_log` 测试构造器，覆盖时间戳剥离、整块保留、预算不足回退三条关键行为，防止回归。
- `.github/workflows/policies/ci-failure-analysis.json`（模块 失败策略；类别 `infra`；类型 `configuration`）：策略配置同步调整：`max_log_chars_per_job` 改为按失败计，`max_total_evidence_chars` 从 60000 提升到 160000。

关键符号：`_failure_block_ranges`, `extract_log_evidence`, `_log_evidence`, `redact_and_normalize`, `_collect_evidence`

关键源码片段

`.github/workflows/scripts/ci_failure_analysis.py`

核心变更文件：新增失败块范围识别、整块优先保留逻辑、按失败数分配日志预算，并在日志归一化时剥离时间戳。

```
# 匹配 suite summary 中的 "Last output of <test>:" 块头。
FAILURE_BLOCK_RE = re.compile(r"(?m)^\s*Last output of (?P<path>\S+):\s*$")
```

```
def _failure_block_ranges(lines: list[str]) -> list[tuple[int, int]]:
    """定位每个失败测试的 "Last output of" 块的起止行区间。
```

套件已经按失败测试挑好了这些块，这里让它们整块存活，

避免通用 marker 窗口与它们竞争同一个 job 的预算。
块的结束行是下一个块的开始行，最后一个块延伸到日志末尾。

```
"""
starts = [index for index, line in enumerate(lines) if FAILURE_BLOCK_RE.match(line)]
return [
    (start, starts[position + 1] if position + 1 < len(starts) else len(lines))
    for position, start in enumerate(starts)
]

def extract_log_evidence(text: str, job_id: int, char_limit: int) -> dict[str, Any] | None:
    """从 job 日志中提取证据片段，优先保留完整失败块。"""
    sanitized = redact_and_normalize(text)
    if not sanitized.strip() or char_limit <= 0:
        return None
    lines = sanitized.splitlines()
    blocks = _failure_block_ranges(lines)
    if blocks:
        ranges = _merge_ranges(blocks)
        excerpt = _render_ranges(lines, ranges)
        if len(excerpt) <= char_limit:
            # 预算够就整块保留，直接返回
            return _log_evidence(job_id, lines, ranges, excerpt)
    # 预算不足时回退到原有 marker 窗口策略
    markers = [index for index, line in enumerate(lines) if FAILURE_MARKER_RE.search(line)]
    seeds = [(max(0, index - 8), min(len(lines), index + 13)) for index in markers[-4:]]
    ranges = _merge_ranges(seeds) if seeds else [(max(0, len(lines) - 40), len(lines))]
    excerpt = _render_ranges(lines, ranges)
    if not excerpt.strip():
        return None
    return _log_evidence(job_id, lines, ranges, excerpt)

def _log_evidence(job_id: int, lines: list[str], ranges: list[tuple[int, int]], excerpt: str) -> dict[str, Any]:
    # evidence id 用行号区间标识，text 是去时间戳后的摘要
    return {
        "id": f"job:{job_id}:log:{ranges[0][0] + 1}-{ranges[-1][1]}",
        "kind": "job_log",
        "text": excerpt,
        "sha256": hashlib.sha256(excerpt.encode()).hexdigest(),
    }
```

评论区精华

该 PR 没有真实的人工 review 评论（仅 claude[bot] 自动提醒配置了手动审查）。实质性的技术讨论都发生在 PR body 内，作者自述了根因分析和决策过程：

- 承认此前卡片上“证据没有失败输出”是工具缺陷而非模型问题：That sentence was literally true, and it was our fault, not the model's.
- 用表格量化了每个失败块与预算的差距，证明 8400 字符的 per-job 预算对 9-13K 的块完全不够。
- 明确一个已知缺口并主动排除：test_qwen3_4B_offload_disaggregated.py 的 traceback 从 84.7KB 前延伸到 6.8KB，FAILURE_TAIL_BYTES 只保留尾部 8KB，所以块内只有 teardown 断言而没有根因；作者认为该修复应属于 suite runner（保留首个 traceback 与 tail），而不是继续放大块预算。
- 暂无高价值评论线程

风险与影响

- 风险：风险点如下：
- 输入 token 成本上升：作者自测该次运行的 token 消耗从约 19K 增至 35-45K（约翻倍），max_total_evidence_chars 提升到 160000 后，若多个 job 同时携带多失败块的完整 traceback，可能显著增加每次 Lark 通知的分析调用成本。脚本设了 HARD_MAX_TOTAL_EVIDENCE_CHARS = 240000 作为硬顶，但超限时仍可能截断。
- 依赖日志格式：FAILURE_BLOCK_RE 依赖 Last output of ...: 的固定格式，LOG_TIMESTAMP_RE 依赖 GitHub 日志行首时间戳格式（YYYY-MM-DDTHH:MM:SS.ffffffZ）；若上游格式变化，两个正则都会静默失效，使证据回退到旧行为。
- 256KB 扫描窗口固定：_collect_evidence 现在固定下载 256KB，对于超大日志，早于该窗口的失败块或早期 traceback 不会被读取；对绝大多数 job 足够，但不绝对。
- 跨 job 预算复杂化：log_limit 公式中 remaining - floor_for_others 与 job_budget 的取法依赖 job 遍历顺序，若后续 job 数量变化或失败分布极端，仍可能出现某 job 证据不足；测试未覆盖多 job 预算分配的边界场景。
- 已知缺口遗留：Qwen3 4B offload 测试的根因 traceback 仍在证据块之外，作者明确不在此 PR 修复，因此该失败在未来卡片上仍可能呈现“缺失根因”。
- 影响：影响范围集中在 CI Lark 失败通知链路上：
- 对开发者：三个此前“证据无输出”的失败 job 现在能携带完整 traceback 和断言，显著降低排查 CI 失败的难度；受影响面覆盖所有使用 Lark 卡片通知的 CI 用户。
- 对系统：每次失败分析的输入 token 成本约翻倍，但仍在可接受的预算范围内；HARD_MAX_TOTAL_EVIDENCE_CHARS 提升到 240000 给策略留出了余量。
- 对团队：CI 工具链的可靠性提高，失败归因从“靠猜”变为“有据可查”；同时为后续 suite runner 修复早期 traceback 保留问题提供了更清晰的边界。整体影响为正向，成本略增但可控。
- 风险标记：CI 证据链路核心逻辑，输入 token 成本约翻倍，已知证据缺口未解决，依赖日志格式正则

关联脉络

- PR #3149 ci: repeat a failed test's last output in the suite summary: 本 PR 的失败块来源，且暴露了块完整保留与 per-job 预算冲突的问题。
- PR #3152 ci(lark): explain each failed job on the CI status card: 引入逐 job 失败分析的 Lark 卡片，本 PR 修复其证据提取丢块问题。
- PR #3168 ci(lark): report every failing test in a job: 让每个失败测试都有独立分析，本 PR 为其 "Last output of" 块提供完整证据。
- PR #3197 ci(lark): collapse jobs that failed identically into one card row: 同一 Lark 卡片链路的后续演进，共享证据包与预算逻辑。