PR #2815 完整报告

radixark/miles

dashboard: land Rollouts on the newest step that has data

合并时间：2026-09-01 04:32

原文链接：http://prhub.com.cn/radixark/miles/pull/2815

执行摘要

PR #2815 修复了 dashboard 在实时训练期间点击 Rollouts标签页会落到错误页的问题：step 的 dump 文件一出现即被列入 meta.rollout_ids，但该 step 可能仍在写入、被截断或无样本，旧逻辑把最新 id 直接烘焙进 href，导致 503/500/404 死路。改动引入 #/rollout/latest 保留路由，由 resolveLatest 从最新 step 往回有界探测（最多 5 步），落在第一个可读且有样本的 step 上，并用 entryHash 守卫异步竞态；显式 #/rollout/N 永不重定向，共享链接语义不变。纯前端改动（+68/-13），无 API、schema 或 reader 行为变化。

功能与动机

PR body 明确描述了痛点：

> A step is listed as soon as its rollout dump file appears, which is earlier than that step can be read, so during a live run clicking the tab regularly landed on an error page——and the only way out was editing the step number in the URL by hand.

作者系统枚举了 5 种破坏着陆的状态：dump 仍在写入（503 DumpStillWriting）、截断且 mtime 过期（torch.load 抛 RuntimeError，500）、记录但 0 样本（summary() 返回 0 行 0 列，groups() 里 group_by("group_index") 抛 ColumnNotFoundError，500）、文件已轮转（404）。其中“无样本”正是 live run 中最常见的形态。核心设计哲学是：不把修复调优到单一错误形状，而是把“抛错”与“加载成功但无样本”同等视为 not usable yet，统一走回退探测。

实现拆解

1. 路由层（miles/dashboard/static/app.js）：parseRoute 新增 latest 分支，把 #/rollout/latest 映射为 { view: "rollout", rolloutId: null }；crumbs 中 Rollouts 链接从 #/rollout/${latest} 改为常量 #/rollout/latest，并新增 replaceInPlace 处理器——当用户已在 rollout/tokens 视图内再点标签页时 preventDefault 加 location.replace 原地替换，避免路由仅监听 hashchange 导致历史栈压入两个相同 fragment、Back 键失效。step 面包屑在 rolloutId === null 时跳过渲染。
2. 视图层（miles/dashboard/static/views_rollout.js）：新增 resolveLatest(ids, evaluation)，从 ids.slice(-LANDING_LOOKBACK).reverse() 顺序探测 summary 接口，命中 rows.length > 0 即返回。LANDING_LOOKBACK = 5 是刻意设定的语义边界：超过 5 步就不再是“新 dump 尚未写完”的竞态，宁可让真实错误浮现，也不把用户静默带进旧数据。renderRollout 增加 null id 分支：先处理“无任何 step”的空态文案，再进入解析；解析完成后用 location.replace 把 hash 重写为真实 step id，使刷新、Prev/Next、面包屑都基

于真实 id；entryHash 守卫确保用户在解析期间导航离开后不会被拽回。

3. 请求层（miles/dashboard/static/api.js）：fetchOk 与 api 增加 retry503 选项。探测时传 retry503: false，因为此场景下 503 是“step 尚不可读”的确定性回答而非瞬态故障；若沿用默认重试，每探测一个坏 step 会白耗 4 个请求与 4.5 秒，最多 5 个候选会让页面长时间停在“finding the newest step with data…”。
4. 配套与验证：README.md 同步更新 Rollouts 行为描述。仓库无前端 JS 测试设施（无 package.json，tests/fast/dashboard/ 的 Python 测试不加载静态文件），作者用 Playwright 驱动真实 Chrome，对 dummy_dump.py 构造的正常、空样本、截断、双截断 dump 逐一验证着陆结果，并确认显式 #/rollout/2、刷新、Back、Metrics→Rollouts→sample→token 全链路不回环。

miles/dashboard/static/views_rollout.js

核心改动文件：新增 resolveLatest 有界回看解析与 renderRollout 的 null id 分支，是“着陆到首个可读 step”这一功能的实现主体。

```
// views_rollout.js：有界回看 + 着陆解析 + 竞态守卫
const LANDING_LOOKBACK = 5;

// 从最新 step 往回逐个探测：step 被列出（dump 文件出现）往往早于可读
//（写一半的 503、截断的 500、或“有记录但 0 样本”的 500），
// 所以取第一个能读到且至少有一行样本的 step 作为落点
async function resolveLatest(ids, evaluation) {
  for (const id of ids.slice(-LANDING_LOOKBACK).reverse()) {
    try {
      const summary = await api(`/api/rollout/${id}/summary`, { eval: evaluation }, { retry503:
      false });
      if (summary.rows.length) return id;
    } catch {
      // 写一半、被截断或已被轮转掉：继续往前探测前一个 step
    }
  }
  // 附近没有可读 step：仍回落到最新 step，让真实错误呈现给读者
  return ids.at(-1);
}

export async function renderRollout(view, meta, route) {
  const { evaluation } = route;
  let { rolloutId } = route;
  if (rolloutId === null) {
    // "#/rollout/latest" 分支：候选人列表限定为 train 或 eval
    const candidates = evaluation ? meta.rollout_ids.eval : meta.rollout_ids.train;
    if (!candidates.length) {
      const kind = evaluation ? "eval" : "rollout";
      view.replaceChildren(el("p", { class: "muted" }, [`No ${kind} steps have been dumped yet.`]));
      return;
    }
    view.replaceChildren(el("p", { class: "muted" }, ["finding the newest step with data…"]));
```

```
    // resolve 会发起多次请求；若用户在此期间已导航离开，此时重写 URL
    // 会把用户拖回本视图，所以在进入异步前先记录入口 hash
    const entryHash = location.hash;
    rolloutId = await resolveLatest(candidates, evaluation);
    if (location.hash !== entryHash) return;
    // 把 URL 重写为实际展示的 step：刷新、Prev/Next 与面包屑都基于真实 id；
    // 用 replace 而非赋值，避免把未解析的 latest 残留进历史栈
    location.replace(`#/rollout/${rolloutId}${evaluation ? "?eval=1" : ""}`);
    return;
  }
  // 显式 step id：保持原逻辑，用户输入什么就展示什么，绝不重定向
  const [summary, groups] = await Promise.all([
    api(`/api/rollout/${rolloutId}/summary`, { eval: evaluation }),
    api(`/api/rollout/${rolloutId}/groups`, { eval: evaluation }),
  ]);
  // … 后续渲染逻辑不变
}
```

miles/dashboard/static/app.js

路由与导航入口：parseRoute 识别 latest 保留路由，crumbs 把 Rollouts 链接改为常量并新增 replaceInPlace 原地替换，避免历史栈重复与 Back 失效。

```
// app.js：latest 保留路由与原地替换
function parseRoute() {
  const [path, query] = (location.hash.slice(1) || "/").split("?");
  const segments = path.split("/").filter(Boolean);
  const params = new URLSearchParams(query || "");
  // ...timeline / metrics 等其他分支省略 ...
  if (segments[0] === "rollout" && segments.length >= 2) {
    const evaluation = params.get("eval") === "1";
    // "latest" 是保留路由：在渲染期针对 dump 解析，而不是在链接里烘焙死 id；
    // 手输的 step 号始终精确兑现，不参与解析
    if (segments[1] === "latest") {
      return { view: "rollout", rolloutId: null, evaluation };
    }
    const rolloutId = Number(segments[1]);
    if (segments[2] === "sample" && segments.length === 4) {
      return { view: "tokens", rolloutId, sampleIndex: Number(segments[3]), evaluation };
    }
    return { view: "rollout", rolloutId, evaluation };
  }
  return { view: "metrics" };
}

function crumbs(route, meta) {
  const nav = (label, href, active, onclick = null) =>
    el("a", { class: `nav${active ? " active" : ""}`, href, onclick }, [label]);
  const parts = [nav("Metrics", "#/", route.view === "metrics")];
  // ...Compute Utilization 等导航省略 ...
```

```
  // Rollouts 链接恒为 "#/rollout/latest"；已在 rollout/tokens 视图内时改用
  // preventDefault + location.replace 原地替换，避免 hashchange 不触发时
  // 历史栈积压两个相同 fragment、Back 键失效
  if (meta.rollout_ids.train.length) {
    const onRollout = route.view === "rollout" || route.view === "tokens";
    const replaceInPlace = (event) => {
      event.preventDefault();
      location.replace("#/rollout/latest");
    };
    parts.push(nav("Rollouts", "#/rollout/latest", onRollout, onRollout ? replaceInPlace : null));
  }
  // 未解析出真实 id 前不渲染 step 面包屑
  if ((route.view === "rollout" || route.view === "tokens") && route.rolloutId !== null) {
    // ...step crumb 渲染省略 ...
  }
  // ...
}
```

miles/dashboard/static/api.js

请求封装层新增 retry503 开关，支撑 resolveLatest 探测时跳过 503 重试预算，避免每个坏 step 白等 4.5 秒。

```
// api.js：可关断的 503 重试
const RETRIES_503 = 3;

async function fetchOk(path, params, { retry503 = true } = {}) {
  const url = new URL(path, location.origin);
  for (const [k, v] of Object.entries(params)) {
    if (v !== undefined && v !== null) url.searchParams.set(k, v);
  }
  for (let attempt = 0; ; attempt++) {
    const res = await fetch(url);
    // 默认对 503 重试 3 次、每次间隔 1.5 秒（DumpStillWriting 场景）；
    // 但 landing 探测时 503 是“step 尚不可读”的确定性回答，传 retry503: false
    // 跳过重试，否则每探测一个坏 step 会白等 4.5 秒
    if (res.status === 503 && retry503 && attempt < RETRIES_503) {
      await new Promise((r) => setTimeout(r, 1500));
      continue;
    }
    if (!res.ok) {
      let detail = "";
      try {
        detail = (await res.json()).detail ?? "";
      } catch {
        /* non-json error body */
      }
      throw new Error(`HTTP ${res.status}: ${detail || url.pathname}`);
    }
    return res;
```

```
  }
}

export async function api(path, params = {}, options = {}) {
  return (await fetchOk(path, params, options)).json();
}
```

评论区精华

线程一：异步竞态（claude[bot] → 已修复）

claude[bot] 在 views_rollout.js 行内评论中指出：

> renderRollout awaits resolveLatest (up to 5 sequential API calls) then unconditionally calls location.replace with no check that the route is still current, so navigating away from '#/rollout/latest' while resolution is in flight gets you yanked back into the rollout view once it finishes, which cannot happen on base since the old code resolved the id synchronously with no async gap.

作者提交 e71d9df 修复：进入异步前记录 entryHash，解析完成后若 location.hash 已变化则跳过 location.replace。claude[bot] 复评确认："the follow-up commit addresses the mid-resolve navigation race I flagged... Checked the guard logic and the rest of the resolveLatest/renderRollout/parseRoute flow; no other issues found."

线程二：修饰键点击回归（claude[bot] → 未修复，作为可选 nit 合入）

claude[bot] 在 app.js 行内评论指出 replaceInPlace 无条件 preventDefault()，导致在 rollout 视图内 ctrl/cmd/ 中键点击 Rollouts 无法在新标签页打开（base 版本是普通锚点，始终支持修饰键）。claude[bot] 自标 severity: nit（minor UX regression），并明确"no need to push a fix before merging"，最终未修复即合入。

线程三：合并者直接上手的两个工程修复

Zhichenzzz 在 review 之外直接提交了两个修复：4902d01 解决历史栈重复条目问题（点击标签页会 push 两个相同 fragment，Back 键之间不触发 hashchange），6d9679e 为探测关闭 503 重试预算。这是"reviewer 发现真实问题、直接改代码"的协作模式，最终由 Zhichenzzz 批准合入。

风险与影响

- 测试覆盖缺口：仓库没有前端 JS 测试设施，本 PR 的验证完全依赖 Playwright 手动驱动；resolveLatest 的探测顺序、entryHash 守卫等逻辑后续被改动时，CI 不会自动发现回归。建议把 PR 中的验证矩阵沉淀为文档化的回归用例清单。
- 修饰键 UX 回归：已在 review 中确认并接受——位于 rollout/tokens 视图时，ctrl/cmd/ 中键点击 Rollouts 会吞掉"新标签页打开"意图。影响范围小但属于 base 行为退化。
- 竞态守卫的边界：entryHash 守卫只比较 hash 字符串。若未来路由包含会随渲染变化的查询参数或 hash 规范化逻辑，守卫可能失效；目前 dashboard 路由简单，风险可控。
- 保留字约定：latest 成为哈希路由保留段，未来新增 rollout 路由需避开；显式数字 step id 不受影响（id 恒为数字）。

- 行为契约：显式 #/rollout/N 永不重定向的设计保证了共享链接语义，但也意味着“URL 必须精确表达意图”的约定需要团队知晓。

关联脉络

- PR #2817（dashboard: give a step with no samples the summary schema）：在同一 dashboard 的 dump 读取层为“无样本的 step”补全声明式 schema，修复空 step 导致 500；本 PR 在前端把这类 step 视为“不可用”并跳过。两者从后端 schema 与前端着陆两个方向处理同一个“空 step”问题，属于同一健壮性主题的连续演进。
- PR #2794（dashboard: scroll the token strip instead of paging through it）：同属 miles/dashboard/static/ 前端迭代线，说明 dashboard 近期在密集打磨交互与导航。
- PR #2595（[RL] Represent and transport rollout sampling support）：定义了 rollout dump 的采样掩码与传输契约，dashboard 的 summary()/groups() 数据形态即源于该契约；未来 dump 格式变更需同步验证本 PR 的探测逻辑。