

# PR #57024 完整报告

vllm-project/vllm

[Bugfix] Avoid nested score cancellation handlers for /v1/score alias

合并时间: 2026-09-16 03:41

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/57024>

## 执行摘要

- 一句话: 移除 /v1/score 别名上的嵌套装饰器, 修复负载计数泄漏
- 推荐动作: 值得花 3-5 分钟精读, 但重点不在代码本身 (+0/-2), 而在它揭示的装饰器组合陷阱与 PR body 中的复现方法论。建议关注三点: (1) 记住 `with_cancellation/load_aware_call` 这类「每请求一次」的装饰器不可在委托链上重复应用, 别名必须只做委托; (2) 参考作者用 `SimpleNamespace + asyncio.Queue` 伪造 ASGI `receive` 的验证手法, 把该场景补成回归测试; (3) 顺带审一遍仓库里其他 deprecated 别名 (如 `rerank` 之外的转发路由), 确认没有残留的嵌套包装。

## 功能与动机

PR body 明确说明动机: /v1/score 是历史别名, `create_score_v1()` 先应用 `@with_cancellation` 与 `@load_aware_call`, 再委托给同样应用了两者的 `create_score()`, 于是「A single request therefore created two disconnect listeners and incremented server load twice; one listener could consume the disconnect while the real score handler continued running, leaving the load counter nonzero」。作者还提供了可复现脚本, 在 `main` 上输出 `before_disconnect receive_calls=2 load=2 / after_disconnect handler_cancelled=False load=1`, 在分支上输出 `receive_calls=1 load=1 / handler_cancelled=True load=0`, 说明这既是负载计数泄漏, 也是断连取消失效。PR body 同时指出其与 PR#55602 一脉相承: 后者已移除 /v1/rerank、/v2/rerank 别名的冗余 `cancellation` 包装器, 本 PR 把同一「窄委托」原则延伸到 /v1/score, 并额外移除其冗余的负载计数包装器。

## 实现拆解

1. 变更入口: `vllm/entrypoints/pooling/scoring/api_router.py` 中注册在 `POST /v1/score` 上的路由函数 `create_score_v1`。该函数本身不含业务逻辑, 只 `logger.warning(...)` 提示迁移到 `/score`, 然后 `return await create_score(request, raw_request)`。
2. 核心逻辑改造: 删除 `create_score_v1` 上方的 `@with_cancellation` 与 `@load_aware_call` 两行装饰器 (-2 行、无新增行), 保留路由装饰器、依赖 `validate_json_request` 与响应 `schema`。原因是这两个装饰器管理的是「每请求一次」的生命周期事件, 不具备幂等性; 别名与规范入口同时应用时会重复注册。
3. 收敛后的请求链路: 请求进入 `create_score_v1` (仅打日志) → 委托 `create_score` → `@load_aware_call` 对 `app.state.server_load_metrics` 自增并在 `finally` 中自减 →

@with\_cancellation 注册唯一的 http.disconnect 监听 → score(raw\_request) 从 app.state.serving\_scores 取出实际打分 handler 执行。断连时由该唯一监听器取消真实 handler，取消异常再经由 await 链向上传播回别名协程，因此别名自身的取消语义并未丢失。

4. 配套改动：无测试、配置、文档或部署改动。验证依赖作者在 PR body 中给出的最小复现脚本（用 SimpleNamespace 伪造 app.state 与 ASGI receive，用 asyncio.Queue 注入 http.disconnect），以及既有 scoring 测试套件（test\_io\_processor\_unit.py 等 17 passed、test\_bi\_encoder\_online.py 16 passed）与 pre-commit。
5. 影响边界：仅 /v1/score 别名的装饰器组合发生变化；/score、/rerank、/v1/rerank 的实现在 with\_cancellation/load\_aware\_call 共享 helper 均未改动，路由路径与请求 / 响应契约保持兼容。

关键文件：

- vllm/entrypoints/pooling/scoring/api\_router.py（模块 评分接口；类别 source；类型 endpoint；符号 create\_score\_v1, create\_score）：唯一的变更文件，也是本次修复的全部内容所在：删除 /v1/score 别名 create\_score\_v1 上的 @with\_cancellation 与 @load\_aware\_call，让取消监听与负载计数收敛到规范入口 create\_score。

关键符号：create\_score\_v1, create\_score

## 关键源码片段

### vllm/entrypoints/pooling/scoring/api\_router.py

唯一的变更文件，也是本次修复的全部内容所在：删除 /v1/score 别名 create\_score\_v1 上的 @with\_cancellation 与 @load\_aware\_call，让取消监听与负载计数收敛到规范入口 create\_score。

```
# vllm/entrypoints/pooling/scoring/api_router.py —— 规范入口与 deprecated 别名（变更后）
# 顶部已导入 with_cancellation / load_aware_call / logger
等，这里只保留与本次变更相关的实现单元。
```

```
# === 规范入口 /score：整条请求链路上唯一持有生命周期装饰器的地方 ===
```

```
@router.post(
    "/score",
    dependencies=[Depends(validate_json_request)],
    responses={
        HTTPStatus.BAD_REQUEST.value: {"model": ErrorResponse},
        HTTPStatus.INTERNAL_SERVER_ERROR.value: {"model": ErrorResponse},
    },
)
```

```
@with_cancellation # 注册唯一的 http.disconnect 监听协程：客户端断开时取消真正在跑的打分逻辑
```

```
@load_aware_call # 每请求给 app.state.server_load_metrics 加 1，退出（含异常 / 取消）时减 1
```

```
async def create_score(request: ScoreRequest, raw_request: Request):
```

```
    handler = score(raw_request) # 从 app.state.serving_scores 取出实际打分实现
    return await handler(request, raw_request)
```

```
# === 历史别名 /v1/score: 本 PR 删掉了这里的两行装饰器 (+0/-2) ===
@router.post(
    "/v1/score",
    dependencies=[Depends(validate_json_request)],
    responses={
        HTTPStatus.BAD_REQUEST.value: {"model": ErrorResponse},
        HTTPStatus.INTERNAL_SERVER_ERROR.value: {"model": ErrorResponse},
    },
)

async def create_score_v1(request: ScoreRequest, raw_request: Request):
    logger.warning(
        "To indicate that Score API is not part of standard OpenAI API, we "
        "have moved it to `/score`. Please update your client accordingly."
    )

    # 直接委托给 create_score: 取消监听与负载计数全部由被委托方负责。
    # 早前这里同样挂着 @with_cancellation / @load_aware_call, 于是单次请求会:
    # 1) 注册 2 个 disconnect 监听协程, 两个协程都 await raw_request.receive(),
    # 即两次消费同一条 ASGI receive 队列, 其中一个监听器抢走 disconnect 消息;
    # 2) 对 server_load_metrics 自增 2 次, 而断开后只有一个 finally 执行,
    # 计数停在 1 形成泄漏 (reproducer: after_disconnect load=1);
    # 3) 真正的打分 handler 拿不到取消信号继续空跑 (reproducer: handler_cancelled=False)。
    # 删除两个装饰器后, 同一脚本变为 receive_calls=1 / load=1, 断开后 handler_cancelled=
    True、load=0。
    return await create_score(request, raw_request)
```

## 评论区精华

本 PR 的 review 交互很少: [claude\[bot\]](#) 以「This pull request is from a fork — automated review is disabled」告知自动审查未运行, [yewentao256](#) 直接 APPROVED 并留言「LGTM, thanks for the work!」, 没有行内 review 评论, 说明分歧点主要不在实现而是在是否需要更多验证。真正的技术交锋发生在 PR body: 作者用可执行复现脚本论证了「两个监听器共享同一条 ASGI receive 队列, 其中一个抢先消费 disconnect, 导致真实 handler 继续运行、负载计数无法归零」, 并用 main / 分支两段输出对照 (load=2→1 与 load=1→0、handler\_cancelled=False→True) 证明修复有效。值得记录的未决点是: 本次没有把复现脚本沉淀为 tests/entrypoints/pooling/scoring/ 下的回归用例, 也没有一并排查其他 deprecated 别名是否存在相同的嵌套装饰器模式。

- 别名与规范入口重复应用生命周期装饰器, 导致 disconnect 监听与负载计数翻倍 (correctness): 确认是装饰器非幂等造成的请求生命周期重复注册, 删除别名层两个装饰器后由规范入口统一负责, maintainer [yewentao256](#) 直接 APPROVED。
- 是否需要把复现脚本沉淀为自动化回归测试 (testing): 未在 PR 内补测试, 回归保护依赖手工脚本; 若同一模式再次被引入, 问题会无声复现。
- 与 PR#55602 的一致性与适用范围: 别名只做窄委托 (design): 确立「转发别名只负责日志与委托, 生命周期装饰器由规范入口独占」的约定, 且刻意控制改动范围避免波及共享实现。

## 风险与影响

- 风险:

1. 缺少自动化回归保护（中低风险）：本次只删 2 行、未新增测试，复现脚本停留在 PR body 中。若后续有人再次给别名加回生命周期装饰器，server\_load\_metrics 泄漏会无声复现；建议把「断开后 load 归零 + handler 被取消」写成单测。
2. 取消语义变化（低风险但需知晓）：main 上断连后 handler\_cancelled=False，真实打分 handler 仍继续消耗算力；分支上变为 True，在途打分会被真正中止。对曾经「客户端断开但仍计算完」的行为属于行为变更，不过这正是修复目标。
3. 监控口径变化（低风险）：app.state.server\_load\_metrics 在 /v1/score 路径上不再虚高一倍，线上按旧口径标定的负载告警或弹性阈值在切换到本版本后读数会下降，需要确认不会误判为流量下降。
4. 同类模式排查缺失（低风险）：PR body 提到 #55602 已处理 rerank 别名，但仓库中其他转发型别名是否仍有 with\_cancellation/load\_aware\_call 嵌套尚未系统确认。
5. 兼容性风险很低：路由路径、请求 schema、响应 schema、deprecated 警告日志全部保持原样，客户端与 SDK 无感知；被删除的仅是别名层的冗余包装。- 影响：对用户：使用 deprecated /v1/score 的客户端行为基本不变（同样返回打分结果、同样打印迁移警告），差异只在断开连接时请求会被真正取消，不会继续空跑占算力。对系统：请求生命周期管理从「两个监听协程争抢 receive 队列 + 两次计数」收敛为单实例，修复了 server\_load\_metrics 不归零造成的长期负载统计偏差，对依赖该指标做容量判断与准入决策的部署更准确。对团队：确立了「转发型别名只做委托、生命周期装饰器只由规范入口持有」的约定，改动面仅 1 文件 2 行，review 成本极低，适合作为同类清理的模板。
  - 风险标记：缺少测试覆盖，指标口径变化，取消语义变更，同类模式未全面排查

## 关联脉络

- PR #55602 [Bugfix] Remove redundant cancellation wrappers from /v1/rerank and /v2/rerank aliases（标题按 PR body 描述转述）：PR body 明确引用：该 PR 移除了 /v1/rerank、/v2/rerank 别名上的冗余 cancellation 包装器。本 PR 沿用同一窄委托原则处理 /v1/score，并额外清理了别名上的 @load\_aware\_call，属于同一条清理线的延续。
- PR #56505 [Pooling] Cap max-length padding for chunked embeddings: 同属 vllm/entrypoints/pooling/ 子树的前端接口修正（改 vllm/entrypoints/pooling/base/protocol.py），反映 pooling 系列入口近期在持续做行为纠偏，与本 PR 的接口层 bugfix 属于同一模块脉络。
- PR #54746 [Frontend] Share max\_num\_queued\_reqs across API server processes: 同样处理前端侧的请求准入与负载计量（vllm/v1/engine/admission\_control.py），与本 PR 关注的 server\_load\_metrics 计量准确性主题相邻：一个是跨进程共享准入计数，一个是消除别名路径的计数虚高。
- PR #56760 [Bugfix] Measure complete pooling responses: 涉及 pooling 接口侧的度量正确性（vllm/benchmarks/lib/endpoint\_request\_func.py），与本 PR「修正前端计量 / 生命周期统计」的问题类别一致，可作为同类度量类缺陷的参照。