

PR #55602 完整报告

vllm-project/vllm

[Bugfix] Avoid nested rerank cancellation handlers

合并时间: 2026-09-15 11:19

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/55602>

执行摘要

- 一句话: rerank 别名删除冗余取消装饰器, 修复断连后仍处理
- 推荐动作: 值得精读, 但重点不在 2 行代码, 而在讨论区展示的设计取舍:
- 当症状出现在公共装饰器 (`with_cancellation`) 而根因在调用方重复包装时, 应优先消除重复装饰而不是修改公共取消语义; 本 PR 的 reviewer 三轮质疑 (全局语义变更、清理循环使任务免疫取消、`return_exceptions=True` 吞异常) 几乎覆盖了这类改法的全部陷阱, 可作为反面清单复用。
- 建议后续补一个成本可控的回归测试 (可沿用作者删除前用同一 `raw_request` 调用两个真实别名的写法, 断言只发生一次 `receive()` 与 `cancel` 传播), 把「别名不得再叠加 `@with_cancellation`」固化成可执行约束。
- 建议顺带审计 `vllm/entrypoints/pooling/scoring/api_router.py` 中其余别名 / 转发路由 (如通往 `create_score` 的 `Score` 别名) 以及所有返回 `await <已装饰 handler>(request, raw_request)` 的位置, 确认不存在第二处嵌套取消监听。

功能与动机

Issue #55502 报告: 即使路由上有 `@with_cancellation`, `/rerank` 在客户端断开后仍会处理完所有排队的文档。复现数据显示 20 个 rerank 请求 (620 对 query/document) 在 `curl --max-time 2` 断连后仍持续计算了 57 秒, `prompt_tokens_total` 增加 469,080; 报告者形容在网关侧「a two-minute burst into a fifteen-minute outage before we added a replica」。PR 作者进一步定位到根因: `/v1/rerank` 与 `/v2/rerank` 别名把同一请求交给已装饰的 `/rerank`, 形成两个并发的 `raw_request.receive()` 消费者, 「could leave the real rerank handler running after disconnect」, 因此只要保留规范路由上的唯一取消监听即可。

实现拆解

1. 定位入口与调用链: `vllm/entrypoints/pooling/scoring/api_router.py` 定义 `/rerank` (`do_rerank`)、`/v1/rerank` (`do_rerank_v1`)、`/v2/rerank` (`do_rerank_v2`) 三个路由。别名函数只打一条 `logger.warning_once` 告警后 `return await do_rerank(request, raw_request)`, 而 `do_rerank` 内部通过 `handler = rerank(raw_request)` 取出 pooling 侧的 rerank handler, 每个 document 会提交一次 `encode()`, 这正是长耗时、可取消的部分。
2. 根因分析 (嵌套装饰器): `@with_cancellation` 定义在 `vllm/entrypoints/serve/utils/api_utils.py`, 其实现会为每个请求 `asyncio.create_task(listen_for_disconnect(request))`。当别

名和规范路由同时被装饰时，同一个 `raw_request` 被两个监听任务消费 `receive()`：外层别名先收到 `http.disconnect` 并取消自己，内层 `/rerank` 任务却继续执行，于是 `issue` 中的现象出现——取消只终止了外壳，引擎里排队的工作照常跑完。

- 3. 核心改动：删除 `do_rerank_v1` 和 `do_rerank_v2` 上的 `@with_cancellation`，保留 `/rerank` 上的 `@with_cancellation + @load_aware_call`。这样每个请求恰好一个断连监听者，取消信号可以直接传播到承载 `encode()` 的内层任务，同时 `load_aware_call` 的计数不会因为多层包装而重复增减。别名对外行为（路径、`validate_json_request` 依赖、`ErrorResponse` 响应模型、`deprecation` 告警）完全不变。
- 4. 演进与回退（重要）：初版方案并非此文件，而是修改 `vllm/entrypoints/serve/utils/api_utils.py` 中 `with_cancellation` 的收尾逻辑（`finally` 中 `cancel` 两个子任务、`asyncio.gather(..., return_exceptions=True)` 并循环等待清理完成），并新增 `tests/entrypoints/serve/utils/test_api_utils.py`。reviewer 指出这是全局语义变更、会让路由任务对后续取消免疫、且会吞掉清理异常；作者接受建议回退到别名级最小修复。
- 5. 测试配套（最终为空）：作者一度新增 `tests/entrypoints/pooling/scoring/test_api_router.py`（自测 2 passed），但 `nooooop` 与 `taneem-ibrahim` 都认为不必新增该测试，作者在 `commit c92bd925` 中删除，因此合并版本仅有 1 个源文件、-2 行，没有任何测试改动。

维度	结论
变更面	1 个文件、删除 2 行、无新增代码
受影响路由	<code>/v1/rerank</code> 、 <code>/v2/rerank</code> (<code>/rerank</code> 行为不变)
配置 / schema / 部署	无
测试覆盖	最终未新增测试，回归由 <code>issue</code> 复现脚本手工验证

关键文件：

- `vllm/entrypoints/pooling/scoring/api_router.py`（模块 重排接口；类别 `source`；类型 `entrypoint`；符号 `do_rerank`, `do_rerank_v1`, `do_rerank_v2`, `with_cancellation`）：唯一改动文件：删除 `/v1/rerank` 与 `/v2/rerank` 别名上的 `@with_cancellation`，使每个 `rerank` 请求只有 `/rerank` 内的一个断连监听者，从根本上消除两个并发消费 `raw_request.receive()` 的消费者。

关键符号：`do_rerank`, `do_rerank_v1`, `do_rerank_v2`, `with_cancellation`, `listen_for_disconnect`, `load_aware_call`

关键源码片段

`vllm/entrypoints/pooling/scoring/api_router.py`

唯一改动文件：删除 `/v1/rerank` 与 `/v2/rerank` 别名上的 `@with_cancellation`，使每个 `rerank` 请求只有 `/rerank` 内的一个断连监听者，从根本上消除两个并发消费 `raw_request.receive()` 的消费者。

```
# vllm/entrypoints/pooling/scoring/api_router.py（合并后的 rerank 路由实现）

# 规范路由 /rerank：保留 @with_cancellation + @load_aware_call。
```

```

# 它现在是整条 rerank 调用链上唯一的断连监听者,
# 也是唯一消费 raw_request.receive() 的地方。
@router.post(
    "/rerank",
    dependencies=[Depends(validate_json_request)],
    responses={
        HTTPStatus.BAD_REQUEST.value: {"model": ErrorResponse},
        HTTPStatus.INTERNAL_SERVER_ERROR.value: {"model": ErrorResponse},
    },
)
@with_cancellation # 内部为每个请求 create_task(listen_for_disconnect(request))
, 只允许存在一层
@load_aware_call # 负责 server load 计数, 必须与取消路径配对, 避免重复加减
async def do_rerank(request: RerankRequest, raw_request: Request):
    handler = rerank(raw_request) # 取出 pooling 侧 rerank handler
    return await handler(request, raw_request) # 每个 document 一次 encode()
    , 耗时长且需要可取消

# 别名路由 /v1/rerank: 把同一个 raw_request 原样转发给已被装饰的 do_rerank。
# 本次修复就是删掉这里的 @with_cancellation ——
# 否则同一 request 会出现两个 listen_for_disconnect 消费者:
# 外层别名先看到 http.disconnect 并取消自己, 内层 /rerank 任务却被留下继续向引擎排队,
# 这正是 issue #55502 中“客户端离开后仍算完 620 对文档”的成因。
@router.post(
    "/v1/rerank",
    dependencies=[Depends(validate_json_request)],
    responses={
        HTTPStatus.BAD_REQUEST.value: {"model": ErrorResponse},
        HTTPStatus.INTERNAL_SERVER_ERROR.value: {"model": ErrorResponse},
    },
)
# 注意: 这里刻意不再叠加 @with_cancellation, 取消监听交由内层 do_rerank 独享
async def do_rerank_v1(request: RerankRequest, raw_request: Request):
    logger.warning_once(
        "To indicate that the rerank API is not part of the standard OpenAI"
        " API, we have located it at `/rerank`. Please update your client "
        "accordingly. (Note: Conforms to JinaAI rerank API)"
    )
    return await do_rerank(request, raw_request) # 只做告警与转发, 不再创建第二个取消消费者

# 别名路由 /v2/rerank: 与 /v1/rerank 同构, 同样只保留转发职责。
@router.post(
    "/v2/rerank",
    dependencies=[Depends(validate_json_request)],
    responses={
        HTTPStatus.BAD_REQUEST.value: {"model": ErrorResponse},
        HTTPStatus.INTERNAL_SERVER_ERROR.value: {"model": ErrorResponse},
    },
)

```

```
    },  
)  
async def do_rerank_v2(request: RerankRequest, raw_request: Request):  
    return await do_rerank(request, raw_request) # 断连时由内层 @with_cancellation 传播取消
```

评论区精华

本次 PR 的讨论价值远大于代码本身：初版改的是公共取消装饰器，被 reviewer 完整地推回到根因修复。

1. [taneem-ibrahim](#) (CHANGES_REQUESTED) 针对 `vllm/entrypoints/serve/utils/api_utils.py` 指出: "This changes cancellation semantics for every decorated endpoint, while the asserted root cause is redundant cancellation wrappers on the rerank aliases... Could we instead remove the redundant outer decorators from the aliases so each request has exactly one disconnect listener?" 作者回复 "I removed the redundant decorators from /v1/rerank and /v2/rerank, so the shared request now has one disconnect listener." 这是本次 PR 的定调性评论。
2. 同一位 reviewer 质疑初版 [finally](#) 清理循环的安全性: "This makes the route task resistant to every subsequent cancellation until both children terminate. A handler that delays or suppresses CancellationError can therefore block timeouts or server shutdown indefinitely. Please bound the cleanup wait or avoid the global behavior change by removing the redundant alias wrappers."
3. 针对 `asyncio.gather(*tasks, return_exceptions=True)`: "`return_exceptions=True` retrieves and then silently discards non-cancellation failures raised during handler cleanup. That can hide an engine-abort failure and leave work running without diagnostics." 作者确认 "That cleanup path is gone with the global helper change, so this patch no longer swallows cleanup exceptions."
4. 初版测试被指未复现真实拓扑: "This test instead gives the wrappers independent request objects and independent disconnect streams, so it misses the competing receive() consumers and possible duplicate load-counter updates. Please reproduce the actual alias topology using one request." 作者随后改为用同一个 request 调用两个真实别名，但该测试最终被整体删除。
5. 测试是否保留存在明确分歧并已定论: [nooooop](#) 说 "I don't think we really need this test.", [taneem-ibrahim](#) 附和 "I agree with @nooooop. We don't need to add another additional test for this change. Can you please remove this?", 作者在 commit [test: drop extra scoring api_router test per review](#) 中删除 `tests/entrypoints/pooling/scoring/test_api_router.py`。
6. 流程层面: [claude\[bot\]](#) 因来自 fork 而跳过自动评审; [github-actions\[bot\]](#) 先因分支落后上游 416 个 commit 拒绝启动 CI，作者 rebase 后由 [nooooop](#) 再次 `/ci run` 触发 Buildkite CI#88980 才通过。作者也主动 bump 过 "Happy to drop the regression test if preferred."
- 应改公共 `with_cancellation` 还是只去掉别名上的冗余装饰器 (design): 作者接受建议，回退公共 helper 改动，把修复收敛为删除 `/v1/rerank` 与 `/v2/rerank` 的 `@with_cancellation` (最终 diff 仅 -2 行)。

- finally 清理循环会让路由任务对后续取消免疫 (correctness): 随全局 helper 改动一起回退, 最终版本不引入任何等待清理的循环。
- return_exceptions=True 会静默吞掉清理阶段的异常 (correctness): 方案回退后该路径不再存在, 风险自然消除; 但也说明最终版本未对 engine abort 失败做任何可观测性补充。
- 回归测试未复现真实别名拓扑 (testing): 测试被改写为贴合真实拓扑, 但在后续 review 中该文件被整体删除, 最终合并版本没有任何自动化测试。
- 是否保留 tests/entrypoints/pooling/scoring/test_api_router.py (testing): 测试文件被删除 (commit c92bd925), 合并版本只保留源码改动; 留下的疑虑是别名层取消语义缺少回归保护。
- CI 因分支落后上游 416 个 commit 被拒, rebase 后才启动 (other): 流程性阻塞已通过 rebase + 重新触发 CI 解决, 与代码正确性无关。

风险与影响

• 风险:

1. 缺少自动化回归保护: 最终版本删除了 tests/entrypoints/pooling/scoring/test_api_router.py, 没有任何用例锁定「别名不得再叠加取消装饰器」这一约束。若后续新增 /v3/rerank 或在 router 级别统一加 @with_cancellation, issue #55502 会静默复现; 这也是作者唯一自测过 (2 passed) 却未保留的证据。
1. 取消语义靠隐式约定维持: vllm/entrypoints/serve/utils/api_utils.py 中的 with_cancellation 本身没有防护——它依旧会为每次装饰调用创建独立的 listen_for_disconnect 任务, 同一 raw_request 被多层包装时仍会出现多个 receive() 消费者。本次只是把重复装饰拆掉, 未从机制上阻止同类嵌套 (同文件里还有向 create_score 转发的 Score 别名路由, 本次材料未展示其完整装饰器组合, 建议一并审计)。
2. 端到端取消是否真正到达引擎未验证: 作者明确说明 "No model evaluation was run", 验证仅限于针对性 pytest 与 pre-commit。issue 报告者也承认 "I have not traced where the cancellation is lost"——移除外层装饰器后, 取消必须经 do_rerank → await handler(request, raw_request) 一路传播到 ServingScores 为每个 document 创建的 encode() 请求, 这条链路的 engine abort 行为没有在本 PR 中用指标 (num_requests_waiting、prompt_tokens_total) 复测。
3. 行为差异面: 正常完成的请求输出不变; 变化只体现在断连时机——此前别名返回后内层可能仍在跑, 现在取消直接作用于内层任务。对依赖旧有「取消后仍能读完」行为的客户端属于行为收紧, 但符合 issue 的预期。- 影响: 用户侧: RAG / gateway 场景是最大受益者。issue 中描述的 60 秒客户端超时在突发流量下会持续堆积无人读取的 rerank 文档、耗尽 GPU 时间并造成十余分钟不可用; 修复后断连即可取消对应 rerank 调用的排队工作, 配合 --max-num-seqs 等限流参数可更快恢复服务能力。

系统侧: 影响半径被刻意限制在 vllm/entrypoints/pooling/scoring/api_router.py 的 /v1/rerank 与 /v2/rerank 两个别名; /rerank 主路径、其它 pooling 路由以及 vllm/entrypoints/serve/utils/api_utils.py 的公共取消语义均未被改动, 因此回归面很小。这也意味着收益是局部的——其它存在同类嵌套装饰模式的路由不会因此受益。

团队侧：这是一次典型的「reviewer 把过宽的全局修复收敛为最小根因修复」案例，
`with_cancellation` + 别名透传的交互模式值得作为 entrypoint 层代码审查的检查项固化下来。

- 风险标记：缺少回归测试，取消语义依赖隐式约定，未跑端到端模型验证，同类嵌套装饰器未全面审计

关联脉络

- PR #56844 [Bugfix] Validate routed-expert prompt offsets before engine submission: 同为 entrypoint / 请求生命周期边界的防御性修复：在 vllm/v1/engine/input_processor.py 与 scheduler 之间补校验，避免非法请求把影响扩散到引擎内部，与本 PR「把取消 / 校验问题挡在路由层而不是改引擎公共逻辑」的思路一致。
- PR #56486 [Bugfix][KV Offload] Ignore pending chunks in invalid sliding windows: 同属「客户端 / 窗口失效后仍继续处理无意义工作」一类语义缺陷，修复点同样是让已失效的上下文不再阻塞或驱动后续执行，可作为同类问题的对照案例。
- PR #56899 [Minor] Invert MRV1 specdec method fallback logic: 同样是极小 diff 的行为语义调整（vllm/config/vllm.py 中回退判断反转），体现了本仓库偏好用「收敛到单一判定点」的小改动来消除隐含的多重语义，与本 PR 去掉重复装饰器的取舍方式相近。