PR #33940 完整报告

sgl-project/sglang

fix: route scheduler aborts to multi-tokenizer workers

合并时间：2026-08-12 02:15

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33940

执行摘要

- 一句话：修复多 tokenizer 下调度器 AbortReq 路由丢失致请求挂起
- 推荐动作：值得精读。改动虽仅 11 行，但卡点在于类型边界的技术权衡：用 getattr 替代 isinstance 放宽输入类型、把类型判断移到 output 上以保证不覆盖已有路由。测试设计（用真实调度器 Req 构造 AbortReq，再经真实 MultiTokenizerRouter._distribute_result_to_workers 验证最终目的地）比纯 mock 更有说服力，可作为多 worker 路由类修复的参考模板。

功能与动机

Issue #33858 报告：以 --tokenizer-worker-num 2 部署 deepseek-v4-flash-0731 时，队列溢出后 benchmark 卡在 1309/4096，num_running_reqs 与 num_queue_reqs 均为 0，未完成请求的 TCP 连接停留在 ESTABLISHED 状态且永不收到响应；仅将 tokenizer worker 数改为 1 后同一 benchmark 在 118 秒内正常完成。PR body 进一步定位根因：调度器内部 Req 携带来源 worker 的 http_worker_ipc，但 SenderWrapper 只从 IPC BaseReq 复制路由，AbortReq 因此没有目的地、被 MultiTokenizerRouter 静默丢弃。

实现拆解

1. 根因定位：python/sglang/srt/managers/scheduler_components/output_sender.py 中 SenderWrapper.send_output 的路由补全条件为 isinstance(recv_obj, BaseReq)，而调度器内部的 Req（schedule_batch.py）并非 BaseReq，因此队列溢出产生的 AbortReq 拿不到回程路由。
2. 核心修复：将 recv_obj 参数类型放宽为 Optional[object]，改用 getattr(recv_obj, "http_worker_ipc", None) 安全读取；同时把 isinstance 判断从 recv_obj 移到 output（要求输出是 BaseReq 且未指定目的地）。这样既能让调度器 Req 的回程路由注入 AbortReq，又保证已显式路由的输出不被覆盖，BaseBatchReq 批量输出仍不参与单条路由补全。
3. 回归测试：新增 test/registered/unit/managers/scheduler_components/test_output_sender.py，注册进 CPU CI（register_cpu_ci，suite base-a-test-cpu）。两个用例分别验证「已路由输出不被覆盖」与「无目的地 AbortReq 经 MultiTokenizerRouter._distribute_result_to_workers 到达来源 worker」，链路覆盖真实 Req -> AbortReq -> MultiTokenizerRouter。

关键文件：

- python/sglang/srt/managers/scheduler_components/output_sender.py（模块 输出路由；类别 source；类型 core-logic；符号 SenderWrapper, send_output）：修复核心文件。send_output 是调度器输出统一出口，本 PR 在此放宽 recv_obj 类型、用 getattr 读取 http_worker_ipc 并把 isinstance 判断移到 output 上，使调度器内部 Req 的回程路由能注入 AbortReq。
- test/registered/unit/managers/scheduler_components/test_output_sender.py（模块 输出路由；类别 test；类型 test-coverage；符号 _make_scheduler_req, TestSenderWrapper, test_preserves_existing_output_route, test_scheduler_abort_routes_to_origin_worker）：新增 CPU 回归测试，覆盖真实调度器 Req 到 MultiTokenizerRouter 分发的完整链路，并验证已路由输出不被覆盖，是本修复正确性的关键保障。

关键符号：SenderWrapper.send_output, _make_scheduler_req, test_preserves_existing_output_route, test_scheduler_abort_routes_to_origin_worker, MultiTokenizerRouter._distribute_result_to_workers

关键源码片段

python/sglang/srt/managers/scheduler_components/output_sender.py

修复核心文件。send_output 是调度器输出统一出口，本 PR 在此放宽 recv_obj 类型、用 getattr 读取 http_worker_ipc 并把 isinstance 判断移到 output 上，使调度器内部 Req 的回程路由能注入 AbortReq。

```
# python/sglang/srt/managers/scheduler_components/output_sender.py
from typing import Optional, Union

import zmq

from sglang.srt.managers.io_struct import BaseBatchReq, BaseReq, sock_send


class SenderWrapper:
    """调度器输出统一出口：把输出消息发往 tokenizer 侧对应的 worker。"""

    def __init__(self, socket: zmq.Socket):
        self.socket = socket

    def send_output(
        self,
        output: Union[BaseReq, BaseBatchReq],
        recv_obj: Optional[object] = None,
    ):
        if self.socket is None:
            return

        # 调度器内部 Req 不是 BaseReq，但同样携带来源 worker 的回程信息；
        # 用 getattr 安全读取，避免依赖具体类型。
```

```
        http_worker_ipc = getattr(recv_obj, "http_worker_ipc", None)
        if (
            isinstance(output, BaseReq)
            and http_worker_ipc is not None
            and output.http_worker_ipc is None
        ):
            # 仅当输出本身尚未指定目的地时才补全路由，
            # 保证已显式路由的控制消息（如已路由的 AbortReq）不被覆盖。
            output.http_worker_ipc = http_worker_ipc

        sock_send(self.socket, output)
```

test/registered/unit/managers/scheduler_components/test_output_sender.py

新增 CPU 回归测试，覆盖真实调度器 Req 到 MultiTokenizerRouter 分发的完整链路，并验证已路由输出不被覆盖，是本修复正确性的关键保障。

```
# test/registered/unit/managers/scheduler_components/test_output_sender.py（节选）
import asyncio
from unittest.mock import MagicMock, patch

from sglang.test.test_utils import CustomTestCase, maybe_stub_sgl_kernel
from sglang.srt.managers.io_struct import AbortReq
from sglang.srt.managers.multi_tokenizer_mixin import MultiTokenizerRouter
from sglang.srt.managers.schedule_batch import Req
from sglang.srt.managers.scheduler_components.output_sender import SenderWrapper
from sglang.srt.sampling.sampling_params import SamplingParams

maybe_stub_sgl_kernel()


def _make_scheduler_req(http_worker_ipc: str) -> Req:
    # 调度器侧的 Req 不是 IPC 消息类型，但携带来源 worker 的 http_worker_ipc
    return Req(
        rid="scheduler-request",
        origin_input_text="prompt",
        origin_input_ids=[1],
        sampling_params=SamplingParams(),
        http_worker_ipc=http_worker_ipc,
    )


class TestSenderWrapper(CustomTestCase):
    @patch("sglang.srt.managers.scheduler_components.output_sender.sock_send")
    def test_scheduler_abort_routes_to_origin_worker(self, mock_sock_send):
        """队列溢出产生的 AbortReq 必须回到来源 tokenizer worker。"""
        socket = MagicMock()
        worker_ipc = "ipc:///tokenizer-worker-2"
        # AbortReq 构造时未指定目的地，路由需从调度器 Req 继承
```

```
    output = AbortReq(rid="overflow-request")

    SenderWrapper(socket).send_output(output, _make_scheduler_req(worker_ipc))

    routed_output = mock_sock_send.call_args.args[1]
    # 用真实 MultiTokenizerRouter 的分发逻辑验证最终目的地
    router = MultiTokenizerRouter.__new__(MultiTokenizerRouter)
    router.socket_mapping = MagicMock()
    asyncio.run(router._distribute_result_to_workers(routed_output))

    router.socket_mapping.send_output.assert_called_once_with(
        worker_ipc, routed_output
    )
```

评论区精华

作者 jeremyzhang866 强调修复刻意保持最小化（"The runtime diff is intentionally small"），回归测试使用真实调度器 Req 与 MultiTokenizerRouter 的最终分发路径而非纯 mock。维护者 cctry 执行了多轮 /rerun-test：早期失败源于缺少 run-ci 标签导致 CI 门禁未真正运行，以及 py3.10 x86_64 CPU 安装失败（rebase 纳入 #34321 的 cuda-tile==1.6.0rc5 pin 后修复），与本次逻辑无关。最终 6 个相关测试全绿，cctry 给出结论 "looks good to me. merge now" 并合入。

- 修复方案与回归测试设计确认 (design): 方案通过，cctry 评价 "looks good to me. merge now" 并合入，未提出额外修改意见。
- CI 门禁与 rerun-test 流程 (other): rebase 后的新 head 上 6 个相关测试全绿，CI 门禁通过后合入。

风险与影响

- 风险：
  1. SenderWrapper.send_output 是调度器所有输出（token 输出、AbortReq 等控制消息）的统一出口，改动影响所有调用方。条件改为 isinstance(output, BaseReq) 后，BaseBatchReq 输出不再从 recv_obj 继承路由，若存在依赖旧行为的调用点可能出现行为变化，需要确认 batch 输出路径已显式携带目的地。
  2. recv_obj 类型放宽为 object 并用 getattr 读取后，任何携带 http_worker_ipc 属性的对象都会参与路由补全，语义边界变宽，未来新增类型时需注意该字段的含义一致性。
  3. 修复有效性依赖 Req 构造时正确设置 http_worker_ipc，若某条路径未设置该字段，问题仍会复现。
  4. CI 曾因 py3.10 CPU 安装问题失败（与本次改动无关），提示该仓库 CPU 测试环境对依赖 pin 敏感。 - 影响：用户侧：多 tokenizer 模式 + 队列溢出场景从 " 无限挂起 " 变为快速返回 HTTP 503，超载行为可预期，客户端不再长时间占用连接。系统侧：改动仅为一次 getattr 查找，无性能影响；控制消息路由完整性提升，减少 ESTABLISHED 连接泄漏。团队侧：该修复为多 tokenizer 控制消息路由提供了端到端回归测试范式（真实 Req + 真实 MultiTokenizerRouter 分发），后续修改 SenderWrapper 有兜底保障。 -

风险标记：核心输出路径变更，类型放宽语义边界，回程路由依赖 Req 字段

关联脉络

- PR #33858 [Bug] Requests hang indefinitely on queue overflow with --tokenizer-worker-num 2: 关联 Issue（非 PR）：本 PR 修复的正是该 issue 描述的请求无限挂起问题，复现条件与验证数据均来自此 issue。