PR #24445 完整报告

sgl-project/sglang

[fix] /pause_generation and /continue_generation wrong for --tokenizer-worker-num > 1

合并时间：2026-05-06 07:27

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24445

执行摘要

- 一句话：修复多 tokenizer worker 下暂停 / 继续生成不一致的 bug
- 推荐动作：推荐精读 MultiTokenizerRouter.router_worker_obj 的广播实现和 MultiHttpWorkerTokenizerMixin 的广播处理逻辑，可作为多进程间状态广播的参考设计。建议关注 follow-up：通过 op_id 映射修复并发竞态条件。

功能与动机

Issue #21235 报告：多 tokenizer worker 模式下，暂停 / 继续生成只作用于处理请求的那个 worker，概率性导致部分 worker 永久处于暂停状态，后续路由到该 worker 的请求无限等待。共享调度器的 _engine_paused 标志正确切换，但 per-worker 的请求门控 (is_pause_cond) 被破坏。

实现拆解

1. 新增数据类：在 io_struct.py 中添加 TokenizerWorkerRegisterReq（worker 启动时向路由器注册 IPC 名称）和 PauseContinueBroadcast（路由器广播暂停 / 继续状态）。
2. 路由器改造：在 MultiTokenizerRouter 中维护 all_worker_ipcs 集合（记录所有注册过的 worker IPC），router_worker_obj 协程在处理 PauseGenerationReqInput/ContinueGenerationReqInput 时，遍历集合通过 socket_mapping.send_output 广播 PauseContinueBroadcast；同时保留向 scheduler rank 0 的转发（abort 模式除外）。
3. Worker 端处理：MultiHttpWorkerTokenizerMixin 在初始化时发送 TokenizerWorkerRegisterReq 注册自身 IPC，并新增 _pause_continue_future 和 _handle_pause_continue_broadcast/_apply_pause_continue_broadcast 方法处理广播：接收到广播后更新 is_pause 标志并通知 is_pause_cond，同时通过 Future 同步等待本地标志翻转后再返回 HTTP 响应。
4. 测试与配套：PR 描述中列出了手动回归测试和现有单元测试的通过计划，但本次提交未包含新增的自动化测试文件；handle_loop 的响应路由路径也共享了 socket_mapping，避免重复创建。

关键文件：

- python/sglang/srt/managers/multi_tokenizer_mixin.py（模块 请求路由；类别 source；类型 core-logic；符号 pause_generation, continue_generation, _handle_pause_continue_broadcast, _apply_pause_continue_broadcast）：核心逻辑变

更：实现了暂停 / 继续信号从路由器到所有 tokenizer worker 的广播机制，包括 worker注册、广播发送和响应处理。

- python/sglang/srt/managers/io_struct.py（模块 数据结构；类别 source；类型 core-logic；符号 TokenizerWorkerRegisterReq, PauseContinueBroadcast）：新增 TokenizerWorkerRegisterReq 和 PauseContinueBroadcast 两个 dataclass，定义了 worker 注册和广播的数据契约。

关键符号：pause_generation, continue_generation, _handle_pause_continue_broadcast, _apply_pause_continue_broadcast

关键源码片段

python/sglang/srt/managers/multi_tokenizer_mixin.py

核心逻辑变更：实现了暂停 / 继续信号从路由器到所有 tokenizer worker 的广播机制，包括 worker 注册、广播发送和响应处理。

```
async def router_worker_obj(self):
    """Forward path: workers → scheduler, with pause/continue broadcast."""
    while True:
        recv_obj = await self.receive_from_worker.recv_pyobj()

        if isinstance(recv_obj, TokenizerWorkerRegisterReq):
            # 每个 TokenizerWorker 启动时注册自己的 IPC 名称，避免重复添加
            if recv_obj.worker_ipc_name not in self.all_worker_ipcs:
                self.all_worker_ipcs.add(recv_obj.worker_ipc_name)
                logger.info(
                    f"Router registered worker IPC: {recv_obj.worker_ipc_name} "
                    f"(total: {len(self.all_worker_ipcs)})"
                )
            continue

        if isinstance(
            recv_obj, (PauseGenerationReqInput, ContinueGenerationReqInput)
        ):
            # 广播暂停 / 继续信号到所有已注册 worker
            is_pause = isinstance(recv_obj, PauseGenerationReqInput)
            broadcast = PauseContinueBroadcast(is_pause=is_pause)
            for ipc_name in self.all_worker_ipcs:
                self.socket_mapping.send_output(ipc_name, broadcast)

            # abort 模式通过轮询释放，不需要转发到 scheduler
            if not (
                isinstance(recv_obj, PauseGenerationReqInput)
                and recv_obj.mode == "abort"
            ):
                # 只需转发到 scheduler rank 0，内部会广播到所有 TP/PP/DP ranks
                await self.send_to_scheduler.send_pyobj(recv_obj)
            continue
```

```
# 普通请求直接转发给 scheduler
await self.send_to_scheduler.send_pyobj(recv_obj)
```

评论区精华

Review 中重点关注了三个技术点：(1) socket_mapping.send_output 是同步阻塞调用，在 async 循环中使用可能导致性能问题或全局挂起；(2) 建议用 set 替代 list 存储 worker IPC 名称，避免 worker 重启时重复注册；(3) 单个 _pause_continue_future 在并发暂停 / 继续请求到同一 worker 时会因覆盖导致前一个请求挂起。前两个建议已被作者采纳，第三个作为 follow-up 记录。

- 同步 ZMQ 调用在 async 循环中的性能风险 (performance): 未在评论中明确解决，但性能影响被作者认为可接受（延迟 sub-ms），PR 合并时未引发进一步争论
- 使用 set 而非 list 存储 worker IPC 名称 (design): 作者在第三次提交中接受建议，将 all_worker_ipcs 从 list 改为 set，并调整注册逻辑去重
- 单个 future 在并发暂停 / 继续请求下的竞态条件 (design): PR 作者在 body 的 Notes/follow-ups 中承认此问题，认为顺序使用（bug 重现场景）没问题，并建议后续使用 op_id 键化的 future map 加固

风险与影响

- 风险：
  - 同步 ZMQ 调用：socket_mapping.send_output 在 router_worker_obj 和 handle_loop 的 async 循环中阻塞，若 worker 处理慢或 ZMQ 缓冲满，可能导致整体延迟增长甚至事件循环挂起。当前场景下广播仅涉及同主机 IPC，延迟 sub-ms，风险可控。
  - 并发竞态条件：同一 worker 进程上连续收到多个暂停 / 继续请求时，_pause_continue_future 可能被覆盖，导致前端请求挂起。PR 明确指出后续需用 op_id 键化的 future map 修复。
  - worker 注册时序依赖：依赖所有 worker 在 HTTP 流量开始前完成注册，当前启动顺序满足条件，但若未来动态增减 worker，可能注册不完全。
  - 缺少测试覆盖：PR 在 body 中列出了测试计划，但本次提交未包含自动化回归测试，后续若重构可能引入回归。
- 影响：
  - 用户影响：修复了多 tokenizer worker 部署下暂停 / 继续生成命令不可用的问题，提升 PD 分离场景的可靠性。所有使用 --tokenizer-worker-num > 1 的用户均受益。
  - 系统影响：新增的广播过程引入了一次额外的 IPC 往返（worker → router → worker），同主机环境下延迟可忽略。无 ABI/API 破坏。
  - 团队影响：为后续类似广播模式提供了可复用的 all_worker_ipcs 注册和 PauseContinueBroadcast 机制，降低未来扩展成本。
  - 风险标记：核心路径变更，同步阻塞调用，并发竞态条件，缺少测试覆盖

关联脉络

- 暂无明显关联 PR