PR #33118 完整报告

sgl-project/sglang

[PD] Fix false health-503 during decode retraction re-admission

合并时间：2026-08-04 00:44

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33118

执行摘要

- 一句话：修复 PD 解码回退期间健康探测误报 503
- 推荐动作：建议优先阅读 python/sglang/srt/managers/scheduler.py 中 is_fully_idle 与 on_idle 的改动。这是 PD 健康检查核心路径上一个精准的小修复，展示了健康探测与调度队列语义的耦合方式：探测不能只依赖 running_batch，还要把自愈中的回退队列算作活跃。值得关注的设计点是 _engine_paused 豁免和 on_idle 信号补发时序。若团队后续完善 PD 健康检查，应补充针对 retracted_queue 卡住、暂停并发等场景的单元测试。

功能与动机

PR body 明确指出：is_fully_idle(for_health_check=True) 没有统计 retracted_queue，一旦 running_batch/waiting_queue 排空，健康探测会认为引擎空闲，注入的 fake-bootstrap 请求直到重新接纳完成前无法推进，超时转为 503，而“A router acting on that 503 can eject a healthy decode engine and fail requests with no available engines”。非 PD 模式不受影响，因为被回退的请求会回到 waiting_queue，该队列本来就被统计。

实现拆解

本 PR 只修改 python/sglang/srt/managers/scheduler.py，共 2 处核心逻辑调整：

1. is_fully_idle 健康检查分支补齐 retracted_queue 统计：在原有的 idle &= len(self.waiting_queue) == 0 之后，新增一个带条件的前置判断。仅当满足 for_health_check=True、引擎未暂停（not self._engine_paused）、disaggregation_mode == DisaggregationMode.DECODE 且 prealloc 队列对象存在时，才把 len(self.disagg_decode_prealloc_queue.retracted_queue) == 0 并入 idle 条件。这样解码回退再接纳期间引擎被判定为非空闲，健康探测不会误注入 fake-bootstrap 请求。
2. on_idle 入口补发延迟的健康检查信号：在原有 if not self.is_fully_idle(): return 之前插入 self.maybe_send_health_check_signal()，把引擎繁忙期间推迟的信号在事件循环空闲检查开始时立即发送，避免探测等待到下一个调度周期。
3. 条件收敛与兼容性：非健康检查路径原有的 retracted_queue 统计保持不变；引擎暂停（_engine_paused）时沿用旧语义，不把该队列计入忙碌，避免与暂停逻辑冲突。
4. 测试与配套：本 PR 未新增自动化测试，也未改动配置、schema 或部署脚本；第二个 commit 为 merge main 解决冲突。修复在内部 PD 环境验证：KV 压力下的解码回退不再使 /health_generate 在重新接纳期间翻转 503。

关键文件：

- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 core-logic；符号 is_fully_idle, on_idle）：唯一的变更文件，承载 PD 健康检查空闲判定的核心逻辑：is_fully_idle 补统计 retracted_queue，on_idle 补发延迟健康检查信号，直接决定 /health_generate 的探测行为。

关键符号：on_idle, is_fully_idle, maybe_send_health_check_signal

关键源码片段

python/sglang/srt/managers/scheduler.py

唯一的变更文件，承载 PD 健康检查空闲判定的核心逻辑：is_fully_idle 补统计 retracted_queue，on_idle 补发延迟健康检查信号，直接决定 /health_generate 的探测行为。

```
def on_idle(self):
    """Idle housekeeping: guard, check, metrics, reset, sleep."""
    # 引擎繁忙期间延迟的健康检查信号在这里补发，
    # 避免探测一直等到下一个空闲周期才被应答
    self.maybe_send_health_check_signal()

    if not self.is_fully_idle():
        return

    # 以下为原有空闲收尾逻辑：统一内存 flush、内存泄漏检查、
    # tree cache 检查、指标上报、KV 事件发布、负载快照与睡眠等
    # （完整实现详见仓库 head 版本）


def is_fully_idle(self, for_health_check=False) -> bool:
    # 健康检查依赖 process_output 中真实运行的请求来带活；
    # disagg 各类队列（bootstrap / prealloc / transfer）可能只有
    # 排队请求却没有正在 GPU 上执行的批次（例如握手卡住、KV 满、
    # 传输停滞），因此它们不能单独作为健康依据。
    idle = (
        self.running_batch.is_empty()
        and self.chunked_req is None
        and not self.dllm_manager.any_staging_reqs()
        and (self.last_batch is None or self.last_batch.is_empty())
        and (not self.enable_overlap or len(self.result_queue) == 0)
        and self._pp_microbatches_drained()
    )

    # 普通等待队列：waiting + bootstrapping + preallocation + kv transfer (decode)
    idle &= len(self.waiting_queue) == 0

    # 健康检查路径：decode 引擎在 KV 压力下会把运行中的请求回退到
    # retracted_queue 等待重新接纳，此时引擎仍在自愈，必须视为活跃；
    # 但引擎显式暂停（_engine_paused）时不干扰原探测语义。
```

```
    if (
        for_health_check
        and not self._engine_paused
        and self.disaggregation_mode == DisaggregationMode.DECODE
        and self.disagg_decode_prealloc_queue is not None
    ):
        idle &= len(self.disagg_decode_prealloc_queue.retracted_queue) == 0

    if not for_health_check:
        # 非健康检查路径：grammar 队列、prefill inflight 队列等
        # 也会影响引擎是否空闲的判断。
        idle &= len(self.grammar_manager.grammar_queue) == 0
        if self.disaggregation_mode == DisaggregationMode.PREFILL:
            idle &= len(self.disagg_prefill_inflight_queue) == 0
            idle &= len(self.disagg_prefill_bootstrap_queue.queue) == 0
        if self.disaggregation_mode == DisaggregationMode.DECODE:
            idle &= len(self.disagg_decode_prealloc_queue.queue) == 0
            idle &= len(self.disagg_decode_prealloc_queue.retracted_queue) == 0
            idle &= len(self.disagg_decode_transfer_queue.queue) == 0
            if self.decode_offload_manager is not None:
                idle &= len(self.decode_offload_manager.ongoing_offload) == 0
        # HiSparse staging 与 HiCache 异步操作检查此处省略

    return idle
```

评论区精华

本 PR 无实质 review 评论，也没有 review 线程。Issue 区仅有两条自动化消息：Gemini Code Assist 通知（消费者版已停止服务，无代码审查活动）以及作者触发的 /tag-and-rerun-ci CI 重跑指令。因此缺少关于 _engine_paused 豁免语义、retracted_queue 长期卡住时健康检查行为等问题的公开讨论。

- 暂无高价值评论线程

风险与影响

- 风险：改动位于调度器健康检查核心路径，主要风险如下：

1. retracted_queue 非空即视为忙碌的语义风险（python/sglang/srt/managers/scheduler.py）：如果 retracted_queue 因异常长期卡住（例如重新接纳流程未完成），健康检查会持续判定引擎非空闲，可能掩盖引擎失联。这是刻意取舍：引擎处于自愈中就不应响应探测。
2. _engine_paused 豁免的不确定性：暂停状态下条件不成立，retracted_queue 不会被计入忙碌，此时健康探测可能给出 idle 结论。暂停与回退并发时的语义需要确认，现有代码注释未展开说明。
3. on_idle 信号时序变化：maybe_send_health_check_signal() 被提前到 idle 判断之前调用，虽然只应在存在延迟信号时发送，但其幂等性和并发时序未经测试覆盖，可能引入重复信号。

4. 缺少自动化测试：本 PR 没有配套测试文件，修复仅靠内部 PD 环境验证，回归风险由后续 CI 承担。 - 影响：影响范围集中在 PD（prefill-decode disaggregation）部署的调度器健康检查路径：修复后，解码引擎在 KV 压力下执行回退再接纳（retraction re-admission）期间，/health_generate 不会误报 503，路由器不会因此剔除健康引擎，也不会出现 no available engines 的连锁请求失败。非 PD 模式完全不受影响，因为新增逻辑严格限定在 for_health_check 且 DisaggregationMode.DECODE 条件下。改动仅增加 11 行判断，不改变 GPU 执行路径与性能特征，对团队而言提升了 PD 场景的运维稳定性。 - 风险标记：核心路径变更，缺少测试覆盖，条件行为依赖队列语义

关联脉络

- PR #31901 [HiSparse]Fix DeepSeek V4 HiSparse PD Transfers with Separate Host and Device KV Indices: 同为 PD（prefill-decode 分离）场景下的调度与传输修复，涉及 decode 引擎对回退 / 转移请求的处理，与本 PR 的 retracted_queue 健康检查语义相关。
- PR #33336 config: keep runtime hicache and weight-version updates off ServerArgs: 同改 python/sglang/srt/managers/scheduler.py，将运行时配置读写逻辑迁出 ServerArgs，属于同一调度器模块的配置演进，阅读时可对照理解调度器事件循环的整体结构。
- PR #33334 config: stop writing config onto the published ServerArgs at three sites: 同样修改 python/sglang/srt/managers/scheduler.py 及 attention 相关模块，属于调度器配置路径的重构线，与本 PR 在文件级有交集。