# PR #27028 完整报告

sgl-project/sglang

Fix orphaned aborted prefill bootstrap requests in PP disaggregation

合并时间：2026-06-02 19:03

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27028

## 执行摘要

- 一句话：修复 PP 下预填充引导请求孤儿问题
- 推荐动作：该 PR 值得审阅，尤其是对涉及 PP 分解和分布式系统状态的开发者。变更逻辑清晰，但缺少对应的单元测试。建议后续添加测试以覆盖 PP 分解中 bootstrap 请求被中止的场景。

## 功能与动机

修复 Issue #26912：PP 分解模式下，非第一个 PP rank 上被中止的预填充引导请求会永远残留在引导队列中，导致 num_prefill_bootstrap_queue_reqs 指标在无流量时也不为 0。这带来了观测性上的误导和潜在的资源泄漏。

## 实现拆解

变更集中在 python/sglang/srt/disaggregation/prefill.py 中的 pop_bootstrapped 方法。

1. 修改循环过滤条件：原本的逻辑是如果 rids_to_check 不为 None，且请求的 rid 不在该集合中，则跳过该请求（continue）。新的逻辑改为：只有当 rids_to_check 不为 None 且请求的 rid 不在其中，并且该请求的轮询结果 不是KVPoll.Failed 时，才跳过。
2. 核心意图：对于成功或仍在引导中的请求，仍然需要跨 rank 共识，因此如果 rid 不在检查集合中（说明前一个 rank 已经移除该请求），当前 rank 也应跳过。但对于 本地失败的请求（poll == KVPoll.Failed），说明该请求在当前 rank 上已经出现了异常（例如 KV 传输错误），这种本地失败是终局的，必须被清理，即使前一个 rank 已经把它从共识列表中移除。
3. 副作用：这次修复确保被中止的引导请求能够被正常弹出、记录错误日志、输出流输出、并从队列中移除，从而释放相关资源并让引导队列计数归零。

关键文件：

- python/sglang/srt/disaggregation/prefill.py（模块 分解器；类别 source；类型 core-logic；符号 pop_bootstrapped）：该文件是 PR 中唯一被修改的文件，其中的 pop_bootstrapped 方法负责在 PP 分解模式下弹出已完成或失败的引导请求。修复的核心逻辑在此。

关键符号：pop_bootstrapped

## 关键源码片段

## python/sglang/srt/disaggregation/prefill.py

该文件是 PR 中唯一被修改的文件，其中的 pop_bootstrapped 方法负责在 PP 分解模式下弹出已完成或失败的引导请求。修复的核心逻辑在此。

```
# python/sglang/srt/disaggregation/prefill.py

class PrefillBootstrapQueue:
    # ... 其他方法 ...

    def pop_bootstrapped(
        self,
        return_failed_reqs: bool = False,
        rids_to_check: Optional[List[str]] = None,
    ) -> List[Req]:
        """
        Pop the reqs which has finished bootstrapping.

        return_failed_reqs: For PP, on rank 0, also return the failed reqs to notify the next rank.
        rids_to_check: For PP, on rank > 0, check the rids from the previous rank
                       has consensus with the current rank.
        """

        bootstrapped_reqs = []
        failed_reqs = []
        indices_to_remove = set()

        if len(self.queue) == 0:
            if return_failed_reqs is False:
                return []
            else:
                return [], []

        polls = poll_and_all_reduce_attn_cp_tp_group(
            [req.disagg_kv_sender for req in self.queue],
            self.scheduler.attn_cp_cpu_group,
            self.scheduler.attn_tp_cpu_group,
        )

        for i, (req, poll) in enumerate(zip(self.queue, polls)):
            if (
                rids_to_check is not None
                and req.rid not in rids_to_check
                and poll != KVPoll.Failed # ② 关键修复：即使 rid 不在检查集合中，
                                          # 如果本地已失败（Failed），也必须处理
            ):
                # In PP mode, successful bootstrap still requires cross-rank
                # consensus. Local failures are terminal and must be drained
                # even if an earlier PP rank has already removed the request.
```

```
            continue

        if poll == KVPoll.Bootstrapping:
            continue
        elif poll == KVPoll.Failed:
            error_message = f"Prefill bootstrap failed for request rank={self.tp_rank} {req.rid=}
            {req.bootstrap_room=}"
            try:
                req.disagg_kv_sender.failure_exception()
            except Exception as e:
                error_message += f" with exception {e}"
            logger.error(error_message)
            req.time_stats.trace_ctx.abort(abort_info={"reason": error_message})
            prepare_abort(
                req, error_message, status_code=HTTPStatus.INTERNAL_SERVER_ERROR
            )
            self.scheduler.output_streamer.stream_output([req], req.return_logprob)
            indices_to_remove.add(i)
            failed_reqs.append(req)
            if self.scheduler.metrics_reporter.enable_metrics:
                self.scheduler.metrics_collector.increment_bootstrap_failed_reqs()
            if self.scheduler.enable_hicache_storage:
                # to release prefetch events associated with the request
                self.scheduler.tree_cache.release_aborted_request(req.rid)
            continue

        # KVPoll.WaitingForInput - decode is ready to receive. initialize the kv sender
        req.time_stats.set_bootstrap_done_time()
        # ... 后续处理成功引导的请求 ...
```

## 评论区精华

没有实质性的 review 讨论。合并者 ShangmingCai 给出了 'LGTM, thanks for the fix.' 的批准。代码机器人 gemini-code-assist 自动评论，没有提出具体问题。

- 暂无高价值评论线程

## 风险与影响

- 风险：风险较低。变更仅为单行条件逻辑的调整，且专门针对已失败（KVPoll.Failed）的请求。没有影响成功引导或仍在引导中的请求的正常流程。但需要注意：如果某个请求在非第一个 PP rank 上本地失败，但却不在前一个 rank 的 rids_to_check 中，当前 rank 会独立清理并上报错误，这可能导致潜在的多重错误上报（但这是正确的行为，因为每个 rank 都应该独立处理本地失败）。
- 影响：影响范围：仅限于 PP 分解模式下的非第一个 PP rank。修复后，这些 rank 上的引导队列在空闲时能正确归零，修复了观测性指标的误导问题。资源泄漏（队列中的请求永远不会被释放）也被修复。对模型精度和推理性能无影响。
- 风险标记：缺少测试覆盖

# 关联脉络

- PR #26780 [PD] Optimistic prefill: 该 PR 是同一个 PP 分解功能线的扩展，引入了乐观预填充，可能也涉及类似的 bootstrap 队列交互。
- PR #26227 [PD]: Support HiCache prefetching and pd-incremental transfer on decode side: 同样是 PP 分解（PD）方向的 PR，增加了 decode 端的 HiCache 预取和增量传输，与本 PR 的引导队列相关。