PR #31869 完整报告

sgl-project/sglang

[PD+PP] Honor PP consensus for bootstrap and prealloc

合并时间：2026-07-30 01:19

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/31869

执行摘要

- 一句话：修复 PD+PP 中 bootstrap 和 prealloc 的共识竞态问题
- 推荐动作：此 PR 设计清晰，值得精读：通过"共识映射"避免二次同步的策略可推广至其他多 rank 协调场景；处理局部失败的设计考虑周全。关注 poll_and_all_reduce_pp 函数和 pop_bootstrapped 中的 PP/ 非 PP 分支选择。

功能与动机

在 PD 解耦部署中，Prefill bootstrap 和 Decode preallocation 已通过 all-reduce 计算 PP consensus（global_ready 和 global_failed）。但队列消费者在 consensus 后再次轮询本地 sender/receiver 状态，导致 consensus 与队列处理间到达的 abort 可能使 PP 阶段应用不同结果而分歧。此 PR 遵循 #31030 的 review comments，范围限于 Prefill bootstrap 和 Decode preallocation，避免二次轮询。

实现拆解

1. 新增映射函数：在 python/sglang/srt/disaggregation/utils.py 中添加 poll_and_all_reduce_pp，利用传入的 good_rids/bad_rids 集合直接返回 poll 结果，避免再次执行 all-reduce。
2. Prefill bootstrap 改造：在 python/sglang/srt/disaggregation/prefill.py 的 pop_bootstrapped 中，当 self.pp_size > 1 时调用新函数，并对未覆盖的请求（consensus 中不存在）执行本地 poll_and_all_reduce_attn_cp_tp_group 仅捕获 Failed 状态，防止资源泄漏。
3. Decode prealloc 改造：在 python/sglang/srt/disaggregation/decode.py 的 _update_handshake_waiters 和 pop_preallocated 中同理，PP 模式下使用共识结果，非 PP 模式保持原有本地轮询。
4. 调度器 PP 混入层适配：在 python/sglang/srt/managers/scheduler_pp_mixin.py 的 process_bootstrapped_queue 和 process_prealloc_queue 中将旧参数 rids_to_check 替换为 pp_good_rids 和 pp_bad_rids。
5. 测试兼容：在三个测试文件中为新增加的 pp_size 属性设置默认值 1，确保非 PP 用例继续通过。

关键文件：

- python/sglang/srt/disaggregation/utils.py（模块 解耦工具；类别 source；类型 dependency-wiring；符号 poll_and_all_reduce_pp）：新增核心函数 poll_and_all_reduce_pp，实现共识到 poll 结果的直接映射，是本次修复的基础。
- python/sglang/srt/disaggregation/prefill.py（模块 预填充处理；类别 source；类型 core-logic）：在 pop_bootstrapped 中增加 PP 分支，使用共识结果并处理局部失败，是 Prefill bootstrap 正确性的关键修改。
- python/sglang/srt/disaggregation/decode.py（模块 解码处理；类别 source；类型 core-logic）：在 _update_handshake_waiters 和 pop_preallocated 中应用一致修改，影响 Decode preallocation 路径。
- python/sglang/srt/managers/scheduler_pp_mixin.py（模块 调度 PP 层；类别 source；类型 core-logic）：适配新接口：将 process_bootstrapped_queue 和 process_prealloc_queue 中的旧参数替换为 pp_good_rids/pp_bad_rids。
- test/registered/unit/disaggregation/test_decode_queue_cleanup.py（模块 解码测试；类别 test；类型 test-coverage）：为 DecodePreallocQueue 测试设置 pp_size = 1 以兼容新分支。
- test/registered/unit/managers/test_priority_scheduling_disaggregation.py（模块 调度测试；类别 test；类型 test-coverage）：类似地设置 pp_size = 1 维持非 PP 测试覆盖率。
- test/registered/unit/mem_cache/test_decode_radix_lock_ref.py（模块 锁测试；类别 test；类型 test-coverage）：同样添加 pp_size = 1 设置，保持测试通过。

关键符号：poll_and_all_reduce_pp, pop_bootstrapped, pop_preallocated, _update_handshake_waiters, process_bootstrapped_queue, process_prealloc_queue

关键源码片段

python/sglang/srt/disaggregation/utils.py

新增核心函数poll_and_all_reduce_pp，实现共识到poll结果的直接映射，是本次修复的基础。

```
def poll_and_all_reduce_pp(
    rids: Iterable[str], # 所有待检查的请求 ID
    ready_poll: int, # 共识通过时返回的 poll 状态（如 KVPoll.WaitingForInput）
    pp_good_rids: Optional[List[str]] = None, # 所有 PP stage 认为 ready 的 RID 集合
    pp_bad_rids: Optional[List[str]] = None, # 所有 PP stage 认为 failed 的 RID 集合
) -> List[Optional[int]]:
    """将权威的 PP 共识映射为 poll 状态，无需再次轮询本地状态。"""
    if pp_good_rids is None or pp_bad_rids is None:
        raise ValueError("PP consensus is required")

    # 转为 set 实现 O(1) 查找，failure 优先
    good_rids = set(pp_good_rids)
    bad_rids = set(pp_bad_rids)
    return [
        KVPoll.Failed if rid in bad_rids # 任何 stage 失败 => Failed
        else ready_poll if rid in good_rids # 所有 stage 准备好 => ready_poll
        else None # 未在共识中 => 需要本地检查
        for rid in rids
```

```
    ]
```

python/sglang/srt/disaggregation/prefill.py

在 pop_bootstrapped 中增加 PP 分支，使用共识结果并处理局部失败，是 Prefill bootstrap 正确性的关键修改。

```
def pop_bootstrapped(
    self,
    return_failed_reqs: bool = False,
    pp_good_rids: Optional[List[str]] = None,
    pp_bad_rids: Optional[List[str]] = None,
) -> List[Req] | tuple[List[Req], List[Req]]:
    """
    pop 已完成 bootstrap 的请求。
    PP 模式下使用传入的共识结果，无需再次轮询本地 sender。
    """
    bootstrapped_reqs = []
    failed_reqs = []
    indices_to_remove = set()

    if not self.queue:
        return ([], []) if return_failed_reqs else []

    if self.pp_size > 1:
        # PP 模式：使用权威共识结果
        polls = poll_and_all_reduce_pp(
            (req.rid for req in self.queue),
            KVPoll.WaitingForInput,
            pp_good_rids,
            pp_bad_rids,
        )
        uncovered = [i for i, poll in enumerate(polls) if poll is None]
        if uncovered:
            # 对未覆盖的请求做本地轮询，只捕获 Failed，防止泄漏
            local_polls = poll_and_all_reduce_attn_cp_tp_group(
                [self.queue[i].disagg_kv_sender for i in uncovered],
                self.scheduler.attn_cp_cpu_group,
                self.scheduler.attn_tp_cpu_group,
            )
            for i, local_poll in zip(uncovered, local_polls):
                if local_poll == KVPoll.Failed:
                    polls[i] = KVPoll.Failed
    else:
        # 非 PP 模式：传统本地轮询
        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 poll is None:
        continue # 未决定，下次再处理
    if poll == KVPoll.Failed:
        # 失败处理 ...
```

评论区精华

> 命名改进(ShangmingCai): “I think the naming should be improved. Not yet success, just good to check.” 最终采用 ready_rids 语义，避免与 successful_rids 混淆（bootstrap 仅为 ready 而非 transfer 成功）。 防御性分支(ShangmingCai): “How about checking self.pp_size > 1 first, then run into this logic block?” 作者在后续版本中添加了 if self.pp_size > 1 检查，PP=1 时走原始路径。 未覆盖请求处理(ShangmingCai): “Local failures are terminal and must be drained even if PP consensus no longer covers the request…” 作者对 uncovered 请求增加本地轮询，仅应用 Failed 状态，避免元数据泄漏。 移除简化(ShangmingCai): “_apply_handshake_polls here is weird.” 作者移除了该抽象，在 decode prealloc 中直接应用共识结果及小助手 _abort_handshake。

- 函数命名与语义 (design): 采用 ready_rids 和 ready_poll 避免误导，文件为 poll_and_all_reduce_pp。
- PP 分支的防御性检查 (design): 作者在 prefill 和 decode 中均添加了 if pp_size > 1: 分支，PP=1 走原始路径。
- 未覆盖请求的局部失败处理 (correctness): 仅对 uncovered 请求执行本地轮询，只应用 Failed 状态；good 未覆盖的请求会留到下一周期。
- 移除多余抽象 (refactor): 作者移除该函数，在 decode prealloc 中直接使用共识结果和 _abort_handshake 辅助。

风险与影响

- 风险：
  1. 局部失败及时序风险：PP 共识后、队列处理前，某 rank 本地可能发生失败。对于共识中的 good 请求，代码不再轮询，若本地失败未被之前共识捕获（极低概率），该请求可能被错误认为良好，但共识 all-reduce 已汇集最近状态，风险可控。
  2. 未覆盖请求的遗漏：若 uncovered 请求本地轮询也返回 None（既非 Failed 也非 good），该请求会被跳过，下个周期继续处理，不会泄漏（代码中 continue 但仍在队列中）。
  3. 无性能风险：非 PP 路径完全不变，PP 路径反而减少一次 all-reduce。 - 影响：用户：修复了 PD+PP 部署下可能导致请求永久阻塞或被错误中止的竞态条件，提升可靠性。系统：控制面延迟从 ~50μs 降至 ~0.3μs（PP 路径），减少同步开销。团队：代码更加简洁，移除了冗余轮询逻辑和复杂参数 rids_to_check，增强可维护性。影响范围仅限于 PD + PP > 1 的场景。
- 风险标记：局部失败时序风险，共识窗口竞态，测试覆盖仅 base PP=1

关联脉络

- PR #31030 [PD+PP] Authoritative bootstrap queue consumer for PP consensus: 此 PR 的原始提案和设计讨论，本 PR 直接基于其 review comments 实现。
- PR #30476 WIP: PP consensus for disaggregated prefill bootstrap: 早期探索性 PR，涉及相同概念，本 PR 最终实现方案。