PR #35886 完整报告

sgl-project/sglang

refactor(disagg): extract _all_reduce_polls helper

合并时间：2026-08-22 02:30

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35886

执行摘要

- 一句话：提取 _all_reduce_polls 统一三处 MIN all_reduce 逻辑
- 推荐动作：改动很小，但值得作为“安全重构分布式 collective 代码”的范例阅读：先点明不变量、明确不相关函数（poll_and_all_reduce_pp）的边界、用调用日志级等价性验证而非目测。建议关注两点：一是三类轮询路径的语义差异（staging 降级、metadata 门控、两段式同步）如何被保留；二是验证脚本未入库这一遗留改进点。

功能与动机

PR body 指出三个函数 poll_and_all_reduce、poll_and_all_reduce_attn_cp_tp_group、poll_and_all_reduce_with_staging 都以相同三行结尾——构建 uint8 CPU tensor、MIN all_reduce、返回 tolist。MIN 归约是真实不变量而非偶然：它阻止任何 rank 比 peers 更早提交传输状态。重复三遍“邀请漂移”，且事实上已出现命名漂移（第三个调用点用 poll_tensor，另两处用 tensor_to_reduce）。属于作者发起的 disaggregation 清理系列（#35838/#35843/#35844/#35847）之一。

实现拆解

1. 新增 _all_reduce_polls(polls, group) 帮助函数：封装 uint8 CPU tensor 构建、ReduceOp.MIN 的 all_reduce 与 tolist() 返回，并用 docstring 记录 MIN 归约的语义不变量；位置紧邻 _apply_metadata_gate，便于集中阅读门控与同步逻辑。
2. poll_and_all_reduce 尾部改为 return _all_reduce_polls(polls, gloo_group)，失败注入与 metadata 门控前置逻辑原样保留。
3. poll_and_all_reduce_attn_cp_tp_group 第二段同步改为 _all_reduce_polls(polls, attn_cp_cpu_group)，保持先 attn-tp 后 attn-cp 的两段式顺序不变。
4. poll_and_all_reduce_with_staging 尾部改为 _all_reduce_polls(raw_polls, gloo_group)，staging 推进、失败降级、metadata 门控均不动，顺带消除 poll_tensor / tensor_to_reduce 命名漂移。
5. 明确不碰 poll_and_all_reduce_pp：它把 PP consensus 映射到 poll 状态，从不执行 all_reduce，不属于该抽象。
6. 测试与验证：未新增测试文件；等价性用 fake torch/dist 在 30 个场景（5 个 poll 向量 x 3 个函数 x 4 种 staging 状态）中对比返回值与完整 all_reduce 调用日志（组、输入向量、op）；CI 通过 /rerun-group disaggregation 在 4-gpu-gb300、2-gpu-h100、

8-gpu-h20 共 11 个分布式测试全部通过。

关键文件：

- python/sglang/srt/disaggregation/utils.py（模块 PD 分离；类别 source；类型 core-logic；符号 _all_reduce_polls）：disaggregation 模块唯一改动文件：提取 _all_reduce_polls 并将三个轮询函数统一路由到该帮助函数，是本次重构的核心载体。

关键符号：_all_reduce_polls, poll_and_all_reduce, poll_and_all_reduce_attn_cp_tp_group, poll_and_all_reduce_with_staging

关键源码片段

python/sglang/srt/disaggregation/utils.py

disaggregation 模块唯一改动文件：提取 _all_reduce_polls 并将三个轮询函数统一路由到该帮助函数，是本次重构的核心载体。

```
def _all_reduce_polls(polls: List[int], group: dist.ProcessGroup) -> List[int]:
    """MIN-reduce poll states so no rank commits ahead of its peers.

    核心不变量：poll 状态经过 MIN 归约后，值为 0 的 KVPoll.Failed
    会从任一 rank 传播到全体，从而阻止任何 rank 比 peers 更早提交传输状态。
    三个调用点共用此逻辑，避免复制粘贴产生漂移。
    """
    tensor_to_reduce = torch.tensor(polls, dtype=torch.uint8, device="cpu")
    dist.all_reduce(tensor_to_reduce, op=dist.ReduceOp.MIN, group=group)
    return tensor_to_reduce.tolist()


def poll_and_all_reduce(
    pollers,
    gloo_group: dist.ProcessGroup,
    decode_reqs=None,
    metadata_buffers: Optional[MetadataBuffers] = None,
    server_args: Optional[ServerArgs] = None,
):
    """基础路径：失败注入 → metadata 门控 → MIN all_reduce。"""
    polls = _poll_with_failure_injection(pollers)
    # metadata 门控会把 Success 降级为 Transferring，同样参与 MIN 归约，
    # 确保 TP 各 rank 在 metadata 真正落地前不提交。
    if (
        decode_reqs is not None
        and metadata_buffers is not None
        and server_args is not None
    ):
        _apply_metadata_gate(polls, decode_reqs, metadata_buffers, server_args)
    return _all_reduce_polls(polls, gloo_group)


def poll_and_all_reduce_attn_cp_tp_group(
```

```
    pollers,
    attn_cp_cpu_group: dist.ProcessGroup,
    attn_tp_cpu_group: dist.ProcessGroup,
):
    """两段式同步：先 attn-tp 后 attn-cp，顺序对全局收敛语义至关重要。"""
    # 第一段：同一 (dp, cp) 分片的 TP 参与方先观察一致的状态转移。
    polls = poll_and_all_reduce(pollers, attn_tp_cpu_group)
    # 第二段：再跨 CP 同步，让单个 DP 分片内所有 TPxCP 参与方收敛到同一状态。
    return _all_reduce_polls(polls, attn_cp_cpu_group)
```

评论区精华

该 PR 没有 review 评论，仅有的两条 comments 是 CI 重跑交互。作者在 PR body 中阐述了核心验证思想：MIN 归约是不变量而非偶然；等价性对比必须看 complete all_reduce 调用日志（组、输入向量、op）而非仅返回值，因为"对同一个组归约两次"或"交换顺序"在返回值对比中不可见；fake all_reduce 实现了真实逐元素 min，使 MIN 语义被实际执行而非假设。

- 分布式 CI 验证与合入 (other): 分布式测试全绿后合入 main；无 review 层面的技术争论。

风险与影响

- 风险：技术风险：disaggregation 的 poll/all_reduce 是跨 rank collective，语义错误表现为 hang 而非错误答案，影响面是整个 PD 集群；但本次调用序列、组、op 完全一致，风险极低。回归防护风险：30 场景等价性验证脚本未入库，无自动化回归测试，未来类似重构需重新手工验证。兼容性风险：无 API、配置或数据契约变化，_all_reduce_polls 为模块内私有函数。
- 影响：对运行系统零功能影响：collective 数量、顺序、组、op 全部不变，仅多一次 Python 函数调用。对团队维护收益：dist.all_reduce 在 utils.py 中只出现一次，MIN 不变量单点陈述，命名漂移被消除。对并行开发：与 #35847 同文件不同区域，可任意顺序合并。
- 风险标记：核心路径变更，等价性验证未入库

关联脉络

- PR #35838 refactor(disagg): remove unreferenced dead code: 同系列 disaggregation 清理，删除 utils.py 中无调用者的死代码。
- PR #35843 refactor(disagg): remove dead build_and_send_encode_request: 同系列 per-file 清理，删除 encoder/receiver.py 中的无调用方法。
- PR #35844 refactor(disagg): remove dead get_embedding_port: 同系列 per-file 清理，删除 encoder/server.py 中 targeting 不存在路由的方法。
- PR #35847 refactor(disagg): collapse duplicated branches in get_kv_class: 同系列清理且同改 utils.py，本 PR 与之改动区域不重叠，可任意顺序合并。