PR #15771 完整报告

sgl-project/sglang

[6/N] (Elastic EP) Recover failed ranks

合并时间：2026-04-28 15:44

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/15771

执行摘要

- 一句话：实现 Elastic EP 失败进程恢复与动态重新加入
- 推荐动作：值得精读，特别是对分布式系统和容错机制感兴趣的开发者。该 PR 展示了如何在不阻塞在线服务的前提下恢复失败节点，设计上的权衡（如同步与异步、性能与可靠性）具有教育意义。但需注意当前实现依赖 Mooncake 库的内部语义，通用性受限；性能优化留给后续 PR。

功能与动机

作为 #11657 的后续，该 PR 使 SGLang 能够动态添加先前失败的进程，恢复最佳吞吐量。当节点失败时，系统管理员可以使用附加标志 --elastic-ep-rejoin 重新启动节点，而剩余的健康进程继续服务现有请求，并定期轮询重新启动进程的状态，一旦准备就绪即可无缝重新加入现有进程组，从而最小化服务中断。

实现拆解

1. 新增 CLI 参数与初始化断言：在 server_args.py 添加 --elastic-ep-rejoin 标志，在 _handle_elastic_ep 中增加 pp_size == 1 断言，确保弹性 EP 当前仅支持单管道阶段。
2. 分布式初始化适配：在 parallel_state.py 的 GroupCoordinator、init_world_group、init_model_parallel_group、init_distributed_environment 中新增 recovered_rank 参数，使进程组创建时传递 Mooncake 恢复选项；同时跳过恢复进程的消息队列初始化，改为在恢复完成后重建。
3. 恢复辅助函数：在 elastic_ep.py 添加核心恢复逻辑，包括 _wait_for_peer_state（轮询 Mooncake 后端确认对等就绪）、_iter_live_parallel_groups（按 unique_name 排序保证集体操作顺序一致）、_map_global_to_group_local_ranks、_maybe_create_message_queue、_refresh_ep_members、try_recover_ranks（对每个活组执行 recover_ranks）以及 join_process_groups（供重新加入进程调用）。
4. 模型运行器集成：在 ModelRunner.__init__ 中，检测 elastic_ep_rejoin 后执行 join_process_groups、广播专家位置元数据、重置 ElasticEPState；新增 maybe_recover_ep_ranks 方法在每个前向迭代中检查活跃秩状态，发现失败时调用 try_recover_ranks 并同步 seed 与 expert location。
5. 专家位置元数据同步：在 expert_location.py 增加 broadcast_global_expert_location_metadata 函数，通过分布式广播将健康源的设备张量和 CPU 副本同步给所有秩，确保恢复后

expert routing 一致。

6. 数据并行控制器调整：_broadcast_ports_as_server 在弹性 EP 启用时不再关闭 ZMQ 套接字，而是启动守护线程 _reply_ports_as_server 持续响应恢复秩的端口请求。
7. EPLB 管理器重置：在 eplb_manager.py 增加 reset_generator 方法，使恢复后调度从一致状态恢复。
8. 文档更新：在 docs/advanced_features/server_arguments.md 记录新参数。

关键文件：

- python/sglang/srt/elastic_ep/elastic_ep.py（模块 弹性 EP；类别 source；类型 core-logic；符号 reset, _get_process_group_backend, _iter_live_parallel_groups, _map_global_to_group_local_ranks）：核心实现文件，新增了恢复辅助函数和状态管理方法，是整个 PR 的基石。
- python/sglang/srt/model_executor/model_runner.py（模块 模型执行器；类别 source；类型 data-contract；符号 maybe_recover_ep_ranks, _get_healthy_expert_location_src_rank）：在模型初始化中集成 rejoin 路径，并在每个前向迭代中检查回复发恢复。
- python/sglang/srt/eplb/expert_location.py（模块 专家路由；类别 source；类型 core-logic；符号 broadcast_global_expert_location_metadata）：新增 broadcast_global_expert_location_metadata 函数，确保恢复后专家位置元数据一致。
- python/sglang/srt/managers/data_parallel_controller.py（模块 数据并行控制；类别 source；类型 entrypoint；符号 _reply_ports_as_server）：修改端口广播逻辑，为恢复进程提供持久化的 REP 服务线程。
- python/sglang/srt/distributed/parallel_state.py（模块 分布式状态；类别 source；类型 dependency-wiring；符号 init_world_group, init_model_parallel_group, init_distributed_environment, GroupCoordinator.init）：分布式初始化传递 recovered_rank 标志，控制进程组创建和消息队列初始化。
- python/sglang/srt/server_args.py（模块 服务参数；类别 source；类型 core-logic；符号 elastic_ep_rejoin）：添加新参数和 pp_size==1 断言。
- python/sglang/srt/eplb/eplb_manager.py（模块 EPLB 管理器；类别 source；类型 core-logic；符号 reset_generator）：新增 reset_generator 方法，支持恢复后重置 EPLB 状态。
- docs/advanced_features/server_arguments.md（模块 文档；类别 docs；类型 documentation）：记录新参数 --elastic-ep-rejoin 的文档说明。

关键符号：reset, _get_process_group_backend, _iter_live_parallel_groups, _map_global_to_group_local_ranks, _wait_for_peer_state, _maybe_create_message_queue, _refresh_ep_members, try_recover_ranks, join_process_groups, maybe_recover_ep_ranks, _get_healthy_expert_location_src_rank, broadcast_global_expert_location_metadata, _reply_ports_as_server, reset_generator

关键源码片段

python/sglang/srt/model_executor/model_runner.py

在模型初始化中集成 rejoin 路径，并在每个前向迭代中检查回复发恢复。

```
def maybe_recover_ep_ranks(self):
    # TODO(perf): `active_ranks.all()` 在 CUDA 张量上会触发主机 - 设备同步，
    # 且此函数在前向路径上。此检查仅在启用 --elastic-ep-backend 时运行，
    # 因此同步开销不会传播到其他配置。留给未来弹性 EP 路径优化。
    if self.tp_group.active_ranks.all() and self.tp_group.active_ranks_cpu.all():
        # 所有秩健康，无需恢复
        return

    # 将设备张量拷贝到 CPU 并转换为 numpy 数组用于掩码计算
    tp_active_ranks = self.tp_group.active_ranks.detach().cpu().numpy()
    tp_active_ranks_cpu = self.tp_group.active_ranks_cpu.detach().numpy()
    # 取两者交集：仅当两个视图都标记为不活跃时才需要恢复
    tp_active_ranks &= tp_active_ranks_cpu

    # 注意：ranks_to_recover 使用 tp_group 内索引。当前 Mooncake 弹性 EP
    # 实现假设 --pp-size=1，因此 tp-group 索引等于全局秩索引。
    ranks_to_recover = [
        i for i in range(len(tp_active_ranks)) if not tp_active_ranks[i]
    ]
    if not ranks_to_recover:
        return

    # 调用恢复辅助函数，该函数会在每个并行组上执行集体操作
    try_recover_ranks(ranks_to_recover)

    # 选择一个健康的秩作为元数据广播源
    healthy_local_indices = [
        i for i in range(len(tp_active_ranks)) if tp_active_ranks[i]
    ]
    src_rank = self.tp_group.ranks[healthy_local_indices[0]]
    # 广播专家位置元数据，确保所有秩一致
    broadcast_global_expert_location_metadata(src_rank=src_rank)
    # 重置状态管理器，使活跃秩重新标记为全 1
    ElasticEPStateManager.instance().reset()
```

python/sglang/srt/eplb/expert_location.py

新增 broadcast_global_expert_location_metadata 函数，确保恢复后专家位置元数据一致。

```
def broadcast_global_expert_location_metadata(
    src_rank: int = 0, group: Optional[torch.distributed.ProcessGroup] = None
):
    """广播全局 ExpertLocationMetadata 从 src_rank 到所有秩。

    用于 Elastic EP 秩恢复后确保所有秩（包括新恢复的）共享相同的专家位置元数据。
    """
    metadata = get_global_expert_location_metadata()
    assert metadata is not None, "ExpertLocationMetadata not initialized"
```

```
    # 确保设备张量是连续的，以便就地广播
    metadata.physical_to_logical_map = metadata.physical_to_logical_map.contiguous()
    metadata.logical_to_all_physical_map = (
        metadata.logical_to_all_physical_map.contiguous()
    )
    metadata.logical_to_all_physical_map_num_valid = (
        metadata.logical_to_all_physical_map_num_valid.contiguous()
    )
    if metadata.logical_to_rank_dispatch_physical_map is not None:
        metadata.logical_to_rank_dispatch_physical_map = (
            metadata.logical_to_rank_dispatch_physical_map.contiguous()
        )

    device_tensors = [
        metadata.physical_to_logical_map,
        metadata.logical_to_all_physical_map,
        metadata.logical_to_all_physical_map_num_valid,
    ]
    if metadata.logical_to_rank_dispatch_physical_map is not None:
        device_tensors.append(metadata.logical_to_rank_dispatch_physical_map)

    # 逐个张量广播，利用 PyTorch 分布式原语
    for tensor in device_tensors:
        torch.distributed.broadcast(tensor, src=src_rank, group=group)

    # 广播完成后更新 CPU 副本，避免后续 GPU -> CPU 同步
    metadata.physical_to_logical_map_cpu = metadata.physical_to_logical_map.cpu()
    metadata.logical_to_all_physical_map_cpu = (
        metadata.logical_to_all_physical_map.cpu()
    )
```

评论区精华

- 同步与死锁风险（critical）：gemini-code-assist 指出 try_recover_ranks 基于本地轮询可能导致不同秩决策不一致从而死锁。作者回应 Mooncake 库内部保证一致性，并添加注释说明。
- 组迭代顺序（high）：字典迭代无序导致集体操作死锁。作者在 _iter_live_parallel_groups 中按 unique_name 排序修复。
- 性能开销（high）：前向路径中 active_ranks.all() 触发 GPU 同步。作者承认并标记为 TODO，但认为仅影响弹性 EP 路径且可接受。
- 源秩选择（high）：广播专家元数据时默认 rank 0 可能正在恢复。作者修复为动态选择健康秩作为源。
- 全局秩计算（critical）：maybe_recover_ep_ranks 中使用了错误的局部秩索引。作者改为 get_world_group().rank。
- ZMQ 线程安全（medium）：后台线程无关闭机制，且 decode 可能抛出异常。作者添加 try-except 并标记生命周期待后续。

- 消息队列资源泄漏（medium）：_maybe_create_message_queue 未关闭旧广播器。作者标记为 false positive。
- 随机种子广播（high）：恢复秩需要参与种子广播。作者说明种子广播在 ModelRunner 初始化之后进行，当前路径无需额外处理。
  - 恢复决策同步与死锁风险 (correctness): 作者回应 Mooncake 库内部保证一致性，并添加注释说明。
  - 组迭代顺序一致性 (correctness): 作者在 _iter_live_parallel_groups 中按 unique_name 排序修复。
  - 前向路径 CUDA 同步性能 (performance): 作者承认并标记为 TODO，但认为仅影响弹性 EP 路径且可接受。
  - 专家元数据广播源选择 (correctness): 作者修复为动态选择健康秩作为源。
  - 全局秩计算错误 (correctness): 作者改为 get_world_group().rank。
  - ZMQ 线程生命周期 (security): 作者添加 try-except 并标记生命周期待后续。
  - 消息队列资源泄漏 (performance): 作者标记为 false positive。
  - 随机种子广播缺失 (correctness): 作者说明种子广播在 ModelRunner 初始化之后进行，当前路径无需额外处理。

风险与影响

- 风险：
  1. 性能回归：前向路径中的 CUDA 同步（active_ranks.all()）可能增加延迟，但仅限于启用弹性 EP 的配置。
  2. 分布式死锁：如果 Mooncake 库的同步保证有 bug 或版本不兼容，可能导致集群死锁。
  3. ZMQ 线程泄漏：_reply_ports_as_server 线程在进程退出时不会优雅关闭，可能导致端口占用。
  4. 元数据不一致：广播源秩若尚未完全恢复可能广播陈旧数据；当前实现动态选择健康秩，但 --elastic-ep-rejoin 标志在滚动重启场景下可能失效。
  5. 缺少测试覆盖：无自动化测试，仅靠手动多节点验证。 - 影响：对用户：启用弹性 EP 的部署可获得节点级容错，系统管理员可在不停止服务的情况下替换节点。对系统：引入每前向迭代的健康检查（微小开销），但仅在 elastic EP 模式生效。对团队：此 PR 是系列的第 6 个，表明弹性 EP 功能趋于成熟，后续可在此基础上优化性能。 - 风险标记：CUDA 同步开销，缺少自动化测试，ZMQ 线程泄漏，Mooncake 版本依赖

关联脉络

- PR #11657 [Elastic EP] ...: 此 PR 的基础，引入了弹性 EP 框架和 Mooncake 后端支持。