PR #31744 完整报告

sgl-project/sglang

[Elastic EP] Fix recovery lifecycle and add manual coverage

合并时间：2026-07-25 14:32

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/31744

执行摘要

- 一句话：修复 Elastic EP 恢复生命周期并添加手动 4+4 恢复测试
- 推荐动作：建议仔细阅读此 PR，特别是 post_capture_elastic_ep_recover 方法的引入和生命周期顺序的调整。新增的手动测试是理解 Elastic EP 恢复流程的良好起点。对于使用 Elastic EP 的团队，应确保 CI 中包含类似测试。

功能与动机

A previous initialization lifecycle refactor moved recover-joiner work away from the ordering required by Elastic EP recovery. The recovery path must complete local memory-pool setup, attention-backend initialization, warmup, and CUDA graph capture before joining the recovered process groups. This PR restores that ordering and adds a manual 4+4 recovery test that exercises the full path.

实现拆解

1. 分离 recovery 与 scale-join 逻辑：在 model_runner.py 的 _initialize_elastic_ep_joiner 方法中，将条件从 is_ep_joiner 改为 is_ep_scale_joiner，使 recovery joiner 跳过该路径，不再在此处执行 join_process_groups。同时移除了 recovery 路径的 get_healthy_expert_location_src_rank 和 ElasticEPStateManager.reset() 调用，避免重复执行。
2. 新增 post_capture_elastic_ep_recover 方法（在 model_runner.py），负责在 CUDA 图捕获完成后执行 join_process_groups、广播 expert 位置元数据、初始化 ExpertDistributionRecorder 并重置状态。在 scheduler.py 的 init_model_worker 中，仅在 --elastic-ep-backend 启用且 ep_join_mode == "recover" 时调用此方法，确保恢复发生在正确的生命周期点。
3. 参数统一：将 load_model_utils.py 和 server_args.py 中的 is_ep_scale_joiner 替换为 is_ep_joiner，使 recovery joiner 也能通过模型加载后的 barrier 以及正确分配端口。
4. 清理恢复路径：在 elastic_ep.py 的 maybe_recover_ep_ranks 中移除 broadcast_pyobj 种子广播（经 review 确认是空操作），并调整函数签名以接收 model_config 和 moe_ep_rank，以便在恢复时进行 expert 位置元数据广播。
5. 添加手动 E2E 测试：新增 test/manual/ep/test_elastic_recover.py，需要 8 GPU，启动主节点（4 rank）和初始 joiner（4 rank），杀死初始 joiner 后启动 recovery joiner，验

证生成正常。覆盖完整恢复生命周期，CUDA graph capture 保持启用。

关键文件：

- test/manual/ep/test_elastic_recover.py（模块 弹性 EP；类别 test；类型 test-coverage；符号 _visible_device_ids, _server_args, TestElasticRecover4To4, setUpClass）：新增手动 E2E 恢复测试，覆盖完整的 4+4 故障恢复流程，是验证此 PR 正确性的关键测试。
- python/sglang/srt/model_executor/model_runner.py（模块 模型运行器；类别 source；类型 core-logic；符号 post_capture_elastic_ep_recover, _initialize_elastic_ep_joiner）：核心变更文件，新增 post_capture_elastic_ep_recover 方法并调整 _initialize_elastic_ep_joiner 以分离 recovery 和 scale-join 逻辑。
- python/sglang/srt/elastic_ep/elastic_ep.py（模块 弹性 EP 库；类别 source；类型 dependency-wiring；符号 maybe_recover_ep_ranks）：清理恢复路径中的无效 seed 广播，并调整函数签名以支持 expert 位置元数据广播。
- python/sglang/srt/model_executor/model_runner_components/load_model_utils.py（模块 加载工具；类别 source；类型 data-contract；符号 dist_barrier_after_load）：参数名统一，确保 recovery joiner 也能通过模型加载后的 barrier。
- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 core-logic）：在正确生命周期点触发恢复加入流程。
- python/sglang/srt/server_args.py（模块 参数配置；类别 source；类型 core-logic）：统一参数名，使恢复 joiner 能正确分配端口。
- python/sglang/srt/managers/tp_worker.py（模块 TP worker；类别 source；类型 core-logic）：伴随性调整。

关键符号：post_capture_elastic_ep_recover, maybe_recover_ep_ranks, dist_barrier_after_load, _initialize_elastic_ep_joiner, _visible_device_ids, _server_args, setUpClass, _launch, _generate, _generate_ok

关键源码片段

test/manual/ep/test_elastic_recover.py

新增手动 E2E 恢复测试，覆盖完整的 4+4 故障恢复流程，是验证此 PR 正确性的关键测试。

```
@classmethod
def setUpClass(cls):
    """启动主节点和初始 joiner，等待服务就绪。"""
    cls.base_url = f"http://127.0.0.1:{PRIMARY_PORT}"
    visible_devices = _visible_device_ids()
    # 前 4 个 GPU 启动主节点（node_rank=0）
    cls.primary = cls._launch(
        node_rank=0, port=PRIMARY_PORT,
        visible_devices=visible_devices[:LOCAL_EP_SIZE],
        name="primary",
    )
    # 后 4 个 GPU 启动初始 joiner（node_rank=1）
    cls.initial_joiner = cls._launch(
```

```
            node_rank=1, port=JOINER_PORT,
            visible_devices=visible_devices[LOCAL_EP_SIZE:EP_SIZE],
            name="initial_joiner",
        )
        # 等待主节点 HTTP 服务就绪
        wait_for_http_ready(
            f"{cls.base_url}/health_generate",
            timeout=DEFAULT_TIMEOUT_FOR_SERVER_LAUNCH,
            process=cls.primary,
        )

    @classmethod
    def _launch(cls, *, node_rank, port, visible_devices, name, recover=False):
        """启动 SGLang server 进程。"""
        log_path = Path(f"/tmp/elastic_ep_recover_{name}_{int(time.time())}.log")
        log_file = open(log_path, "w")
        env = os.environ.copy()
        env["CUDA_VISIBLE_DEVICES"] = ",".join(visible_devices)
        cmd = _server_args(node_rank, port, recover)
        process = subprocess.Popen(cmd, env=env, stdout=log_file, stderr=log_file)
        cls.processes.append(process)
        cls.log_files.append(log_file)
        cls.log_paths[name] = log_path
        return process
```

python/sglang/srt/model_executor/model_runner.py

核心变更文件，新增 post_capture_elastic_ep_recover 方法并调整 _initialize_elastic_ep_joiner 以分离 recovery 和 scale-join 逻辑。

```
    def post_capture_elastic_ep_recover(self):
        """在 CUDA graph capture 后执行恢复加入流程。"""
        # 1. 加入恢复的进程组（Mooncake 后端会同步状态）
        join_process_groups()
        global_ep_rank = self.ps.tp_rank + self.server_args.ep_join_rank_offset
        # 2. 广播 expert 位置元数据
        broadcast_global_expert_location_metadata(
            model_config=self.model_config,
            moe_ep_rank=global_ep_rank,
            src_rank=get_healthy_expert_location_src_rank(
                invoked_in_elastic_ep_rejoin_path=True # 标记自己为 recovery rank
            ),
        )
        # 3. 初始化 expert 分布记录器
        set_global_expert_distribution_recorder(
            ExpertDistributionRecorder.init_new(
                self.server_args,
                get_global_expert_location_metadata(),
                rank=global_ep_rank,
            )
```

```
)
# 4. 重置弹性 EP 状态管理器
ElasticEPStateManager.instance().reset()
```

评论区精华

1. zackyoray 指出恢复路径中 broadcast_pyobj 种子广播返回值被忽略，实际是空操作，且测试使用固定种子掩盖了差异。UNIDY2002 随后移除了该广播调用。
2. ShangmingCai 要求检查 dist_barrier_after_load 是否也应切换为 is_ep_joiner 以覆盖 recovery joiner，UNIDY2002 确认并修复。

- 恢复路径中 seed broadcast 实际是空操作 (correctness): UNIDY2002 承认并移除了该 broadcast_pyobj 调用。
- dist_barrier_after_load 是否应使用 is_ep_joiner (correctness): UNIDY2002 确认并修复，将参数名从 is_ep_scale_joiner 改为 is_ep_joiner。

风险与影响

- 风险：
  1. 恢复路径变更可能引入回归，但新增的 E2E 手动测试覆盖了 4+4 恢复场景，且 zackyoray 手动验证了 scale-up 路径正常。
  2. 参数统一为 is_ep_joiner 可能影响 scale-join 流程，但 scale-join 时 is_ep_joiner 也为真，行为一致。
  3. 移除 broadcast_pyobj 可能破坏随机种子同步，但实际 recovery 路径中种子已通过命令行参数一致传递，且该调用本就是空操作。整体风险较低。 - 影响：影响范围限于启用 Elastic EP（后端为 Mooncake）且使用恢复模式的部署。稳态推理路径无变化，scale-join 路径无变化。团队需要运行新增的手动测试验证恢复功能。对于其他配置，影响为无。 - 风险标记：依赖 Mooncake 后端，需要手动测试验证，生命周期敏感性

关联脉络

- 暂无明显关联 PR