PR #30553 完整报告

sgl-project/sglang

[2/N] elastic-ep: Enable EPLB after scale-up

合并时间：2026-07-30 01:06

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/30553

执行摘要

- 一句话：在弹性 EP 扩容后重新启用 EPLB 专家负载均衡
- 推荐动作：建议阅读 eplb_manager.py 中 _compute_expert_location_metadata 的广播逻辑，以及 model_runner.py 中 _rearm_eplb_after_elastic_scale 和 _reset_eplb_after_elastic_scale_failure 两个方法。这些实现展示了在动态拓扑中如何安全地重新激活周期性负载均衡，是弹性 EP 功能的关键拼图。

功能与动机

在 #30164 实现的运行时弹性 EP 扩容路径中，扩容后 EPLB 被永久禁用，导致拓扑变化后无法恢复周期性专家重排，影响长期负载均衡效果。本 PR 旨在扩容完成后重新启用 EPLB。

实现拆解

1. 扩容请求时禁用 EPLB：在 scheduler.py 的 handle_scale_elastic_ep 中，当收到扩容请求时调用 eplb_manager.disable_rebalance()，防止在扩容过程中触发专家重排。
2. 扩容最终化时重新启用：在 model_runner.py 的 _initialize_elastic_ep_joiner 和 _finalize_scale_up 中，在设置 phase=serving_expanded 后调用 _rearm_eplb_after_elastic_scale()，该方法检查记录器状态并调用 eplb_manager.enable_rebalance()。
3. PLB 循环适配后缩放：在 eplb_manager.py 的 rebalance() 中增加后缩放阶段判断：若 has_scaled 且 phase 为 serving_expanded 或 failed 时才执行重排，否则直接返回。新增 _compute_expert_location_metadata() 方法，在后缩放时由 rank 0 计算映射并广播给所有 rank，且使用全局 rank 而非本地 tp_rank 进行 P2P 移动。
4. 扁平拓扑支持：ExpertLocationMetadata.init_by_eplb 新增 use_flat_topology 参数，当后缩放时置为 True，将 nnodes 强制设为 1，避免多节点拓扑扭曲映射。ExpertLocationUpdater.update 也同步增加该参数并传递到内部。
5. 扩容失败恢复：在 model_runner.py 的 maybe_join_ep_ranks 中，如果超时或 cohort 不匹配导致失败，调用 _reset_eplb_after_elastic_scale_failure()，重置分布记录器并重新启用 EPLB，使系统可在上一次已提交的世界上继续运行。
6. 测试增强：在 test/manual/ep/test_elastic_scale.py 中，在 test_scale_up_on_demand 测试用例末尾增加 _generate_logprob_ok("after post-scale workload")，验证重排后 token 生成和 logprobs 正确性。

关键文件：

- python/sglang/srt/eplb/eplb_manager.py（模块 负载均衡；类别 source；类型 core-logic；符号 enable_rebalance, _compute_expert_location_metadata, _elastic_global_rank）：核心修改：新增 enable_rebalance 方法；在 rebalance 中增加后缩放阶段检查；新增 _compute_expert_location_metadata 处理广播映射。
- python/sglang/srt/model_executor/model_runner.py（模块 模型执行器；类别 source；类型 data-contract；符号 _rearm_eplb_after_elastic_scale, _reset_eplb_after_elastic_scale_failure）：新增 _rearm_eplb_after_elastic_scale 和 _reset_eplb_after_elastic_scale_failure 两个方法，分别处理扩容成功和失败后的 EPLB 恢复。
- python/sglang/srt/eplb/expert_location.py（模块 专家位置；类别 source；类型 core-logic）：init_by_eplb 新增 use_flat_topology 参数，用于弹性后缩放时忽略多节点拓扑。
- python/sglang/srt/eplb/expert_location_updater.py（模块 专家更新；类别 source；类型 core-logic）：update 方法增加 use_flat_topology 参数，传递给内部拓扑计算。
- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 core-logic）：handle_scale_elastic_ep 中增加在扩容待定时禁用 EPLB 的步骤。
- test/manual/ep/test_elastic_scale.py（模块 弹性测试；类别 test；类型 test-coverage）：新增后缩放 logprob 验证断言，确保重排后 token 生成和 logprobs 正确。

关键符号：EPLBManager.enable_rebalance, EPLBManager._compute_expert_location_metadata, ModelRunner._rearm_eplb_after_elastic_scale, ModelRunner._reset_eplb_after_elastic_scale_failure, ExpertLocationMetadata.init_by_eplb (added use_flat_topology), ExpertLocationUpdater.update (added use_flat_topology), Scheduler.handle_scale_elastic_ep (added disable_rebalance)

关键源码片段

python/sglang/srt/eplb/eplb_manager.py

核心修改：新增 enable_rebalance 方法；在 rebalance 中增加后缩放阶段检查；新增 _compute_expert_location_metadata 处理广播映射。

```
# EPLBManager 新增 enable_rebalance 方法，用于扩容完成后重新激活定期重排
def enable_rebalance(self):
    self._rebalance_disabled_reason = None
    self._rebalance_disabled_logged = False
    self.reset_generator() # 重新创建生成器，确保下次迭代进入重排

# rebalance 方法头部新增后缩放阶段检查
def rebalance(self):
    if self._rebalance_disabled_reason is not None:
        ...
        return
```

```
    elastic_state = ElasticEPStateManager.instance()
    is_post_scale_rebalance = elastic_state is not None and elastic_state.has_scaled
    # 如果 scale 仍在进行中（pending 或未达到 serving_expanded/failed），跳过本次重排
    if is_post_scale_rebalance and (
        elastic_state.pending_ep_size is not None
        or elastic_state.scale_phase not in ("serving_expanded", "failed")
    ):
        return

    # 后续正常的 rebalance 逻辑 ...
    expert_location_metadata = self._compute_expert_location_metadata(
        logical_count,
        broadcast_over_world=is_post_scale_rebalance, # 后缩放时要求广播
    )
```

python/sglang/srt/model_executor/model_runner.py

新增 _rearm_eplb_after_elastic_scale 和 _reset_eplb_after_elastic_scale_failure 两个方法，分别处理扩容成功和失败后的 EPLB 恢复。

```
  # 扩容成功后重新启用 EPLB
  def _rearm_eplb_after_elastic_scale(self) -> None:
      if self.eplb_manager is None:
          return
      recorder = get_global_expert_distribution_recorder()
      if not recorder.recording:
          recorder.start_record() # 确保记录器正在工作
      self.eplb_manager.enable_rebalance() # 清除禁用原因，重置生成器

  # 扩容失败后恢复 EPLB（重置记录器并重新启用）
  def _reset_eplb_after_elastic_scale_failure(self) -> None:
      if self.eplb_manager is None:
          return
      # 基于当前 expert 位置重新初始化记录器，避免数据污染
      set_global_expert_distribution_recorder(
          ExpertDistributionRecorder.init_new(
              self.server_args,
              get_global_expert_location_metadata(),
              rank=self._elastic_global_rank(),
          )
      )
      self._rearm_eplb_after_elastic_scale() # 重新启用 EPLB
```

评论区精华

唯二的 review 评论是 ShangmingCai 在 eplb_manager.py 第 193 行提出的优化建议："nit: maybe deduplicate with broadcast_global_expert_location_metadata"，建议将新加的 _compute_expert_location_metadata 中的广播逻辑与已有的 broadcast_global_expert_location_metadata 函数去重。该建议未被合并前的代码采纳，当

前实现保留了独立广播逻辑，可能因为缩放广播与初始化广播的 world 和 rank 映射存在差异。其他 reviewer 均表示 LGTM。

- 广播 expert 映射时的去重建议 (design): 作者未直接回复，当前实现保留独立广播逻辑（可能由于缩放广播与初始化广播的 world 和 rank 映射不同）。评论被标记为 'nit'，无强制修改要求。

风险与影响

- 风险：
  1. 竞态条件：rebalance() 中对 elastic_state.scale_phase 的检查存在时间窗口，如果并发缩放与重排交错，可能导致重排使用不完整的拓扑。当前通过 pending_ep_size 和 phase 双重检查降低风险，但未加锁。
  2. 广播一致性：_compute_expert_location_metadata 中 rank 0 计算后广播，若广播期间其他 rank 执行了其他操作，可能导致不一致。当前依赖 scale_phase 同步，但应确保所有 rank 在广播前处于 barrier 状态（已在 _elastic_scale_ready_barrier 中做了一次全局同步）。
  3. 失败恢复路径：_reset_eplb_after_elastic_scale_failure 会重置 ExpertDistributionRecorder，若此时已有部分重排进行中，可能导致记录器丢失部分数据。
  4. Mooncake 依赖性：依赖 Mooncake commit 9843b897（PR #2623）修复 P2P 阻塞，若未升级至该版本，在多 rank 扩容时存在死锁风险。
  5. 扁平拓扑潜在问题：use_flat_topology=True 将 nnodes 设为 1，可能导致跨节点专家放置策略失效，但当前弹性扩容设计为单节点，暂时可接受。 - 影响：影响范围：仅影响启用了 --elastic-ep-size 的弹性 EP 部署。非弹性 EP 路径完全不受影响（EPLB 逻辑原有行为不变）。影响程度：中。用户将自动获得扩容后的负载均衡恢复能力，无需手动干预；扩容失败后系统能自动恢复到上次可用拓扑，提升健壮性。系统影响：新增约 100 行核心逻辑，增加少量广播和同步开销，但仅在扩容完成后一次，不影响推理时延。
- 风险标记：核心路径变更，分布式同步，竞态条件，依赖 Mooncake 补丁

关联脉络

- PR #30164 [1/N] elastic-ep: Add runtime EP scale-up: 本 PR 是同一弹性 EP 系列的第二部分，依赖 #30164 提供的运行时扩容基础（包括新增的 ElasticEPStateManager 和 scale API）。
- PR #2623 [PG] Fix elastic P2P after max_world_size recovery: 本 PR 的 E2E 测试依赖 Mooncake 的修复（commit 9843b897），否则 P2P 通信在扩容后可能死锁。