PR #32962 完整报告

sgl-project/sglang

Fix silently wrong EPLB output with --moe-a2a-backend none (rank-invariant dispatch)

合并时间：2026-07-31 13:10

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/32962

执行摘要

- 一句话：修复 no-a2a 路径 EPLB 冗余专家静默错误
- 推荐动作：值得精读。该 PR 不仅修复了一个隐蔽正确性问题，还展示了如何通过纯函数索引替代随机数来保证跨 rank 一致性的设计思路。对关注 MoE 调度、专家并行或分布式推理一致性的工程师尤其有参考价值。

功能与动机

PR body 指出：EPLB runs without complaint on the no-a2a MoE path but silently corrupts the output as soon as it has redundant experts to place。在 lmsys/sglang-ci-dsv3-test 上，GSM8K 准确率从约 0.63 跌至 0.42，无任何报错或警告。根因有二：_compute_logical_to_all_physical_map 将候选物理专家折叠到最近 rank（per-rank 构造），且 static 算法从 per-rank 表中选择副本，导致复制专家被多次求和。冗余专家数为 0 时恰好正确，因此从未被发现。

实现拆解

1. 引入 rank_invariant 概念（python/sglang/srt/eplb/expert_location_dispatch.py）：ExpertLocationDispatchInfo 新增 rank_invariant 字段，init_new 依据 server_args.moe_a2a_backend == "none" 设置，作为后续调度决策的前提。
2. 改造 dynamic 算法：_topk_ids_logical_to_physical_dynamic 在 rank_invariant 模式下改用 token 行号（row_index）选择物理副本，替代 torch.randint。原因：随机数依赖默认 CUDA 生成器的 philox 偏移，跨 rank 对齐无保证且破坏可复现性；行号选择是纯函数，无需通信即可一致，且连续行交替副本，与 EPLB 放置求解器的均匀分割假设吻合。
3. 调整候选映射（python/sglang/srt/eplb/expert_location.py）：_compute_logical_to_all_physical_map 在 moe_a2a_backend == "none" 时跳过 per-rank 的最近专家折叠，保留完整副本列表，使冗余副本在初始放置时即可被动态调度访问。
4. 参数默认与校验（python/sglang/srt/server_args.py）：_handle_eplb_and_dispatch 根据 needs_rank_invariant_dispatch 将默认算法设为 dynamic（原为 static），并对 static/lp 抛错以防静默错误；fake 因共享 dynamic 实现而允许。
5. 测试与文档配套：新增 test/registered/ep/test_eplb_no_a2a.py（nightly 2-GPU H200，含无重平衡与 DP attention 两种变体），更新 test_compute_logical_to_rank_dispatch_p

hysical_map.py 的 stub 以支持 moe_a2a_backend 参数，并同步更新文档。

关键文件：

- python/sglang/srt/eplb/expert_location_dispatch.py（模块 专家调度；类别 source；类型 core-logic；符号 ExpertLocationDispatchInfo, _topk_ids_logical_to_physical_dynamic, topk_ids_logical_to_physical）：核心调度逻辑改动：引入 rank_invariant 字段，并让 dynamic 算法在 rank-invariant 模式下改用 token 行号选择副本。
- python/sglang/srt/server_args.py（模块 启动参数；类别 source；类型 core-logic；符号 _handle_eplb_and_dispatch）：参数默认与校验：根据 moe_a2a_backend 决定默认算法，并拒绝不兼容的 static/lp。
- python/sglang/srt/eplb/expert_location.py（模块 专家分布；类别 source；类型 core-logic；符号 _compute_logical_to_all_physical_map）：跳过 per-rank 候选折叠，使冗余副本在无 a2a 后端时可被动态调度访问。
- test/registered/ep/test_eplb_no_a2a.py（模块 端到端测试；类别 test；类型 test-coverage；符号 TestEPLBNoA2A, test_gsm8k, TestEPLBNoA2ADPAttention）：新增 nightly e2e 测试，覆盖初始放置和 DP attention 两种场景，验证 GSM8K 准确率恢复。
- test/registered/unit/eplb/test_compute_logical_to_rank_dispatch_physical_map.py（模块 单元测试；类别 test；类型 test-coverage；符号 _make_server_args）：更新 stub 以支持 moe_a2a_backend 参数，确保单元测试继续覆盖 rank-local collapse。

关键符号：ExpertLocationDispatchInfo.init_new, _topk_ids_logical_to_physical_dynamic, _handle_eplb_and_dispatch, _compute_logical_to_all_physical_map, test_gsm8k

关键源码片段

python/sglang/srt/eplb/expert_location_dispatch.py

核心调度逻辑改动：引入 rank_invariant 字段，并让 dynamic 算法在 rank-invariant 模式下改用 token 行号选择副本。

```
@dataclass
class ExpertLocationDispatchInfo:
    # ... 已有字段 ...
    # 当 rank_invariant 为 True 时，所有 EP rank 必须为同一 token 选择同一个物理专家。
    # 这是无 a2a 后端时的硬性要求：所有 rank 对同一批 token 计算部分和再相加，
    # 若选择不一致，复制的逻辑专家会被多次计数，导致输出损坏。
    rank_invariant: bool = False

    @classmethod
    def init_new(cls, layer_id: int):
        server_args = get_server_args()
        # ...
        return cls(
            # ...
            rank_invariant=server_args.moe_a2a_backend == "none",
            # ...
```

```
        )

    def _topk_ids_logical_to_physical_dynamic(
        topk_ids: torch.Tensor, info: Optional[ExpertLocationDispatchInfo]
    ) -> torch.Tensor:
        """将每个 (token, logical expert) 均匀地分散到该逻辑专家的多个副本上。

        在 rank_invariant 模式下，按 token 行号选择副本而非随机：
        torch.randint 读取默认 CUDA 生成器，跨 rank 对齐无任何保证，且会破坏
        贪心请求的可复现性。行号索引是纯函数，各 rank 无需通信即可一致，且连续行
        交替选择不同副本，正好匹配 EPLB 放置求解器假设的均匀负载分配。
        """
        topk_ids_original_shape = topk_ids.shape
        original_dtype = topk_ids.dtype
        device = topk_ids.device
        topk_ids = topk_ids.flatten()

        num_valid = info.partial_logical_to_all_physical_map_num_valid[topk_ids]
        if info.rank_invariant:
            slots_per_token = (
                topk_ids_original_shape[-1] if len(topk_ids_original_shape) > 1 else 1
            )
            row_index = (
                torch.arange(topk_ids.shape[0], dtype=num_valid.dtype, device=device)
                // slots_per_token
            )
            chosen_dispatch_index = row_index % num_valid
        else:
            chosen_dispatch_index = (
                torch.randint(0, 65536, topk_ids.shape, dtype=torch.int32, device=device)
                % num_valid
            )
        topk_ids = info.partial_logical_to_all_physical_map[topk_ids, chosen_dispatch_index]
        if topk_ids.dtype != original_dtype:
            topk_ids = topk_ids.to(original_dtype)
        return topk_ids.view(topk_ids_original_shape)
```

python/sglang/srt/server_args.py

参数默认与校验：根据 moe_a2a_backend 决定默认算法，并拒绝不兼容的 static/lp。

```
    def _handle_eplb_and_dispatch(self):
        if self.enable_eplb and (self.expert_distribution_recorder_mode is None):
            self.expert_distribution_recorder_mode = "stat"
            logger.warning("EPLB 已启用，自动设置 expert_distribution_recorder_mode。")

        # 无 a2a 后端时所有 EP rank 对相同 token 计算部分和，因此选择必须跨 rank 一致。
        needs_rank_invariant_dispatch = self._resolved().moe_a2a_backend == "none"

        if (self.enable_eplb or (self.init_expert_location != "trivial")) and (
```

```
        self.ep_dispatch_algorithm is None
    ):
        self.ep_dispatch_algorithm = (
            "dynamic" if needs_rank_invariant_dispatch else "static"
        )

    # dynamic / fake 会切换为按行索引选择；static 读取 per-rank 表，lp 在内核内采样。
    if needs_rank_invariant_dispatch and self.ep_dispatch_algorithm in ("static", "lp"):
        raise ValueError(
            f"--ep-dispatch-algorithm {self.ep_dispatch_algorithm} picks a "
            "different physical replica per rank, which only holds up when an "
            "a2a backend routes each token to a single rank. Use "
            "--ep-dispatch-algorithm dynamic with --moe-a2a-backend none."
        )

    if self.enable_eplb and self.ep_join_mode != "scale":
        assert self._resolved().ep_size > 1
```

评论区精华

Codex 自动审查提出多个问题，作者均有所回应：

- P1 _can_dual_stream_graph：Codex 指出无 --enable-eplb 时 graph bypass 仍可能触发，导致自定义物理放置被忽略。作者确认有效，但这是 PR 早期 ungating 引入的，且不通过 EPLB 可达，最终版本已还原 ungating，并建议单独 PR。
- P2 --ep-num-redundant-experts 单独使用：Codex 指出仅设置冗余专家数而不启用 EPLB 时，算法保持 None，冗余副本永不被使用。作者承认有效，但属于 EPLB 之外的独立路径，已移出本 PR。
- P2 行索引均衡局限：Codex 担心热专家只在特定残差行出现时会固定选同一副本。作者未直接回应，但 PR 描述说明连续行交替副本，正常批次下风险可控。
- P2 拒绝 static 过于严格：Codex 认为冗余专家为 0 时 static 安全，无条件拒绝会破坏既有配置。作者最终仍无条件拒绝，属有意设计，避免未来引入冗余专家时静默出错。
- P1: _can_dual_stream_graph 绕过逻辑在无 EPLB 时可能错误 (correctness): 作者确认该问题有效，但因为它由 PR 中的 ungating 引入且不通过 EPLB 可达，已将该修复移出 PR，并在最终版本中还原 ungating。
- P2: 冗余专家单独设置时不会选择 dispatch 算法 (correctness): 作者承认有效，但认为不属于 EPLB 路径，已从本 PR 移除，留待后续单独修复。
- P2: 按行索引选副本可能在某些批次下仍导致热点 (performance): 作者未直接回应，但 PR 描述说明连续行交替副本，均匀分裂是 EPLB 放置求解器的假设，可能认为此风险在正常批次下可接受。
- P2: static 在无副本时可安全使用但被无条件拒绝 (design): 最终代码仍然无条件拒绝 static/lp，作者在 PR 中明确说明这是设计决策，避免未来复制专家时静默错误。

风险与影响

- 风险：
  - 核心调度路径变更：dynamic 成为 no-a2a 路径默认算法，影响所有使用 EPLB 但未显式指定算法的 MoE 模型部署（如 DeepSeek）。
  - 行索引均衡局限：对重复 prompt 等不均衡行分布，可能出现副本闲置；本 PR 不解决该问题，后续可能需要按每个逻辑专家的出现次数选择。
  - 兼容性破坏：原先 --moe-a2a-backend none --ep-dispatch-algorithm static 的配置现在启动即报错，用户需调整。
  - 未解决问题：--init-expert-location 与 --ep-num-redundant-experts 单独使用时的缺陷仍然存在，且 fake 算法在 rank-invariant 路径下使用 uniform_() RNG 可能跨 rank 不一致，属已知遗留风险。
- 影响：
  - 用户侧：修复了 EPLB + 冗余专家在默认 no-a2a 后端下的静默准确率下降，GSM8K 从 0.42 恢复到 0.62-0.66，输出质量显著提升。
  - 系统侧：冗余专家在 no-a2a 路径上真正参与负载分担，而非死重；跨 rank 一致性由纯函数保证，不再依赖隐式 seed 对齐。
  - 团队侧：新增 rank_invariant 概念和夜间准确性测试，为后续 MoE 调度演进提供基础，但也引入了需要维护的前置校验逻辑。
  - 风险标记：核心调度路径变更，兼容性破坏（拒绝 static/lp），存在两个 out-of-scope 已知缺陷

关联脉络

- PR #30756 Integrate pplx a2a backend: 同为 MoE 调度后端演进：一个为 a2a 后端扩展，一个完善无 a2a 后端时的调度语义，二者共同围绕专家并行与冗余专家分配。