PR #26862 完整报告

sgl-project/sglang

Add random-ids dataset, round-robin expert simulation, and kill_process_tree logging

合并时间：2026-06-01 11:50

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26862

执行摘要

- 一句话：添加 random-ids 数据集和轮询专家模拟
- 推荐动作：值得关注，尤其是 MoE 基准测试流程的设计思路和确定性模拟的实现。

功能与动机

基准测试中，使用随机或虚拟权重时，MoE 的路由会引入专家负载不均衡，导致 benchmark 结果噪声大。通过确定性轮询分配专家，可以消除路由随机性，获得可重复的性能数据。此外，预生成 token ID 的数据集 random-ids 可避免 tokenization 开销，专注测试推理引擎本身。

实现拆解

1. 在 environ.py 中添加 SGLANG_SIMULATE_ROUND_ROBIN_EXPERTS 环境变量，默认 False。
2. 在 topk.py 中新增 _make_round_robin_expert_ids 函数，根据 token 序号和 layer_id 进行轮询计算专家 ID，并处理 topk=0 边界。
3. 在 select_experts 中读取两个环境变量，若同时设置则抛出 ValueError；否则根据设置分别执行轮询或均匀模拟，并对均匀模拟增加了 k>0 守卫防止崩溃。
4. 在 common.py 的 kill_process_tree 开头添加 logger.info 记录调用参数。
5. 在 bench_one_batch_server_internal.py 的 CLI 和数据集校验列表中加入 random-ids。

关键文件：

- python/sglang/srt/layers/moe/topk.py（模块 MoE 路由；类别 source；类型 core-logic；符号 _make_round_robin_expert_ids）：核心变更：新增轮询专家 ID 生成函数，重构 uniform 模拟，添加互斥检查
- python/sglang/srt/environ.py（模块 环境配置；类别 source；类型 configuration）：注册新环境变量入口
- python/sglang/srt/utils/common.py（模块 工具函数；类别 source；类型 core-logic）：为 kill_process_tree 增加日志，便于调试进程清理问题
- python/sglang/test/bench_one_batch_server_internal.py（模块 基准测试；类别 test；类型 test-coverage）：扩展 benchmark 支持 random-ids 数据集，用于预生成 token ID 的基准测试

关键符号：_make_round_robin_expert_ids, select_experts, kill_process_tree

关键源码片段

python/sglang/srt/layers/moe/topk.py

核心变更：新增轮询专家 ID 生成函数，重构 uniform 模拟，添加互斥检查

```
# 注：以下代码位于 python/sglang/srt/layers/moe/topk.py

def _make_round_robin_expert_ids(
    num_tokens: int,
    topk: int,
    num_experts: int,
    *,
    device: torch.device,
    dtype: torch.dtype,
    layer_id: Optional[int] = None,
) -> torch.Tensor:
    """生成轮询专家 ID，使每个 token 按顺序分配不同专家，避免路由随机性。

    当 topk == 0 时返回空张量，防止后续运算崩溃。
    """
    if topk == 0:
        return torch.empty((num_tokens, 0), device=device, dtype=dtype)

    step = max(num_experts // topk, 1) # 每步跳过 expert 数，确保均匀分布
    layer_offset = 0 if layer_id is None else layer_id # 不同层引入偏移，避免层间对齐
    offsets = torch.arange(num_tokens, device=device, dtype=dtype).unsqueeze(1) # token
    行索引
    steps = torch.arange(topk, device=device, dtype=dtype).unsqueeze(0) * step # topk 列步长
    return (offsets + layer_offset + steps) % num_experts


def select_experts(...):
    # ... 省略前面路由逻辑 ...

    # 读取环境变量并做互斥检查
    simulate_uniform_experts = envs.SGLANG_SIMULATE_UNIFORM_EXPERTS.get()
    simulate_round_robin_experts = envs.SGLANG_SIMULATE_ROUND_ROBIN_EXPERTS.get()
    if simulate_uniform_experts and simulate_round_robin_experts:
        raise ValueError(
            "SGLANG_SIMULATE_UNIFORM_EXPERTS and "
            "SGLANG_SIMULATE_ROUND_ROBIN_EXPERTS are mutually exclusive"
        )

    if simulate_uniform_experts:
        # 均匀模拟：为每个 token 随机设置起始偏移，然后按步长取专家
        num_tokens, k = topk_ids.shape
        num_experts = router_logits.shape[1]
        if k > 0: # 防止 topk_ids 为空时除零
            offsets = torch.randint(
```

```
            0, num_experts, (num_tokens, 1), device=topk_ids.device
        )
        steps = torch.arange(k, device=topk_ids.device).unsqueeze(0)
        step = max(num_experts // k, 1)
        topk_ids = ((offsets + steps * step) % num_experts).to(topk_ids.dtype)
        topk_weights = torch.ones_like(topk_weights) / k
    elif simulate_round_robin_experts:
        # 轮询模拟：基于 token 和 layer 确定性分配专家
        num_tokens, k = topk_ids.shape
        num_experts = router_logits.shape[1]
        topk_ids = _make_round_robin_expert_ids(
            num_tokens, k, num_experts,
            device=topk_ids.device, dtype=topk_ids.dtype, layer_id=layer_id,
        )
        if k > 0:
            topk_weights = torch.full_like(topk_weights, 1.0 / k)

    # ... 后续处理 ...
```

评论区精华

PR 无实质性讨论，gemini-code-assist 机器人仅确认无 review 意见。

- 暂无高价值评论线程

风险与影响

- 风险：新增环境变量若与现有配置冲突可能引发 ValueError，但已做互斥检查；轮询模拟仅用于 benchmark，不影响生产；kill_process_tree 日志不会造成问题。
- 影响：影响范围小，主要影响 MoE 基准测试用户；可复现性提升；日志有助于调试进程清理问题。
- 风险标记：互斥环境变量需同时设置，k=0 边界已修复，仅用于 benchmark 不影响生产

关联脉络

- 暂无明显关联 PR