PR #7218 完整报告

PaddlePaddle/FastDeploy

[RL] support moe-topk use topk_reduce_func

合并时间：2026-04-09 11:01

原文链接：http://prhub.com.cn/PaddlePaddle/FastDeploy/pull/7218

PR 7218 分析报告

执行摘要

本 PR 引入 topk_reduce_func 参数以支持 MoE TopK 计算的自定义归一化，当 FD_USE_PHI_MOE_TOPK 生效时将 normalize 和 scaling 计算移至算子外部，并移除旧的 moe_topk_select 实现。影响使用 MoE 的模型，需注意参数配置以避免数值风险，是一个有意义的改进。

功能与动机

为解决不同模型在 MoE TopK 计算逻辑的细微差异，支持在组网时传入 topk_reduce_func 参数，保证数值准确性。根据 PR body，动机是“支持在组网时传入 topk_reduce_func 保证数值的准确性，并在 FD_USE_PHI_MOE_TOPK 生效时，不在 noaux_ac 算子内部计算 normalize 和 scaling”。

实现拆解

关键改动按模块梳理：

模块	文件	关键变更
MoE 层	moe.py	在 get_moe_scores 函数添加 topk_reduce_func 参数，处理 FD_USE_PHI_MOE_TOPK 下的归一化和缩放。
MoE 后端	fused_moe_deepgemm_backend.py	移除 moe_topk_select 函数，调整调用逻辑。
专家选择	ep.py	传递 topk_reduce_func 参数给底层算子。
模型示例	glm4_moe.py	显式传入 topk_reduce_func 参数。
测试	多个测试文件	更新测试用例以适配新参数和移除旧函数。

代码示例（从 moe.py 提取）：

```
if envs.FD_USE_PHI_MOE_TOPK:
    if original_renormalize:
```

```
        if topk_reduce_func is not None:
            topk_values = topk_values / topk_reduce_func(topk_values)
        else:
            topk_values = topk_values / (topk_values.sum(axis=-1, keepdim=True) + 1e-20)
    if original_routed_scaling_factor != 1.0:
        topk_values *= original_routed_scaling_factor
```

评论区精华

review 讨论中的关键交锋：

- fastdeploy-bot：“当 FD_USE_PHI_MOE_TOPK=True 且 renormalize=True 时，如果 topk_reduce_func=None，归一化操作不会执行。”
  - zoooo0820（作者）在回复关于 glm4_moe.py 显式传入参数时解释：“get_moe_scores 的默认值是和 noaux_tc 里的行为一致，保证 Flag 打开的时候其他模型行为和之前对齐。这里 glm 只是恰好和算子里是一样的，感觉显式加上是不是更好点”。
- zhangbo9674提问：“routed_scaling_factor 也受 renormalize 控制么？”（未在讨论中明确回答）。

风险与影响

风险：

1. 数值风险：在 FD_USE_PHI_MOE_TOPK=True 且 renormalize=True 条件下，如果模型未传入 topk_reduce_func，归一化可能跳过，导致输出偏差。
2. 行为不一致：仅 glm4_moe.py 显式传入参数，其他模型可能产生隐式差异。
3. 测试覆盖：codecov 报告有 3 行缺失覆盖，可能未覆盖边缘情况。

影响：

- 对用户：需了解新参数以配置模型，特别是在使用 FD_USE_PHI_MOE_TOPK 时。
- 对系统：提升 MoE TopK 灵活性，可能改善推理准确性。
- 对团队：代码简化但引入配置复杂性。

关联脉络

与历史 PR 的关联：

- PR #7238：MoE 在 SM103 架构的 bugfix，共享 MoE 优化脉络。
- PR #7053：支持 Blackwell 架构 MoE GEMM，同属 MoE 性能改进系列。 这些 PR 共同显示仓库在持续优化 MoE 功能，本 PR 通过自定义归一化进一步细化计算逻辑。