# PR #2262 完整报告

THUDM/slime

feat(glm5): align Megatron DeepEP training with SGLang rollout

合并时间：2026-08-11 11:46

原文链接：http://prhub.com.cn/THUDM/slime/pull/2262

## 执行摘要

- 一句话：对齐 Megatron 训练与 SGLang rollouts，新增确定性 GLM-5 对齐路径
- 推荐动作：值得精读。该 PR 展示了如何系统性对齐训练与推理引擎的数值行为，包含大量可借鉴的设计：自定义 autograd 函数封装外部 kernel、通过 Triton 内核实现确定性路由重排、使用直通估计器（STE）处理对齐权重、基于 cache key 的权重对齐缓存、以及通过 layerwise dump 精确验证对齐效果。对从事多后端一致性、MoE 训练、FP8 量化的工程师有很高参考价值。建议关注点：1）routing_replay 对非 GLM-5 模型的潜在影响；2）DeepEP 成为唯一对齐后端的回退策略；3）CI 门禁的 fixture 依赖管理。

## 功能与动机

PR 标题和描述明确指出目标是“添加一条确定性的 GLM-5 训练 / 采样对齐路径”，覆盖 Megatron 训练、SGLang 推理、DeepGEMM、DeepEP、DSA 稀疏注意力和 FP8 KV cache。描述中强调需要“capturing exact top-k slot order”“validating receive layouts”“deterministic backward”，并以精确对齐指标（logprob MAE、layerwise 零差异）作为验证标准。该工作是为了让训练阶段与 rollout 阶段在数值行为上完全一致，从而保证强化学习训练的信号质量。

## 实现拆解

1. 新增 DeepGEMM 对齐前向路径（slime/backends/megatron_utils/alignment/deepgemm_forward.py）：将选中的 Transformer Engine 线性层替换为自定义 autograd 函数，forward 使用 SGLang 风格的 block-FP8 DeepGEMM，backward 使用显式 BF16 GEMM 计算 dgrad/wgrad，并为融合 LayerNorm/RMSNorm 线性层提供解析的 norm 梯度。该实现要求 TP=1，以匹配 SGLang 的 dense-TP1 执行并避免 row-parallel 部分求和的舍入差异。
2. 新增确定性路由 Triton kernels（slime/backends/megatron_utils/alignment/deterministic_route_kernels.py）：实现 _scatter_routes_forward_kernel、_scatter_routes_backward_kernel、_ordered_route_grad_kernel、_compact_route_positions_kernel，用无原子、每输出元素单写者的方式复现 DeepEP 和 SGLang 的确定性路由重排和梯度累积顺序，backward 中特意保持每 top-k slot 的 BF16 in-place 加法舍入边界。
3. 修改 GLM-5 模型（slime_plugins/models/glm5/glm5.py）：新增 _SGLangAbsorbWeightSTE（直通估计器）、_SGLangIndexerHeadWeights（使用

sglang deep_gemm_wrapper 的 FP8 indexer head 计算）、_get_fp8_aligned_absorb_weight（按 weight 版本 / 数据指针等缓存对齐权重）、_SGLangRoPE、SGLangSparseMLA 等，对齐 fused q-RMSNorm 输入、indexerprojections、FP8 logits、RoPE 和 sparse attention。

4. 路由捕获与重放（slime/utils/routing_replay.py）：新增 ORDERED_TOPK_CAPTURE_ROUTER 注册机制，在 forward hook 中捕获当前 router 的 top-k 顺序，并在 target 中使用 ordered top-k 代替 Megatron 默认的 sorted top-k；同时重放时对非连续层视图做 compact 处理，避免持有全量路由信息。
5. 新增层对齐 dump 与比较工具（slime/backends/megatron_utils/alignment/layerwise_alignment.py、slime/utils/compare_glm52_layerwise.py）：Megatron 侧通过 pre_forward/record_layer/record_module/post_forward 将选层输出 dump 到 rank 目录，供 SGLang 侧与 train 侧逐 token 比较；compare 工具负责加载 dump、重建序列、排除 HEALTH_CHECK_ 请求，并计算层输出差异。
6. 配套测试、Docker patch 和 CI 门禁：新增 tests/test_deepgemm_moe_forward.py（2326 行）、test_deepgemm_forward.py、test_glm52_6layer_deterministic_e2e.py、test_glm52_layerwise_comparison.py 等；新增 docker/patch/latest/sglang-deterministic.patch；在 CI workflow 模板中注册 8-GPU logprob 与 layerwise 精确对齐门禁，并在后三个 commit 中修复了代理、fixture 下载、stateless Adam 等 CI 问题。

关键文件：

- slime/backends/megatron_utils/alignment/deepgemm_forward.py（模块 后端对齐；类别 source；类型 core-logic；符号 router_gating_linear_backward, _should_log_deepgemm_summary, _format_int_ranges, _deepgemm_bf16_gemm_nn）：新增 DeepGEMM 对齐前向路径的核心实现，用自定义 autograd 替换 TE linear，支持 block-FP8 forward 和显式 BF16 backward，是全 PR 数值对齐的关键载体。
- slime/utils/compare_glm52_layerwise.py（模块 工具与验证；类别 source；类型 data-contract；符号 TrainSequence, _load_records, _find_suffix, _as_tensor）：新增的层对齐比较工具，用于加载 Megatron 层 dump 并重建训练序列，进而与 SGLang 层输出逐 token 对比，是对齐门禁的关键组成部分。
- slime/backends/megatron_utils/alignment/deterministic_route_kernels.py（模块 路由内核；类别 source；类型 core-logic；符号 _scatter_routes_forward_kernel, _scatter_routes_backward_kernel, _ordered_route_grad_kernel, _compact_route_positions_kernel）：新增的确定性路由 Triton kernels，替代 launch-heavy tensor indexing，保证 route permutation 和梯度累积的确定性，支撑 DeepEP 对齐。
- slime_plugins/models/glm5/glm5.py（模块 GLM-5 模型；类别 source；类型 core-logic；符号 _SGLangAbsorbWeightSTE, forward, backward, _SGLangIndexerHeadWeights）：GLM-5 模型入口，新增多个 autograd.Function 与 helper，对齐 indexer head、FP8 absorb weight、RoPE 和 SGLang sparse attention，是模型侧对齐的核心。
- slime/backends/megatron_utils/alignment/layerwise_alignment.py（模块 层对齐工具；类别 source；类型 test-coverage；符号 _global_rank, _first_tensor, _MegatronLayerwiseDumper, init）：新增 Megatron 层输出 dump 工具，通过在模型前

向钩子中记录指定层输出，为 layerwise 对齐门禁提供训练侧数据。

- slime/utils/routing_replay.py（模块 路由重放；类别 source；类型 core-logic；符号 _set_ordered_topk_capture_router, _capture_ordered_topk, consume_ordered_topk, register_ordered_topk_capture）：修改核心路由捕获逻辑，新增 ordered top-k 捕获机制，用于对齐 DeepEP/SGLang 的路由顺序。

关键符号：router_gating_linear_backward, _deepgemm_bf16_gemm_nn, _deepgemm_bf16_gemm_nt, scatter_routes_forward, scatter_routes_backward, ordered_route_grad, _SGLangAbsorbWeightSTE.forward, _SGLangAbsorbWeightSTE.backward, _SGLangIndexerHeadWeights.forward, _SGLangIndexerHeadWeights.backward, _get_sglang_indexer_head_weights, _get_fp8_aligned_absorb_weight, _apply_sglang_rope_forward, register_ordered_topk_capture, consume_ordered_topk, _compute_topk_for_current_router, load_train_sequences, _sglang_layer_token_rows, _MegatronLayerwiseDumper.pre_forward, _MegatronLayerwiseDumper.record_layer, _MegatronLayerwiseDumper.record_module, _MegatronLayerwiseDumper.post_forward

## 关键源码片段

### slime/utils/routing_replay.py

修改核心路由捕获逻辑，新增 ordered top-k 捕获机制，用于对齐 DeepEP/SGLang 的路由顺序。

```
"""路由重放与 ordered top-k 捕获机制。"""

import os
import torch

# 全局单例：当前 forward 中需要捕获 top-k 顺序的 router module。
ORDERED_TOPK_CAPTURE_ROUTER = None

def set_routing_replay(replay):
    """设置全局 RoutingReplay 实例（历史逻辑，保持兼容）。"""
    global ROUTING_REPLAY
    ROUTING_REPLAY = replay


def _set_ordered_topk_capture_router(router):
    """记录当前正在执行的 router 模块，后续 compute_topk 会检查它。"""
    global ORDERED_TOPK_CAPTURE_ROUTER
    ORDERED_TOPK_CAPTURE_ROUTER = router


def _capture_ordered_topk(top_indices):
    """把当前 router 本次 forward 的 top-k 索引暂存到模块属性中。"""
    router = ORDERED_TOPK_CAPTURE_ROUTER
    if router is not None:
```

```
        router._slime_ordered_topk_indices = top_indices


def consume_ordered_topk(module):
    """取出并清除模块上暂存的 ordered top-k 索引。"""
    return module.__dict__.pop("_slime_ordered_topk_indices", None)


def register_ordered_topk_capture(module):
    """为 MoE router 注册前后向钩子，捕获其 forward 的 top-k 顺序。"""
    if getattr(module, "_slime_ordered_topk_capture_registered", False):
        return

    def pre_forward_hook(patched_module, *args, **kwargs):
        # 在 router 前向之前设置捕获目标。
        del args, kwargs
        _set_ordered_topk_capture_router(patched_module)

    def forward_hook(patched_module, *args, **kwargs):
        # 前向结束后清除捕获目标，避免影响后续其他 router。
        del args, kwargs
        if ORDERED_TOPK_CAPTURE_ROUTER is patched_module:
            _set_ordered_topk_capture_router(None)

    module.register_forward_pre_hook(pre_forward_hook)
    module.register_forward_hook(forward_hook)
    module._slime_ordered_topk_capture_registered = True


def _compute_topk_for_current_router(
    old_compute_topk,
    scores,
    topk,
    num_groups=None,
    group_topk=None,
):
    # SGLang 的确定性的 biased top-k 使用 torch.topk(..., sorted=False),
    # Megatron 默认 sorted=True。选出的专家集合相同，但低延迟 rollouts 的
    # 专家总序按 top-k 列顺序消费，因此排序差异会改变 BF16 累加顺序。
    # 此覆盖仅影响被 DeepEP 对齐桥注册的 router，且仅在非 grouped 路径生效。
    if ORDERED_TOPK_CAPTURE_ROUTER is not None and not group_topk:
        return torch.topk(scores, k=topk, dim=1, sorted=False)

    return old_compute_topk(
        scores,
        topk,
        num_groups=num_groups,
        group_topk=group_topk,
    )
```

## 评论区精华

该 PR 的 review 评论中，gongshaotian 提问“How is 25% calculated? Can you provide a reproducible and testable startup configuration (especially for SGLang)?”；作者 zhuzilin 回应 25% 是根据上述运行的 e2e step time 计算得出，并指出新增的 CI 中包含一个 e2e mismatch 测试和一个 layerwise no-diff 测试，可复现启动配置。没有其他 review 讨论记录。

- 25% 效率提升的计算依据与可复现配置 (question): zhuzilin 回应 25% 由上述运行的 e2e step time 计算得出，并指出新增 CI 中已包含 e2e mismatch 测试和 layerwise no-diff 测试，提供可复现配置。

## 风险与影响

- 风险：
  1. 数值对齐路径默认启用 DeepGEMM 替换 TE linear，若 kernel 或配置失误，会影响训练数值正确性；当前实现要求 TP=1，在 TP>1 场景下可能不适用或产生异常。
  2. DeepEP 桥接移除了普通的 Megatron all-to-all alignment bridge，使 DeepEP 成为唯一受支持的 MoE alignment 后端，若环境中 DeepEP 不可用或行为不一致，可能导致回退缺失。
  3. routing_replay.py 中覆盖 router 的 compute_topk 逻辑为所有“非 grouped”路径（无 group_topk）提供 sorted=False topk，可能影响其他 MoE 模型（非 GLM-5）的数值语义。
  4. 新增的 SGLang patch（sglang-deterministic.patch）和 DeepEP/DeepGEMM fork 依赖（zhuzilin 分支）增加了外部依赖的维护风险，需要持续跟随上游更新。
  5. 层对齐 dump 工具在训练 forward 中记录 CPU tensor，可能导致额外的内存 / 性能开销；CI 中门禁需要下载公开 fixture，若网络不稳定会影响 CI 稳定性。
  6. 大量新增测试有 CUDA/ 多 GPU 依赖（如 8 卡 e2e），在无 GPU 环境下可能被跳过，存在测试盲区。 - 影响：影响范围：主要影响 GLM-5 模型的训练链路（Megatron 后端）与 rollout 链路（SGLang 后端），以及所有使用 MoE 路由的模型（routing_replay 修改是通用逻辑）。对生产训练团队而言，该 PR 提供了可量化的训练 /rollout 对齐能力，但引入了新的数值对齐 hook 和更严格的 CI 门禁。对普通用户，如果未显式开启对齐路径（环境变量、配置），默认行为应保持兼容，但需要验证默认路径是否仍与旧版本一致。对团队而言，该 PR 大幅扩展了测试矩阵和 Docker patch 维护工作，且依赖作者 fork 的 DeepEP/DeepGEMM 分支，需要团队协调上游合入。 - 风险标记：核心训练路径变更，外部依赖 fork, TP=1 限制，CI fixture 依赖网络，影响 MoE 通用路由逻辑

## 关联脉络

- PR #2252 [release] bump to v0.3.1: 该 PR 集成了 SGLang/Megatron 补丁，与本 PR 的 SGLang-deterministic patch 和 Megatron 对齐逻辑高度相关。
- PR #2220 Optimize update weight: 该 PR 优化了 MoE 权重更新，涉及 expert routing 和 hf_weight_iterator，与本 PR 的 DeepEP 对齐和路由顺序改动相关。

- PR #2208 Support reloading the default process group: 该 PR 修改了 slime/utils/reloadable_process_group.py 和 slime/ray/rollout.py，与本 PR 修改的 rollout 相关，且本 PR 也涉及 Ray child 环境的 proxy 处理。
- PR #2250 Add lightweight rollout hooks and sampling controls: 该 PR 新增 rollout 采样钩子，与本 PR 的 rollout 对齐路径存在概念重叠，且都涉及 slime/ray/rollout.py 和参数配置。
- PR #2181 [3/n] Disaggregated rollout: engine-side /pull_weights: 该 PR 为 SGLang 引擎添加 /pull_weights 端点，与本 PR 的 SGLang 对齐 patch 和权重传输链路直接相关。