PR #26673 完整报告

sgl-project/sglang

[refactor] remove unused op_mlp

合并时间：2026-05-29 17:38

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26673

执行摘要

- 一句话：删除 6 个 MoE 模型中的未使用 op_mlp 方法
- 推荐动作：该 PR 属于常规清理，建议快速合并。值得注意的设计决策是 TBO 调度模式中 MLP 步骤已通过其他方式（如直接内联）实现，此删除验证了模块化重构的进展。

功能与动机

PR 说明 op_mlp 函数未被使用，其存在会造成困惑（"op_mlp function is not used and its presence is confusing"）。清理未使用代码以简化模型实现。

实现拆解

1. 在 python/sglang/srt/models/deepseek_v2.py 中删除 op_mlp 方法及其包含的稠密 DP 条件判断逻辑。
2. 在 python/sglang/srt/models/glm4_moe.py 中删除完全相同的 op_mlp 方法。
3. 在 python/sglang/srt/models/glm4_moe_lite.py 中删除相同的 op_mlp 方法。
4. 在 python/sglang/srt/models/minimax_m2.py 中删除仅调用 self.block_sparse_moe 的 op_mlp 方法。
5. 在 python/sglang/srt/models/mimo_v2.py 和 python/sglang/srt/models/qwen3_moe.py 中分别删除直接调用 self.mlp 的 op_mlp 方法。所有删除均未改动其他方法（如 op_comm_prepare_mlp/op_comm_postprocess_layer），因为这些模型实际使用的调度路径已不再经过 op_mlp。

关键文件：

- python/sglang/srt/models/deepseek_v2.py（模块 模型层；类别 source；类型 data-contract；符号 op_mlp）：核心 MoE 模型，删除了含条件判断的 op_mlp，影响最大
- python/sglang/srt/models/glm4_moe.py（模块 模型层；类别 source；类型 data-contract；符号 op_mlp）：另一个重要 MoE 模型，同样删除含条件分支的 op_mlp，代表一类变更
- python/sglang/srt/models/glm4_moe_lite.py（模块 模型层；类别 source；类型 data-contract；符号 op_mlp）：第三个模型，与 glm4_moe 相同模式
- python/sglang/srt/models/minimax_m2.py（模块 模型层；类别 source；类型 data-contract；符号 op_mlp）：MiniMax-M2 模型，删除直接调用 block_sparse_moe 的

op_mlp

- python/sglang/srt/models/mimo_v2.py（模块 模型层；类别 source；类型 data-contract；符号 op_mlp）：Mimo-V2 模型，删除直接调用 self.mlp 的 op_mlp
- python/sglang/srt/models/qwen3_moe.py（模块 模型层；类别 source；类型 data-contract；符号 op_mlp）：Qwen3-MoE 模型，删除直接调用 self.mlp 的 op_mlp

关键符号：op_mlp

关键源码片段

python/sglang/srt/models/deepseek_v2.py

核心 MoE 模型，删除了含条件判断的 op_mlp，影响最大

```
# python/sglang/srt/models/deepseek_v2.py
# 以下是删除 op_mlp 后的连续方法片段，展示了 TBO 调度中的 MLP 准备与后处理直接衔接。
# 注意：op_mlp 原本位于 op_comm_prepare_mlp 与 op_comm_postprocess_layer
之间，现已移除。

class DeepseekV2DecoderLayer(nn.Module):
    # ... 省略其他方法

    def op_comm_prepare_mlp(self, state):
        state.hidden_states_mlp_input, state.residual_after_comm_pre_mlp = (
            self.layer_communicator.prepare_mlp(
                state.pop("hidden_states_after_attn"),
                state.pop("residual_after_input_ln"),
                state.forward_batch,
            )
        )

    # op_mlp 已被移除，该函数原负责调用 self.mlp 并设置 hidden_states_mlp_output。
    # 当前代码中，postprocess 直接使用 state 中已有的 hidden_states_mlp_output，
    # 该值可能由其他路径（如融合的前向）设置。

    def op_comm_postprocess_layer(self, state):
        hidden_states, residual = self.layer_communicator.postprocess_layer(
            state.pop("hidden_states_mlp_output"),
            state.pop("residual_after_comm_pre_mlp"),
            state.forward_batch,
        )
        output = dict(
            positions=state.positions,
            hidden_states=hidden_states,
            residual=residual,
            forward_batch=state.forward_batch,
            zero_allocator=state.zero_allocator,
            tbo_subbatch_index=state.tbo_subbatch_index,
        )
```

```
        state.clear(expect_keys={"positions", "forward_batch", "zero_allocator", "tbo_subbatch_
        index"})
        return output
```

python/sglang/srt/models/glm4_moe.py

另一个重要 MoE 模型，同样删除含条件分支的 op_mlp，代表一类变更

```
# python/sglang/srt/models/glm4_moe.py
# 删除 op_mlp 后，op_comm_prepare_mlp 与 op_comm_postprocess_layer 直接相连。

class Glm4MoeDecoderLayer(nn.Module):
    # ...

    def op_comm_prepare_mlp(self, state):
        state.hidden_states_mlp_input, state.residual_after_comm_pre_mlp = (
            self.layer_communicator.prepare_mlp(
                state.pop("hidden_states_after_attn"),
                state.pop("residual_after_input_ln"),
                state.forward_batch,
            )
        )

    # 原 op_mlp 已被删除。该函数原为：
    # def op_mlp(self, state):
    # hidden_states = state.pop("hidden_states_mlp_input")
    # if not (enable_moe_dense_fully_dp() and (not self.is_layer_sparse) and hidden_states.
    shape[0] == 0):
    # state.hidden_states_mlp_output = self.mlp(hidden_states, state.forward_batch)
    # else:
    # state.hidden_states_mlp_output = hidden_states

    def op_comm_postprocess_layer(self, state):
        hidden_states, residual = self.layer_communicator.postprocess_layer(
            state.pop("hidden_states_mlp_output"),
            state.pop("residual_after_comm_pre_mlp"),
            state.forward_batch,
        )
        # ... 相同后处理逻辑
```

评论区精华

无相关讨论。PR 获得单一批准，未产生 review 评论。

- 暂无高价值评论线程

风险与影响

- 风险：主要风险在于如果存在某些隐藏的执行路径（例如通过配置启用的 TBO 流程）依赖 op_mlp，则删除会导致运行时属性缺失错误。但由于 PR 作者确认未被使用且 reviewer 批

准，当前代码库中所有调用点均已移除，回归可能性低。建议合并后密切关注 CI 测试结果。

- 影响：对用户无影响（无 API 或行为变化）。对开发者，减少了模型实现中的死代码，降低阅读成本。对系统，无性能或功能影响。
- 风险标记：暂无

关联脉络

- 暂无明显关联 PR