PR #25825 完整报告

sgl-project/sglang

[Refactor] Pass PP start_layer via model constructor instead of forward_batch.token_to_kv_pool

合并时间：2026-05-20 13:16

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25825

执行摘要

- 一句话：通过构造函数传递 PP start_layer 以解耦 ForwardBatch
- 推荐动作：该 PR 是典型的接口清洁重构，值得精读。展示了如何分步将静态配置从运行时对象剥离，并且带测试覆盖和连带 bug 修复。设计决策（使用构造函数参数而非全局单例或上下文）值得借鉴。

功能与动机

目前多个模型 Attention 层通过 forward_batch.token_to_kv_pool.start_layer 判断当前层是否为 PP rank 第一层，但 start_layer 是模型初始化时由 make_layers 确定的静态配置，不会在 forward 之间变化。将其从 ForwardBatch 提升为构造参数，概念更清晰，也为后续移除 ForwardBatch 中更多静态信息提供模式。

实现拆解

1. 在 llama.py、glm4_moe.py、qwen3.py、qwen3_moe.py、qwen2.py、qwen2_moe.py 的 Attention 类和 DecoderLayer 类中添加 start_layer 参数（默认 0），在 __init__中保存为 self.start_layer。
2. 在各模型的 __init__中，调用 get_pp_indices 计算当前 PP rank 的起始层号 pp_start_layer，通过 make_layers 的 lambda 传递给每个 DecoderLayer。
3. 在 Attention 的 forward_prepare_npu（以及 GLM4 的 forward_prepare）中，将条件判断从 self.attn.layer_id == forward_batch.token_to_kv_pool.start_layer 改为 self.attn.layer_id == self.start_layer。
4. 修复 EAGLE 子类：llama_eagle.py、llama_eagle3.py、qwen2_eagle.py 中原本使用 positional args 调用 super().init，因新参数插入导致参数错位；改为 keyword args 确保 quant_config 正确传递。 测试：在 B200 集群上执行了 Llama-3.1-8B、Qwen3-8B、Qwen3-30B-A3B、GLM-4.5-Air-FP8 的 PP 一致性测试，全部通过。无性能测试必要。

关键文件：

- python/sglang/srt/models/llama.py（模块 模型层；类别 source；类型 data-contract；符号 LlamaAttention.init, LlamaAttention.forward_prepare_npu, LlamaDecoderLayer.init, LlamaModel.init）：核心修改文件之一；展示了从 Attention、DecoderLayer 到模型类的完整 start_layer 传递链，并修改了 forward_prepare_npu 中的条件判断。

- python/sglang/srt/models/glm4_moe.py（模块 模型层；类别 source；类型 data-contract；符号 Glm4MoeAttention.init, Glm4MoeAttention.forward_prepare, Glm4MoeDecoderLayer.init, Glm4MoeModel.init）：GLM4-MoE 模型的 Attention 层和 DecoderLayer 同样需要 start_layer 迁移，并修改 forward_prepare 中的条件判断。
- python/sglang/srt/models/qwen3.py（模块 模型层；类别 source；类型 data-contract；符号 Qwen3Attention.init, Qwen3Attention.forward_prepare_npu, Qwen3DecoderLayer.init）：Qwen3 模型的 Attention 层同样需要 start_layer 构造参数和 forward_prepare_npu 修改。
- python/sglang/srt/models/qwen3_moe.py（模块 模型层；类别 source；类型 data-contract；符号 Qwen3MoeAttention.init, Qwen3MoeAttention.forward_prepare_npu, Qwen3MoeDecoderLayer.init）：Qwen3-MoE 模型的 Attention 层需要 start_layer 迁移，并修改 forward_prepare_npu。
- python/sglang/srt/models/qwen2.py（模块 模型层；类别 source；类型 data-contract；符号 Qwen2Attention.init, Qwen2DecoderLayer.init, Qwen2Model.init）：Qwen2 模型需要添加 start_layer 传递，包括模型 init 中计算 pp_start_layer。
- python/sglang/srt/models/qwen2_moe.py（模块 模型层；类别 source；类型 data-contract；符号 Qwen2MoeAttention.init, Qwen2MoeDecoderLayer.init, Qwen2MoeModel.init）：Qwen2-MoE 模型需要添加 start_layer 传递。
- python/sglang/srt/models/llama_eagle.py（模块 模型层；类别 source；类型 data-contract；符号 LlamaEagleModel.init）：EAGLE 子类因 start_layer 参数插入导致 super().__init__参数错位，需修复为 keyword args。
- python/sglang/srt/models/llama_eagle3.py（模块 模型层；类别 source；类型 data-contract；符号 LlamaEagle3Model.init）：EAGLE3 子类同样因参数位置问题需要修复。
- python/sglang/srt/models/qwen2_eagle.py（模块 模型层；类别 source；类型 data-contract；符号 Qwen2EagleModel.init）：Qwen2 Eagle 子类因参数位置问题需要修复。

关键符号：LlamaAttention.init, LlamaAttention.forward_prepare_npu, Glm4MoeAttention.init, Glm4MoeAttention.forward_prepare, Qwen3Attention.init, Qwen3Attention.forward_prepare_npu, Qwen3MoeAttention.init, Qwen3MoeAttention.forward_prepare_npu, Qwen2Attention.init, Qwen2DecoderLayer.init, Qwen2MoeDecoderLayer.init, LlamaEagleModel.init, LlamaEagle3Model.init, Qwen2EagleModel.init

关键源码片段

python/sglang/srt/models/llama.py

核心修改文件之一；展示了从 Attention、DecoderLayer 到模型类的完整 start_layer 传递链，并修改了 forward_prepare_npu 中的条件判断。

```
# 文件：python/sglang/srt/models/llama.py
# 关键变更：从 ForwardBatch 中提取 start_layer 到构造参数
```

```
class LlamaAttention(nn.Module):
    """Attention 层，通过构造参数接收 start_layer 而非从 forward_batch 获取。"""
    def __init__(
        self,
        config: LlamaConfig,
        hidden_size: int,
        num_heads: int,
        num_kv_heads: int,
        layer_id: int = 0,
        start_layer: int = 0, # <-- 新增参数：当前 PP rank 起始层号
        rope_theta: float = 10000,
        rope_scaling: Optional[Dict[str, Any]] = None,
        rope_is_neox_style: bool = True,
        max_position_embeddings: int = 8192,
        quant_config: Optional[QuantizationConfig] = None,
        prefix: str = "",
        bias: bool = False,
    ) -> None:
        super().__init__()
        self.hidden_size = hidden_size
        self.start_layer = start_layer # 保存到实例，后续 forward 直接使用
        tp_size = get_tensor_model_parallel_world_size()
        self.total_num_heads = num_heads
        assert self.total_num_heads % tp_size == 0
        self.num_heads = self.total_num_heads // tp_size
        ...

    def forward_prepare_npu(self, positions, hidden_states, forward_batch):
        qkv, _ = self.qkv_proj(hidden_states)
        # 原判断：self.attn.layer_id == forward_batch.token_to_kv_pool.start_layer
        # 现改为：直接对比 self.start_layer（类初始化时已确定）
        if self.attn.layer_id == self.start_layer:
            self.rotary_emb.get_cos_sin_with_position(positions)
        q, k, v = split_qkv_rmsnorm_rope(
            qkv,
            self.rotary_emb.position_sin,
            self.rotary_emb.position_cos,
            self.q_size, self.kv_size, self.head_dim,
            eps=self.q_norm.variance_epsilon,
            q_weight=self.q_norm.weight,
            k_weight=self.k_norm.weight,
            q_bias=getattr(self.q_norm, "bias", None),
            k_bias=getattr(self.k_norm, "bias", None),
        )
        return q, k, v

# 在模型类 (LlamaModel) 的 __init__ 中，利用 get_pp_indices 计算 pp_start_layer
# 并通过 make_layers 的 lambda 传递给每一层
```

```
from sglang.srt.distributed import get_pp_indices

class LlamaModel(nn.Module):
    def __init__(self, config, ...):
        super().__init__()
        ...
        # 通过 get_pp_indices 获取当前 PP rank 负责的首层索引
        pp_start_layer, _ = get_pp_indices(
            config.num_hidden_layers,
            self.pp_group.rank_in_group,
            self.pp_group.world_size,
        )
        self.layers, self.start_layer, self.end_layer = make_layers(
            config.num_hidden_layers,
            lambda idx, prefix: LlamaDecoderLayer(
                config=config,
                quant_config=quant_config,
                layer_id=idx,
                start_layer=pp_start_layer, # <-- 每一层都拿到相同的 pp_start_layer
                prefix=prefix,
            ),
            pp_rank=self.pp_group.rank_in_group,
            pp_size=self.pp_group.world_size,
        )
```

python/sglang/srt/models/glm4_moe.py

GLM4-MoE 模型的 Attention 层和 DecoderLayer 同样需要 start_layer 迁移，并修改 forward_prepare 中的条件判断。

```
# 文件：python/sglang/srt/models/glm4_moe.py
# 关键变更：从 ForwardBatch 中提取 start_layer 到构造参数

class Glm4MoeAttention(nn.Module):
    def __init__(
        self,
        hidden_size: int,
        num_heads: int,
        num_kv_heads: int,
        layer_id: int = 0,
        start_layer: int = 0, # <-- 新增参数：当前 PP rank 起始层号
        rope_theta: float = 1000000,
        partial_rotary_factor: float = 0.5,
        rope_scaling: Optional[Dict[str, Any]] = None,
        max_position_embeddings: int = 8192,
        head_dim: Optional[int] = None,
        rms_norm_eps: float = 1e-05,
        attention_bias: bool = True,
        quant_config: Optional[QuantizationConfig] = None,
        use_qk_norm: bool = False,
```

```
        prefix: str = "",
        alt_stream: Optional[torch.cuda.Stream] = None,
    ) -> None:
        super().__init__()
        self.hidden_size = hidden_size
        self.start_layer = start_layer # 直接存储，不再依赖 forward_batch
        attn_tp_rank = get_attention_tp_rank()
        ...

    def forward_prepare(self, positions, hidden_states, forward_batch):
        qkv, _ = self.qkv_proj(hidden_states)
        # 原来的判断使用了 forward_batch.token_to_kv_pool.start_layer,
        # 现在改为使用实例属性 self.start_layer
        if self.attn.layer_id == self.start_layer:
            self.rotary_emb.get_cos_sin_with_position(positions)
        ...

# 在模型类 (Glm4MoeModel) 的 __init__ 中，利用 get_pp_indices 计算 pp_start_layer
from sglang.srt.distributed import get_pp_indices

# 在 Glm4MoeModel（具体类名以源码为准）的 __init__ 内：
pp_start_layer, _ = get_pp_indices(
    config.num_hidden_layers,
    self.pp_group.rank_in_group,
    self.pp_group.world_size,
)
self.layers, self.start_layer, self.end_layer = make_layers(
    config.num_hidden_layers,
    lambda idx, prefix: Glm4MoeDecoderLayer(
        layer_id=idx,
        start_layer=pp_start_layer, # <-- 传递
        config=config,
        quant_config=quant_config,
        prefix=prefix,
    ),
    pp_rank=self.pp_group.rank_in_group,
    pp_size=self.pp_group.world_size,
)
```

评论区精华

Gemini Code Assist 自动审查提供了三处反馈：

- (1) ascend_backend.py 中 _cp_allgather_and_save_kv_npu 函数误用 self，可能导致 NameError，建议使用全局 pool getter（该文件不属本 PR 变更范围）。
- (2) qwen3.py 的 Qwen3Model 需更新以计算 pp_start_layer 并传递给 make_layers（实际 PR 已做对应修改）。

- (3) base_attn_backend.py 中建议将 pool 属性初始化为 None 避免 AttributeError。 其中 (2) 已通过提交解决，(1) 和 (3) 可作为后续改进的参考。
- ascend_backend.py 中 NameError 风险 (correctness): 该文件不属本 PR 变更范围，但反馈可作为后续改进参考。
- Qwen3Model 中缺少 get_pp_indices 计算 (correctness): 实际 PR 已包含对应修改（提交 ac113dad），reviewer 可能基于旧 diff。
- base_attn_backend.py 属性懒初始化建议 (design): PR 未涉及该文件，但建议合理。

风险与影响

- 风险：主要风险是迁移不完整导致某些 PP 场景下 start_layer 始终为 0，但 PR 已逐一模型修改并通过 PP 一致性测试覆盖。EAGLE 子类参数错位已在最后提交用 keyword args 修复，但未来新增类似子类时需注意参数位置。由于是纯重构，无运行时行为变化，回归风险低。无性能风险。
- 影响：对用户无直接影响，功能一致。对开发者：减少了 ForwardBatch 的语义污染，明确了静态配置的传递路径；为后续从 ForwardBatch 移除更多类似于 token_to_kv_pool 的引用提供了参考模式。影响范围覆盖主流 Transformer 模型族，包括 LLaMA、Qwen2/3、Qwen2-MoE、Qwen3-MoE、GLM4-MoE。
- 风险标记：多模型文件改造，EAGLE 子类参数顺序风险（已修复），需确保所有调用链更新

关联脉络

- PR #25774 drop output ids: 同样是从 ForwardBatch 中移除冗余状态的重构，与本 PR 的动机和方法论一致。