PR #23594 完整报告

sgl-project/sglang

LoRA support for qwen3.5 and nemotron3

合并时间：2026-04-30 12:51

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23594

执行摘要

- 一句话：为 Qwen3.5 和 Nemotron3 添加 LoRA 支持并修复切片 bug
- 推荐动作：值得精读。特别是 _build_moe_gated_map 的设计和 _get_lora_n_slices 的泛化方法，对理解 SGLang LoRA 框架有参考价值。ReplicatedLinearWithLoRA 的修复应关注其对现有模型的兼容性。

功能与动机

Adding missing LoRA support for Qwen3.5 and Nemotron3. In particular supporting ungated mlp/moe in lora-moe, and mamba2/GDN projections. This also fixes an issue in ReplicatedLinearWithLoRA with 2 slices (kimi/deepseek's fused_qkv_a_proj_with_mqa) causing wrong results/nans when the loaded-lora-rank<max_lora_rank

实现拆解

1. 在 LoRAAdapter 初始化中引入 base_model 引用和 _build_moe_gated_map 方法，用于检测 MoE 层是否为门控。
2. 泛化 MergedColumnParallelLinearWithLoRA 的 set_lora_info 和 apply_lora，通过 _get_lora_n_slices 动态计算实际切片数，并回退到 run_qkv_lora 支持任意切片。
3. 修复 ReplicatedLinearWithLoRA：将其 2 切片逻辑从两次独立 sgemm 改为 run_qkv_lora，消除当 loaded-rank < max-rank 时的内存布局错误。
4. 在 Qwen3_5ForCausalLM 和 NemotronHForCausalLM 中添加 supported_lora_modules、get_hidden_dim、get_stacked_multiply 等方法，为 LoRA 分配器提供维度信息。
5. 扩展 lora/utils.py 的 get_stacked_multiply；修改 lora_moe_runners.py 自动支持非门控 MoE。
6. 新增三个端到端回归测试和 chunked SGMV 4 切片测试。

关键文件：

- python/sglang/srt/lora/lora.py（模块 LoRA 核心；类别 source；类型 core-logic；符号 _build_moe_gated_map, _is_non_gated_moe_weight, _normalize_in_proj, _normalize_in_proj_qkvz）：LoRA 适配器核心逻辑，支持任意切片数和非门控 MoE

- python/sglang/srt/models/nemotron_h.py（模块 模型定义；类别 source；类型 data-contract；符号 get_stacked_multiply, get_hidden_dim）：为 Nemotron3 模型添加 LoRA 配置接口（supported_lora_modules, get_hidden_dim, get_stacked_multiply）
- python/sglang/srt/models/qwen3_5.py（模块 模型定义；类别 source；类型 data-contract；符号 get_hidden_dim, should_apply_lora）：为 Qwen3.5 模型添加 LoRA 配置接口（supported_lora_modules, get_hidden_dim, should_apply_lora）
- test/registered/lora/test_lora_qwen3_5_35b_a3b_logprob_diff.py（模块 回归测试；类别 test；类型 test-coverage；符号 kl_v2, get_prompt_logprobs, TestLoRAQwen3_5_35B_A3B_LogprobDiff, test_lora_qwen3_5_35b_a3b_logprob_accuracy）：Qwen3.5-35B-A3B LoRA 回归测试，验证 logprob 精度
- test/registered/lora/test_lora_nemotron_3_super_120b_a12b_logprob_diff.py（模块 回归测试；类别 test；类型 test-coverage；符号 kl_v2, get_prompt_logprobs, TestLoRANemotron3Super120B_A12B_LogprobDiff, test_lora_nemotron_3_super_120b_a12b_logprob_accuracy）：Nemotron3-Super-120B LoRA 回归测试，验证 logprob 精度
- test/registered/lora/test_lora_qwen3_5_4b_logprob_diff.py（模块 回归测试；类别 test；类型 test-coverage；符号 kl_v2, get_prompt_logprobs, TestLoRAQwen3_5_4BLogprobDiff, test_lora_qwen3_5_4b_logprob_accuracy）：Qwen3.5-4B LoRA 回归测试，验证 logprob 精度
- python/sglang/srt/lora/layers.py（模块 LoRA 层；类别 source；类型 core-logic；符号 _get_lora_n_slices）：泛化 MergedColumnParallelLinearWithLoRA 支持任意切片数，修复 ReplicatedLinearWithLoRA
- python/sglang/srt/lora/utils.py（模块 工具函数；类别 source；类型 core-logic；符号 get_stacked_multiply）：扩展 get_stacked_multiply 以支持更多模块
- python/sglang/srt/lora/mem_pool.py（模块 缓存池；类别 source；类型 core-logic；符号 _has_moe_module）：调整 _has_moe_module 以支持新模型检测
- python/sglang/srt/lora/lora_moe_runners.py（模块 MoE 运行器；类别 source；类型 core-logic）：支持非门控 MoE 的 LoRA 权重处理
- test/registered/lora/test_chunked_sgmv_backend.py（模块 后端测试；类别 test；类型 test-coverage；符号 test_4_slice_gdn_qkvz）：新增 4 切片 GDN QKVZ 测试，验证 SGMV 后端正确性
- python/sglang/srt/model_loader/loader.py（模块 模型加载；类别 source；类型 data-contract）：轻微调整以配合 LoRA 初始化流程

关键符号：_build_moe_gated_map, _is_non_gated_moe_weight, _normalize_in_proj, _normalize_in_proj_qkvz, get_stacked_multiply, get_hidden_dim, should_apply_lora, _get_lora_n_slices, _has_moe_module

关键源码片段

python/sglang/srt/lora/lora.py

LoRA 适配器核心逻辑，支持任意切片数和非门控 MoE

```
# python/sglang/srt/lora/lora.py — 核心新增方法

class LoRAAdapter(nn.Module):

    @staticmethod
    def _build_moe_gated_map(base_model: torch.nn.Module) -> Dict[int, bool]:
        # Map layer_id -> moe_runner_config.is_gated for FusedMoE base layers.
        # 遍历 base_model 的所有命名模块，提取 FusedMoE 层（或 BaseLayerWithLoRA 内的 base_
        layer），
        # 记录每个 MoE 层是否为门控（gated）。该映射在加载 LoRA 权重时用于决定 gate_proj
        的处理方式：
        # - 门控（c=2）需要零填充堆叠；
        # - 非门控（c=1）仅重命名（通过模型的 get_stacked_multiply 返回 1）。
        from sglang.srt.layers.moe.fused_moe_triton.layer import FusedMoE

        gated_map: Dict[int, bool] = {}
        for name, module in base_model.named_modules():
            inner = (
                module
                if isinstance(module, FusedMoE)
                else getattr(module, 'base_layer', None)
            )
            if not isinstance(inner, FusedMoE):
                continue
            layer_id = get_layer_id(name)
            if layer_id is not None:
                gated_map[layer_id] = bool(inner.moe_runner_config.is_gated)
        return gated_map

    def _is_non_gated_moe_weight(self, weight_name: str) -> bool:
        # 如果权重名称匹配路由专家模式且对应层为非门控，则返回 True。
        if not _ROUTED_EXPERT_PATTERN.search(weight_name):
            return False
        layer_id = get_layer_id(weight_name)
        if layer_id is None:
            return False
        return not self._moe_is_gated_by_layer.get(layer_id, True)
```

python/sglang/srt/models/nemotron_h.py

为 Nemotron3 模型添加 LoRA 配置接口（supported_lora_modules, get_hidden_dim, get_stacked_multiply）

```
# python/sglang/srt/models/nemotron_h.py — LoRA 维度与堆叠配置

class NemotronHForCausalLM(nn.Module):
    # 新增：声明支持 LoRA 的目标模块
    supported_lora_modules = [
        'qkv_proj',
        'o_proj',
```

```
        'out_proj',
        'in_proj',
        'up_proj',
        'gate_up_proj',
        'down_proj',
        'fc1_latent_proj',
        'fc2_latent_proj',
    ]

    def get_stacked_multiply(self, module_name):
        # 非门控 MoE 对 gate_up_proj_moe 返回 stacked_multiply=1（只有 w1，没有 w3）。
        if module_name == 'gate_up_proj_moe':
            return 1 # Non-gated: only w1, no w3
        # 其他模块回退到默认值
        from sglang.srt.lora.utils import get_stacked_multiply
        return get_stacked_multiply(module_name)

    def get_hidden_dim(self, module_name, layer_idx):
        # 返回 (input_dim, output_dim)，用于 LoRA 缓冲区维度分配。
        config = self.config
        layer_type = config.layers_block_type[layer_idx]
        hidden_size = config.hidden_size
        head_dim = getattr(config, 'head_dim', hidden_size // config.num_attention_heads)

        if module_name == 'qkv_proj':
            return (
                hidden_size,
                head_dim * (config.num_attention_heads + config.num_key_value_heads * 2),
            )
        elif module_name == 'o_proj':
            return (head_dim * config.num_attention_heads, hidden_size)
        elif module_name == 'out_proj':
            mamba_intermediate = config.mamba_num_heads * config.mamba_head_dim
            return (mamba_intermediate, hidden_size)
        elif module_name == 'gate_up_proj':
            if layer_type == 'mamba':
                mamba_intermediate = config.mamba_num_heads * config.mamba_head_dim
                return (hidden_size, mamba_intermediate * 2)
            elif layer_type == 'moe':
                shared_inter = config.moe_shared_expert_intermediate_size * config.n_shared_
                experts
                return (hidden_size, shared_inter * 2)
            else:
                return (hidden_size, config.intermediate_size * 2)
        # ... 其他模块类似处理
```

python/sglang/srt/models/qwen3_5.py

为 Qwen3.5 模型添加 LoRA 配置接口（supported_lora_modules, get_hidden_dim, should_apply_lora）

```
# python/sglang/srt/models/qwen3_5.py — LoRA 配置接口

class Qwen3_5ForCausalLM(nn.Module):
    # 新增：声明支持 LoRA 的目标模块
    supported_lora_modules = [
        'qkv_proj',
        'o_proj',
        'out_proj',
        'in_proj_qkvz',
        'gate_up_proj',
        'down_proj',
        'lm_head',
    ]

    def get_hidden_dim(self, module_name: str, layer_idx: int):
        # 根据模块名称和层索引返回 (input_dim, output_dim)。
        config = self.config
        head_dim = config.head_dim or (config.hidden_size // config.num_attention_heads)

        if module_name == 'qkv_proj':
            attn_output_gate = getattr(config, 'attn_output_gate', True)
            q_heads = config.num_attention_heads * (2 if attn_output_gate else 1)
            return (
                config.hidden_size,
                head_dim * (q_heads + config.num_key_value_heads * 2),
            )
        elif module_name == 'o_proj':
            return (config.num_attention_heads * head_dim, config.hidden_size)
        elif module_name == 'out_proj':
            value_dim = config.linear_value_head_dim * config.linear_num_value_heads
            return (value_dim, config.hidden_size)
        elif module_name == 'in_proj_qkvz':
            key_dim = config.linear_key_head_dim * config.linear_num_key_heads
            value_dim = config.linear_value_head_dim * config.linear_num_value_heads
            return (config.hidden_size, key_dim * 2 + value_dim * 2)
        elif module_name == 'gate_up_proj':
            is_moe = 'moe' in getattr(config, 'model_type', '')
            inter = config.shared_expert_intermediate_size if is_moe else config.intermediate_size
            return (config.hidden_size, inter * 2)
        elif module_name == 'down_proj':
            is_moe = 'moe' in getattr(config, 'model_type', '')
            inter = config.shared_expert_intermediate_size if is_moe else config.intermediate_size
            return (inter, config.hidden_size)
        elif module_name == 'gate_up_proj_moe':
            return (config.hidden_size, config.moe_intermediate_size * 2)
        elif module_name == 'down_proj_moe':
```

```
            return (config.moe_intermediate_size, config.hidden_size)
        elif module_name == 'embed_tokens':
            return (config.vocab_size, config.hidden_size)
        elif module_name == 'lm_head':
            return (config.hidden_size, config.vocab_size)
        else:
            raise NotImplementedError(
                f'get_hidden_dim not implemented for {module_name}'
            )

class Qwen3_5ForConditionalGeneration(...):
    supported_lora_modules = Qwen3_5ForCausalLM.supported_lora_modules

    def get_hidden_dim(self, module_name: str, layer_idx: int):
        return self.model.get_hidden_dim(module_name, layer_idx)

    def should_apply_lora(self, module_name: str) -> bool:
        # 仅对 model.layers 下的模块应用 LoRA。
        return module_name.startswith('model.layers.')
```

评论区精华

本 PR 没有公开的 review 评论，两位维护者 yushengsu-thu 和 hnyls2002 直接 approve。提交历史显示 jybsuper 在初始实现后补充了 gate-only adapters on gated gate_up_proj 支持和 pre-merged in_proj_qkvz 支持（commit 2e07351），并修复了 CI（commit 13192d8）。

- 额外功能添加与 CI 修复 (other): 已合并到最终 PR

风险与影响

- 风险：核心风险在于对 ReplicatedLinearWithLoRA 的布局假设更改，可能影响 kimi/deepseek 等模型（使用 fused_qkv_a_proj_with_mqa）的现有 LoRA 工作流。另一风险是新的模型特定方法（get_hidden_dim、get_stacked_multiply）未实现时可能引发 NotImplementedError，测试已覆盖主要路径但动态加载可能遗漏。性能方面无显著风险，因主要逻辑仅在初始化和加载时执行。
- 影响：用户：Qwen3.5 和 Nemotron3 用户可以部署 LoRA 适配器；原有 LoRA 用户受益于 ReplicatedLinearWithLoRA 修复。系统：LoRA 系统更加通用，支持任意切片数，为未来模型提供基础。团队：需维护新模型的 LoRA 配置，但框架减少重复工作。
- 风险标记：核心路径变更（LoRA 系统），兼容性风险（ReplicatedLinearWithLoRA），多模型适配

关联脉络

- 暂无明显关联 PR