PR #27329 完整报告

sgl-project/sglang

[LoRA] Experimental fast LoRA path with `experimental_sgl_trtllm` MoE backend for FP8 and NVFP4 models

合并时间：2026-06-06 05:45

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27329

执行摘要

- 一句话：实验性快速 LoRA 路径：融合 TRT-LLM MoE + 双流重叠，FP8/NVFP4 吞吐提升 1.7x
- 推荐动作：该 PR 技术含量高，值得精读。核心设计（融合 MoE + LoRA 双流重叠、Split-K 融合、JIT 内核生成）为高性能 LoRA 推理提供了范本。Review 过程严格保障默认路径安全，是实验性功能落地的最佳实践。建议团队在后续重构中关注： 1) 将 *_temp 包合并到正式模块； 2) 补充单元测试和 CI 集成； 3) 扩展支持更多模型和硬件。

功能与动机

默认 triton LoRA 后端在 MoE 模型上吞吐仅达无 LoRA 的 ~62-65%（PR 正文：sustains only ~62–65% of the no-LoRA ceiling）。本 PR 旨在通过直接融合 LoRA 到 TRT-LLM 内核中恢复大部分差距，在 Qwen3.5 和 Kimi-K2.5 上将吞吐提升至 79-93%，并达到约 1.7× 相对默认后端的加速。

实现拆解

该实现按以下 5 个步骤构建：

1. 融合 MoE 内核封装：在 python/sglang/jit_kernel/trtllm_lora_temp/core.py 中新增 trtllm_fp8_block_scale_moe、trtllm_fp8_block_scale_routed_moe_lora 等函数。通过 monkey-patch 替换 FlashInfer 的 gen_trtllm_gen_fused_moe_sm100_module，生成包含 LoRA 融合的自定义 SM100 CUDA 内核。同时支持 NVFP4 量化（trtllm_fp4_block_scale_routed_moe_lora）。
2. 双流重叠 MoE 调度：在 python/sglang/srt/lora/trtllm_lora_temp/moe_overlap.py 中实现 fused_experts_none_to_experimental_sgl_trtllm_fp8_lora_two_stream，将 LoRA shrink（gate_up 的 A 步）放在侧 CUDA 流上与主流 FP8 量化 GEMM 并行执行，再合并 expand（B 步）。类似支持 FP4。仅对解码形状（token ≤ 256）的批次启用，其余回退原始函数。
3. 注意力层 LoRA 重叠：attention.py 提供 qkv_proj_lora_forward、row_parallel_lora_forward 等函数，对 QKV 投影和输出投影采用相同的侧流重叠模式。当批次不符合双流条件时，通过保存的原始 forward 方法回退。

4. DeepSeek MLA 吸收校正：deepseek_mla_correction.py 针对 DeepSeek 模型 absorbed-MLA 路径（kv_b_proj 被绕过）注入 LoRA 校正，通过两个 SGMM 风格 Triton 内核（step A / step B）在吸收路径上添加缺失的 LoRA 增量。
5. 环境变量与 Backend 注册：environ.py 定义 SGLANG_EXPERIMENTAL_LORA_OPTI 等系列开关；experimental_sgl_trtllm_moe.py 注册自定义 CUDA 操作；server_args.py 添加 experimental_sgl_trtllm MoE 后端枚举值。所有实验性代码置于 *_temp 包，默认路径无侵入。

配套文档在 PR body 中提供了详细启动命令和性能 Benchmark，但无端到端单元测试（文档说明 tests to follow）。

关键文件：

- python/sglang/jit_kernel/trtllm_lora_temp/core.py（模块 JIT 内核；类别 source；类型 core-logic；符号 get_sgl_trtllm_moe_sm100_module, get_sgl_trtllm_moe_sm100_raw_module, _validate_routing_replay_out, trtllm_fp8_block_scale_moe）：融合 LoRA 的 TRT-LLM MoE 核心内核封装，生成定制 SM100 CUDA 模块
- python/sglang/srt/lora/trtllm_lora_temp/moe_overlap.py（模块 MoE 重叠；类别 source；类型 core-logic；符号 fused_experts_none_to_experimental_sgl_trtllm_fp8_lora_two_stream, _run_gate_up_lora, _run_down_lora, fused_experts_none_to_experimental_sgl_trtllm_fp4_lora_two_stream）：双流重叠 MoE LoRA 调度核心实现，将 LoRA shrink 与主流 GEMM 并行执行
- python/sglang/srt/lora/trtllm_lora_temp/attention.py（模块 注意力 LoRA 重叠；类别 source；类型 core-logic；符号 qkv_proj_lora_forward, row_parallel_lora_forward, column_parallel_lora_forward, replicated_lora_forward）：实现 QKV 和输出投影层的双流重叠 LoRA，与 MoE 类似模式
- python/sglang/srt/lora/trtllm_lora_temp/deepseek_mla_correction.py（模块 MLA 校正；类别 source；类型 core-logic；符号 _ensure_step_kernels, is_kv_b_lora_active, _get_state, apply_q_correction）：处理 DeepSeek 模型 absorbed-MLA 路径的 LoRA 校正，弥补标准线性层包装无法覆盖的漏洞
- python/sglang/srt/lora/trtllm_lora_temp/__init__.py（模块 LoRA 重叠基础设施；类别 source；类型 core-logic；符号 is_two_stream_active, get_lora_side_stream, init_lora_two_stream_resources, lora_overlap_alloc_stream）：包入口，管理双流状态、侧流生命周期、原始函数保存等全局基础设施

关键符号：fused_experts_none_to_experimental_sgl_trtllm_fp8_lora_two_stream, fused_experts_none_to_experimental_sgl_trtllm_fp4_lora_two_stream, trtllm_fp8_block_scale_routed_moe_lora, trtllm_fp4_block_scale_routed_moe_lora, qkv_proj_lora_forward, row_parallel_lora_forward, apply_q_correction, apply_v_correction, init_lora_two_stream_resources, is_two_stream_active

关键源码片段

python/sglang/srt/lora/trtllm_lora_temp/moe_overlap.py

双流重叠 MoE LoRA 调度核心实现，将 LoRA shrink 与主流 GEMM 并行执行

```
def fused_experts_none_to_experimental_sgl_trtllm_fp8_lora_two_stream(
    dispatch_output,
    quant_info,
    runner_config,
    lora_info,
):
    '''双流快速路径入口。

    仅 decode 形状且 virtual-experts LoRA 时启用，否则回退原始函数。
    '''
    hidden_states = dispatch_output.hidden_states
    use_virtual_lora_store = bool(
        lora_info.lora_use_virtual_experts and lora_info.max_lora_rank > 0
    )
    # 门控：必须同时满足虚拟专家 LoRA 和解码形状
    if not (use_virtual_lora_store and is_two_stream_active(hidden_states)):
        return get_original_moe_lora_func()(
            dispatch_output, quant_info, runner_config, lora_info
        )

    side_stream = get_lora_side_stream()
    gate_up_delta = hidden_states.new_empty(gate_up_delta_shape)

    def _run_gate_up_lora():
        merged_experts_fused_moe_lora_add(
            output=gate_up_delta,
            hidden_states=hidden_states,
            lora_a=lora_info.gate_up_lora_a_weights,
            lora_b=lora_info.gate_up_lora_b_weights,
            topk_ids=topk_ids,
            topk_weights=topk_weights,
        )

    # 将 LoRA shrink 发射到侧流，与主流 FP8 GEMM 并发
    side_stream.wait_stream(torch.cuda.current_stream())
    with torch.cuda.stream(side_stream):
        _run_gate_up_lora()
    # 主流计算（与 side_stream 并发）
    # ...
    # 汇合
    torch.cuda.current_stream().wait_stream(side_stream)
    return output
```

python/sglang/srt/lora/trtllm_lora_temp/attention.py

实现 QKV 和输出投影层的双流重叠 LoRA，与 MoE 类似模式

```
def qkv_proj_lora_forward(self, input_: torch.Tensor):
    '''QKV 投影双流 LoRA forward。
```

```
    侧流执行 LoRA shrink，主流执行量化 QKV GEMM，汇合后 expand。
    '''
    if not self.set_lora or not is_two_stream_active(input_):
        return get_original_qkv_forward()(self, input_)

    from sglang.srt.lora.trtllm_lora_temp.triton_ops import (
        qkv_lora_b_fwd, sgemm_lora_a_fwd,
    )
    side_stream = get_lora_side_stream()
    sgemm_info = self.lora_backend._sgemm_info()

    _alloc = lora_overlap_alloc_stream()
    side_stream.wait_stream(torch.cuda.current_stream())
    with torch.cuda.stream(side_stream):
        shrink_intermediate = sgemm_lora_a_fwd(
            input_, self.A_buffer_qkv, sgemm_info, stack_num=3, out_alloc_stream=_alloc
        )
    # 主流：量化 QKV GEMM（与侧流 shrink 并发）
    output_parallel = self.base_layer.quant_method.apply(self.base_layer, input_, bias)

    torch.cuda.current_stream().wait_stream(side_stream)
    output_parallel = qkv_lora_b_fwd(
        shrink_intermediate, self.B_buffer_qkv, sgemm_info,
        self.output_offset, self.max_qkv_out_dim, output_parallel, n_slices=3,
    )
    # ...
```

评论区精华

- 正确性风险：fzyzcjy 在 python/sglang/srt/layers/moe/topk.py 评论指出两个 ungated 变更（StandardTopKOutput 新增字段、kimi_k2_moe_fused_gate 调用更改）即使在 SGLANG_EXPERIMENTAL_LORA_OPTI 关闭时也会改变普通 MoE 推理行为。作者在最新提交中修复，加上了环境变量护卫。
- 性能开销：fzyzcjy 在 forward_mla.py 建议将 envs.SGLANG_EXPERIMENTAL_LORA_OPTI.get() 提升为模块级常量，避免每次调用运行时开销。作者已全局导入。
- 代码风格：fzyzcjy 要求将 lora_manager.py 中的局部 import 改为全局；作者已修改。
- 设计疑问：fzyzcjy 对 FusedMoEWithLoRA.__getattr__ 表示 looks hacky，作者改为显式定义缺失属性。
- 未使用字段：LoRABatchInfo.single_adapter 被标记为 iirc this is not needed，作者随后移除。
  - topk.py 未护卫的变更破坏默认路径 (correctness): 作者在最新提交 ac51ef5e 中修复，在相关代码周围添加了环境变量检查，确保仅当实验性路径启用时才执行新逻辑。
  - forward_mla.py 运行时开销建议全局导入 (performance): 作者在每个受影响的文件中将 env 读取改为导入时求值的模块级变量。

- FusedMoEWithLoRA.getattr__设计疑虑 (design): 作者替换为显式定义缺失的属性，移除了 __getattr。
- lora_manager.py 中局部导入改为全局 (style): 作者将相关 import 移到文件顶部。

风险与影响

- 风险：
  - 默认路径回归：最初 topk.py 有 ungated 变更，虽已修复，但类似遗漏可能存在于其他上游文件。需进一步审查所有非 *_temp 文件的变更。
  - 缺少测试覆盖：PR 明确说明 tests to follow，目前无端到端单元测试或集成测试验证正确性和稳定性。
  - 硬件兼容性：新路径仅在 GB200 (sm100) 上验证，依赖 TRT-LLM SM100 特定 cubin，在其他架构上可能编译失败或性能不佳。
  - CUDA graph 兼容：双流重叠依赖在捕获前创建侧流，init_lora_two_stream_resources 必须在 capture 区域外调用，若调用时机不当可能导致 graph 捕获失败。
  - 维护成本：大量 monkey-patch 和临时包结构，后续重构时可能产生兼容性问题。
- 影响：
  - 用户影响：默认行为完全不变；需显式设置 SGLANG_EXPERIMENTAL_LORA_OPTI=1 并指定 --moe-runner-backend experimental_sgl_trtllm 才能启用快速路径。当前仅支持 FP8/NVFP4 的 MoE 模型（主要是 Qwen3.5 和 Kimi-K2.5 系列）。
  - 系统影响：添加约 +16.5K 行新代码，但全部隔离在 trtllm_lora_temp 临时包中；对现有核心文件（如 topk.py、lora_manager.py）改动很有限且已护卫。
  - 团队影响：需要后续跟进重构（将临时包合并回主路径）和补充测试，短期内增加维护负担。
  - 风险标记：默认路径回归风险，缺少测试覆盖，仅兼容 sm100，实验性代码维护成本

关联脉络

- 暂无明显关联 PR