PR #28953 完整报告

sgl-project/sglang

[LoRA] BF16 support + EP cuda-graph crash fix for experimental_sgl_trtllm MoE-LoRA

合并时间：2026-06-25 12:16

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/28953

执行摘要

- 一句话：支持 BF16 LoRA 快速路径并修复 EP+CUDA 图崩溃
- 推荐动作：建议团队关注 BF16 路径的性能差异，可考虑添加单元测试覆盖数值正确性。该 PR 的二流重叠设计决策（gate_up LoRA 并发的度、down LoRA 保持串行）值得学习，但需注意 CUDA graph 捕获期间的 event 管理。

功能与动机

根据 PR body，之前 experimental_sgl_trtllm MoE-LoRA 路径仅支持 FP8/NVFP4，BF16 checkpoint 启动时崩溃。同时 NVFP4 在启用 two-stream overlap 和 CUDA graph 时出现非法内存访问，BF16 路径也有路由参数和 CUDA 图分配的正确性问题。需要扩展 BF16 支持并修复这些稳定性问题。

实现拆解

1. 新增 BF16 CUDA 内核：在 trtllm_fused_moe_kernel_launcher.cu 中新增 Bf16LoraLauncher 类，实现无量化的 BF16 端到端 MoE-LoRA 流水线（permute -> gate_up GEMM -> LoRA 感知激活 -> down GEMM），与 FP8/FP4 路径共享 finalize 内核。
2. 新增 Python 内核封装：在 core.py 中新增 trtllm_bf16_routed_moe_lora 函数，调用 CUDA 内核并暴露参数，包括 LoRA 增量与事件处理。
3. 新增单流调度函数：在 lora_dispatch.py 中新增 fused_experts_none_to_experimental_sgl_trtllm_bf16_lora，实现 BF16 模型的单流 MoE-LoRA 调度，路由、gate_up、激活、down 全部 bf16。
4. 新增二流重叠调度函数：在 moe_overlap.py 中新增 fused_experts_none_to_experimental_sgl_trtllm_bf16_lora_two_stream，将 gate_up LoRA 收缩扩展卸载到侧流，与主流的 routing + permute + gate_up GEMM 并发，仅在虚拟专家 LoRA 且 decode 形状时触发，否则回退单流。
5. 修改初始化路径：在 lora_layer.py 中新增 BF16 分支，检测无量化配置时创建 FlashInferTrtllmBf16MoeQuantInfo 并填充 dummy w13/w2_weight 以兼容上游 CUDA graph 缓存。
6. 修复 NVFP4 CUDA graph 崩溃：在 permute 内核中 guard permutedIdx == -1（EP 下无效槽位）避免负偏移写入。

7. 修复 BF16 路由参数与分配安全性：确保 BF16 路径路由参数正确性及 CUDA graph 捕获时内存分配安全。

关键文件：

- python/sglang/srt/lora/trtllm_lora_temp/moe_overlap.py（模块 MoE-LoRA；类别 source；类型 core-logic；符号 fused_experts_none_to_experimental_sgl_trtllm_bf16_lora_two_stream）：新增 BF16 二流重叠调度函数，核心逻辑所在，实现 gate_up LoRA 收缩扩展与主流并发。
- python/sglang/srt/lora/trtllm_lora_temp/lora_dispatch.py（模块 MoE-LoRA；类别 source；类型 core-logic；符号 fused_experts_none_to_experimental_sgl_trtllm_bf16_lora）：新增 BF16 单流调度函数，作为二流版本的回退基线和单流入口。
- python/sglang/jit_kernel/trtllm_lora_temp/core.py（模块 JIT 内核；类别 source；类型 core-logic；符号 trtllm_bf16_routed_moe_lora）：新增 BF16 路由 MoE-LoRA 内核封装函数，调用 CUDA 内核，是 BF16 路径的核心数值计算入口。
- python/sglang/srt/lora/trtllm_lora_temp/lora_layer.py（模块 MoE-LoRA；类别 source；类型 dependency-wiring）：修改初始化函数，为 BF16 模型创建相应的 quant_info，并暴露 w13/w2_weight 兼容上游缓存。
- python/sglang/srt/lora/trtllm_lora_temp/__init__.py（模块 MoE-LoRA；类别 source；类型 core-logic；符号 get_original_bf16_moe_lora_func）：添加 get_original_bf16_moe_lora_func 函数和安装二流重写时保存单流 BF16 函数。
- python/sglang/jit_kernel/trtllm_lora_temp/data/csrc/trtllm_fused_moe_kernel_launcher.cu（模块 JIT 内核；类别 other；类型 core-logic；符号 Bf16LoraLauncher）：新增 BF16 MoE LoRA CUDA 内核类 Bf16LoraLauncher，实现全 BF16 流水线。

关键符号：fused_experts_none_to_experimental_sgl_trtllm_bf16_lora_two_stream, fused_experts_none_to_experimental_sgl_trtllm_bf16_lora, trtllm_bf16_routed_moe_lora, get_original_bf16_moe_lora_func, Bf16LoraLauncher

关键源码片段

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

新增 BF16 二流重叠调度函数，核心逻辑所在，实现 gate_up LoRA 收缩扩展与主流并发。

```
"""Two-stream BF16 sibling of the FP8/FP4 two-stream MoE LoRA dispatches.

O1-bf16 fork: the gate_up LoRA shrink/expand runs on the side stream
concurrent with the bf16 op's routing + permute + gate_up GEMM; the op
waits on ``lora_ready_event`` right before its activation kernel (the only
consumer of ``gate_up_delta``). Fires only for virtual-experts LoRA +
decode-shaped batches; everything else delegates to the saved-original
single-stream bf16 dispatch (byte-identical). Down-LoRA stays serial on
the main stream — the down/finalize overlap was bench-verified
net-neutral-to-negative on the FP8/FP4 paths and corrupted the base
decode path under cuda-graph replay."""

def fused_experts_none_to_experimental_sgl_trtllm_bf16_lora_two_stream(
```

```
    dispatch_output,
    quant_info,
    runner_config,
    lora_info,
):
    hidden_states = dispatch_output.hidden_states
    use_virtual_lora_store = bool(
        lora_info.lora_use_virtual_experts and lora_info.max_lora_rank > 0
    )
    # Fall back to saved single-stream if not eligible
    if not (use_virtual_lora_store and is_two_stream_active(hidden_states)):
        return get_original_bf16_moe_lora_func()(
            dispatch_output, quant_info, runner_config, lora_info
        )

    # ---- two-stream fast path ----
    # 导入所需的模块和函数
    from sglang.jit_kernel.trtllm_lora_temp import trtllm_bf16_routed_moe_lora
    from sglang.jit_kernel.trtllm_lora_temp.topk_pack import fused_pack_topk
    from sglang.srt.distributed import get_tp_group
    from sglang.srt.distributed.device_communicators.pynccl_allocator import (
        use_symmetric_memory,
    )
    from sglang.srt.layers.dp_attention import is_allocation_symmetric
    from sglang.srt.layers.moe.moe_runner.flashinfer_trtllm import (
        get_activation_type,
    )
    from sglang.srt.layers.moe.token_dispatcher.standard import StandardCombineInput
    from sglang.srt.layers.moe.topk import TopKOutputChecker
    from sglang.srt.layers.moe.utils import RoutingMethodType
    from sglang.srt.lora.trtllm_lora_temp.triton_ops import (
        merged_experts_fused_moe_lora_add,
    )

    # 验证配置
    assert (
        runner_config.activation == "silu" and runner_config.is_gated
    ), "BF16 LoRA 当前仅支持门控 SwiGLU"
    topk_output = dispatch_output.topk_output
    assert TopKOutputChecker.format_is_standard(topk_output)
    assert runner_config.top_k is not None

    topk_ids = topk_output.topk_ids
    topk_weights = topk_output.topk_weights
    token_lora_mapping = lora_info.token_lora_mapping
    fused_lora_routing_cache = {}

    inter = runner_config.intermediate_size_per_partition
    side_stream = get_lora_side_stream()
```

```
    # 计算 gate_up LoRA 收缩（侧流）
    gate_up_delta = ... # 收缩逻辑，省略具体代码
    # 主流：routing + permute + gate_up GEMM 等待 lora_ready_event
    # 详见完整实现
    return StandardCombineInput(hidden_states=output)
```

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

新增 BF16 单流调度函数，作为二流版本的回退基线和单流入口。

```
def fused_experts_none_to_experimental_sgl_trtllm_bf16_lora(
    dispatch_output: StandardDispatchOutput,
    quant_info: FlashInferTrtllmBf16MoeQuantInfo,
    runner_config: MoeRunnerConfig,
    lora_info,
) -> StandardCombineInput:
    """BF16 单流 MoE-LoRA：路由 -> 收集 -> gate_up GEMM -> LoRA 感知激活 -> down GEMM -> 最终化。
    无量化，使用与普通 BF16 路径相同的预整理权重（BlockMajorK）。"""
    from sglang.jit_kernel.trtllm_lora_temp import trtllm_bf16_routed_moe_lora
    from sglang.jit_kernel.trtllm_lora_temp.topk_pack import fused_pack_topk
    from sglang.srt.layers.moe.moe_runner.flashinfer_trtllm import (
        fused_experts_none_to_flashinfer_trtllm_bf16,
        get_activation_type,
    )
    from sglang.srt.layers.moe.token_dispatcher.standard import StandardCombineInput
    from sglang.srt.layers.moe.topk import TopKOutputChecker
    from sglang.srt.layers.moe.utils import RoutingMethodType
    from sglang.srt.lora.trtllm_lora_temp.triton_ops import (
        merged_experts_fused_moe_lora_add,
    )
    from sglang.srt.model_executor.runner_utils.capture_mode import get_is_capture_mode

    assert runner_config.activation == "silu" and runner_config.is_gated
    hidden_states = dispatch_output.hidden_states
    topk_output = dispatch_output.topk_output
    assert TopKOutputChecker.format_is_standard(topk_output)
    assert runner_config.top_k is not None

    # 无 LoRA 活跃且不在 capture 时直接走普通 BF16 路径
    if not get_is_capture_mode() and not lora_info.has_active_lora:
        return fused_experts_none_to_flashinfer_trtllm_bf16(
            dispatch_output, quant_info, runner_config, use_routed_topk=True
        )

    topk_ids = topk_output.topk_ids
    topk_weights = topk_output.topk_weights
    use_virtual_lora_store = bool(
        lora_info.lora_use_virtual_experts and lora_info.max_lora_rank > 0
    )
```

```
        assert use_virtual_lora_store, "BF16 trtllm LoRA 需要使用虚拟专家"
        token_lora_mapping = lora_info.token_lora_mapping
        fused_lora_routing_cache = {}

        inter = runner_config.intermediate_size_per_partition
        # 后续计算 gate_up LoRA 增量、激活、down LoRA 等
        # 完整实现参见实际代码
        return StandardCombineInput(hidden_states=output)
```

python/sglang/jit_kernel/trtllm_lora_temp/core.py

新增 BF16 路由 MoE-LoRA 内核封装函数，调用 CUDA 内核，是 BF16 路径的核心数值计算入口。

```
def trtllm_bf16_routed_moe_lora(
    topk_ids: torch.Tensor,
    routing_bias: Optional[torch.Tensor],
    hidden_states: torch.Tensor,
    gemm1_weights: torch.Tensor,
    gemm2_weights: torch.Tensor,
    gate_up_lora_delta: torch.Tensor,
    activation_lora_input: torch.Tensor,
    num_experts: int,
    top_k: int,
    intermediate_size: int,
    local_expert_offset: int,
    local_num_experts: int,
    routed_scaling_factor: Optional[float],
    routing_method_type: int = 0,
    do_finalize: bool = True,
    enable_pdl: Optional[bool] = None,
    output: Optional[torch.Tensor] = None,
    activation_type: Optional[int] = None,
    lora_ready_event: int = 0,
    gemm2_done_event: int = 0,
) -> Union[List[torch.Tensor], torch.Tensor]:
    """BF16 MoE-LoRA 内核封装：permute -> raw gate_up GEMM -> LoRA 感知激活 -> down
    GEMM。
    权重格式与普通 BF16 路径相同（shuffled + BlockMajorK）。
    当 do_finalize=False 时返回 (gemm2_output, expert_weights, expanded_idx_to_permuted_idx)
    供 Python 侧进行 down-LoRA 合并。"""
    from flashinfer.fused_moe.core import ActivationType
    from flashinfer.utils import device_support_pdl

    if activation_type is None:
        activation_type = ActivationType.Swiglu.value
    if enable_pdl is None:
        enable_pdl = device_support_pdl(hidden_states.device)
    if output is None:
        output = torch.empty(
```

```
            hidden_states.shape, dtype=torch.bfloat16, device=hidden_states.device
        )

    assert gate_up_lora_delta.is_contiguous()
    assert activation_lora_input.is_contiguous()

    # 调用 JIT 编译的 CUDA 内核
    result = get_sgl_trtllm_moe_sm100_raw_module().sgl_trtllm_bf16_routed_moe_lora(
        topk_ids, routing_bias, hidden_states,
        gemm1_weights, gemm2_weights,
        num_experts, top_k, intermediate_size,
        local_expert_offset, local_num_experts,
        routed_scaling_factor, routing_method_type,
        do_finalize, enable_pdl, activation_type, output,
        True, # 是否为激活传入 gate_up_lora_delta
        gate_up_lora_delta, activation_lora_input,
        lora_ready_event, gemm2_done_event,
    )

    return output if do_finalize else result
```

评论区精华

该 PR 未产生实质性 review 讨论，由 Fridge003 直接批准合并。

- 整体审查 (other): 无讨论，批准合并。

风险与影响

- 风险:
  1. 核心路径变更：MoE-LoRA 路径涉及 EP 和 CUDA graph，影响范围广，可能在其他硬件上出现兼容性问题。
  2. 缺少测试覆盖：仅有 benchmark 测试，无单元测试验证 BF16 路径的数值正确性和 EP+CUDA graph 稳定性。
  3. CUDA 内核变更：新增 Bf16LoraLauncher 内核，需要维护，且可能与其他 Triton 后端冲突。
  4. 性能风险：BF16 路径未达到与 FP8 路径相同的吞吐，在批量较小时重叠增益不明显。 - 影响：对用户：现在可以为 BF16 模型（如 Qwen3.5-35B-A3B）加载 LoRA 适配器，之前不可能。对开发者：新增约 900 行代码，集中在 trtllm_lora_temp 包内，隔离性较好，不干扰原有 FP8/FP4 路径。对系统：性能接近基线（89%），无退化。对团队：需要维护新增 CUDA 内核。 - 风险标记：核心路径变更，缺少测试覆盖，CUDA 内核变更，EP+CUDA graph 风险

关联脉络

- 暂无明显关联 PR