PR #18612 完整报告

sgl-project/sglang

[Perf][Kernel] Fuse SiLU+Mul into NVFP4 Expert Quantization for CUTLASS MoE

合并时间：2026-06-30 07:51

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/18612

执行摘要

- 一句话：融合 SiLU+Mul 与 NVFP4 量化到单个 CUDA 核，消除中间缓冲
- 推荐动作：该 PR 值得精读，尤其是 CUDA kernel 融合策略、JIT kernel 注册流程以及 MoE 后端选择逻辑。设计上在保持精度 bit 一致的同时获得性能收益，是典型的高阶优化。

功能与动机

原始实现中，GEMM1 输出后需要三步：分配中间张量、调用 silu_and_mul 激活、再调用 scaled_fp4_experts_quant 量化。这些操作可以融合为一个内核，减少显存开销和调度延迟，尤其在高并发场景下提升性能。PR 作者参考了 vllm#31832 的实现思路。

实现拆解

1. CUDA 内核扩展：在 nvfp4_expert_quant.cuh 中添加 use_silu_and_mul 参数，扩展 cvt_fp16_to_fp4 函数使其可处理输入为 gate+up 拼接（shape (m, 2*k)）。新增入口函数 silu_and_mul_scaled_fp4_experts_quant_packed_sm100a，基于 expert offsets 索引，支持非均匀 token 分布。
2. Python Custom Op 注册：在 nvfp4.py 中使用 @register_custom_op 注册 fused op，实现输入检查、输出分配、JIT kernel 调用。
3. MoE 集成：在 cutlass_moe.py 的 cutlass_moe_fp4 中将原先三步替换为单次 silu_and_mul_scaled_fp4_experts_quant_packed 调用。
4. 修复 CUTLASS 后端崩溃：在 modelopt_quant.py 和 runner.py 增加对 CUTLASS 后端的保护，避免构造 MoeRunner 时因 fused function 缺失而失败。
5. 单元测试：新增 test_silu_and_mul_scaled_fp4_experts_quant_packed.py，比较 fused 与 unfused 路径在非均匀 expert offsets 下的 bit 一致性，并采用 fp32 高精度参考避免真空通过。

关键文件：

- python/sglang/jit_kernel/nvfp4.py（模块 JIT 内核；类别 source；类型 core-logic；符号 _silu_and_mul_scaled_fp4_experts_quant_packed_custom_op, silu_and_mul_scaled_fp4_experts_quant_packed）：核心 Python 封装和 Custom Op 注册，定义了 fused kernel 的接口和 JIT 加载逻辑
- python/sglang/srt/layers/moe/cutlass_moe.py（模块 MoE 层；类别 source；类型 core-logic）：MoE 流水线集成，将原来的三步替换为 fused 调用，直接体现性能收益

- python/sglang/jit_kernel/csrc/gemm/nvfp4/nvfp4_expert_quant.cuh（模块 CUDA 内核；类别 other；类型 core-logic）：CUDA kernel 核心实现，新增 use_silu_and_mul 分支和专用入口函数
- test/registered/jit/test_silu_and_mul_scaled_fp4_experts_quant_packed.py（模块 单元测试；类别 test；类型 test-coverage；符号 _nvfp4_supported, _build_expert_offsets, _build_blockscale_offsets, dequantize_nvfp4_to_dtype）：新增单元测试，覆盖非均匀 expert offsets 下的 bit 一致性，确保融合正确性

关键符号：silu_and_mul_scaled_fp4_experts_quant_packed, _silu_and_mul_scaled_fp4_experts_quant_packed_custom_op, cutlass_moe_fp4, create_moe_runner (modelopt_quant.py), cvt_fp16_to_fp4 (CUDA)

关键源码片段

python/sglang/jit_kernel/nvfp4.py

核心 Python 封装和 Custom Op 注册，定义了 fused kernel 的接口和 JIT 加载逻辑

```
@register_custom_op(
    op_name="silu_and_mul_scaled_fp4_experts_quant_packed",
    mutates_args=["output", "output_scales"],
)
def _silu_and_mul_scaled_fp4_experts_quant_packed_custom_op(
    output: torch.Tensor,
    output_scales: torch.Tensor,
    input_tensor: torch.Tensor,
    input_global_scale: torch.Tensor,
    expert_offsets: torch.Tensor,
    blockscale_offsets: torch.Tensor,
) -> None:
    # 从 JIT 缓存中加载目标模块，调用 CUDA 实现的 packed 版本
    module = _jit_nvfp4_expert_quant_module()
    module.silu_and_mul_scaled_fp4_experts_quant_packed(
        output, output_scales, input_tensor,
        input_global_scale, expert_offsets, blockscale_offsets,
    )


@debug_kernel_api
def silu_and_mul_scaled_fp4_experts_quant_packed(
    input_tensor: torch.Tensor,
    input_global_scale: torch.Tensor,
    expert_offsets: torch.Tensor,
    blockscale_offsets: torch.Tensor,
    topk: int,
    expert_map: Optional[torch.Tensor] = None,
) -> tuple[torch.Tensor, torch.Tensor]:
    """Fused SiLU+mul then FP4 quant for packed MoE inputs (expert_offsets aware).
    Input shape: (m, 2*k) — gate+up concatenated."""
```

```
    assert input_tensor.ndim == 2, f"input.ndim must be 2, got {input_tensor.ndim}."
    if expert_map is not None:
        m, k = input_tensor.shape
        input_tensor = _shuffle_rows_torch(
            input_tensor, expert_map, (m * topk, k)
        )
    m_numtopk, k_input_doubled = input_tensor.shape
    k = k_input_doubled // 2
    max_tokens_per_expert = int(os.environ.get("MODELOPT_MAX_TOKENS_PER_EXPERT",
    65536))
    assert m_numtopk <= max_tokens_per_expert * topk, (
        f"m_numtopk {m_numtopk} exceeds max allowed {max_tokens_per_expert * topk}. "
        "Set MODELOPT_MAX_TOKENS_PER_EXPERT to increase."
    )
    scales_k = k // 16
    padded_k_in_int32 = (scales_k + 3) // 4
    output = torch.empty(m_numtopk, k // 2, device=input_tensor.device, dtype=torch.uint8)
    if padded_k_in_int32 * 4 > scales_k:
        output_scales = torch.zeros(
            max_tokens_per_expert * topk, padded_k_in_int32,
            dtype=torch.int32, device=input_tensor.device
        )
    else:
        output_scales = torch.empty(
            max_tokens_per_expert * topk, padded_k_in_int32,
            dtype=torch.int32, device=input_tensor.device
        )
    _silu_and_mul_scaled_fp4_experts_quant_packed_custom_op(
        output, output_scales, input_tensor,
        input_global_scale, expert_offsets, blockscale_offsets
    )
    output_scales = output_scales.view(torch.float8_e4m3fn)
    return output, output_scales
```

python/sglang/srt/layers/moe/cutlass_moe.py

MoE 流水线集成，将原来的三步替换为 fused 调用，直接体现性能收益

```
# 替换前:
# intermediate = torch.empty(...)
# silu_and_mul(c1, intermediate)
# int_fp4, int_blockscale = scaled_fp4_experts_quant(intermediate, ...)

# 替换后（融合）:
int_fp4, int_blockscale = silu_and_mul_scaled_fp4_experts_quant_packed(
    c1, # GEMM1 输出，shape (m*topk, 2*k)
    a2_gscale, # GEMM2 全局缩放
    params.expert_offsets,
    params.blockscale_offsets,
    num_topk,
```

)

评论区精华

主要讨论集中在测试覆盖和 CI 配置上：

- mmangkad要求为 new JIT op 添加 UT，作者随后提交了测试文件。
- mmangkad建议将 CI 注册改为 base-b-kernel-unit-1-gpu-b200 专用于 Blackwell。
- gemini-code-assist[bot]建议提取 CUDA kernel 公共代码减少重复，作者未回应。
- 作者多次 rebase 以解决与 diffusion 测试的冲突。
- 添加单元测试以验证 fused kernel 的正确性 (testing): 作者提交了 test_silu_and_mul_scaled_fp4_experts_quant_packed.py，mmangkad 再次审核后批准。
- CI 测试配置变更 (testing): 作者接受了建议并更新了 CI 注册。
- CUDA kernel 代码重复与抽取公共函数 (style): 作者未回应也未采纳，该评论未引起进一步讨论，最终 PR 合并时未进行抽取。

风险与影响

- 风险：
  1. 平台限制：新 kernel 仅 Blackwell (sm100a) 生效，其他 GPU 不触发。
  2. 精度回归：融合 kernel 必须与原始分步路径 bit 一致，UT 已覆盖非均匀场景，但可能存在未曝光的边界条件。
  3. CUTLASS 后端可用性：修复了之前不可用的崩溃，但该路径此前未经过充分测试，可能存在其他运行时问题。 - 影响：影响范围：仅 NVFP4 量化且使用 CUTLASS 后端的 MoE 模型（如 Qwen3-30B-A3B-NVFP4），在 Blackwell GPU 上获得约 5% 延迟降低和 1.4% 吞吐提升。其他后端和硬件不受影响。用户影响：使用 --moe-runner-backend cutlass 或自动选择的用户受益；此前 CUTLASS 后端不可用，修复后可用。团队影响：引入一个融合 kernel，后续维护需注意与现有量化路径的兼容性。
- 风险标记：仅 Blackwell (sm100a) 生效，CUTLASS 路径之前未测试，代码重复未重构

关联脉络

- 暂无明显关联 PR