PR #26626 完整报告

sgl-project/sglang

[perf] Fuse NVFP4 gate_up_gemm + swiglu + output FP4 quant

合并时间：2026-05-30 04:16

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26626

执行摘要

- 一句话：融合 NVFP4 gate_up_gemm+Swish+ 输出量化，K2.5 推理提升约 2%
- 推荐动作：本 PR 展示了如何利用 CUTLASS DSL 进行算子融合来优化特定硬件路径。推荐算法工程师和推理优化工程师阅读，重点关注融合策略和权重重排方案。对于不涉及 Blackwell 或 NVFP4 的开发者，可跳过源码细节。

功能与动机

提升 DeepSeek K2.5 模型在 Blackwell 上的推理性能。PR body 提供对比图，端到端输出从 487 token/s 提升至 496 token/s。

实现拆解

1. 新增融合内核：创建 python/sglang/srt/layers/quantization/nvfp4_gemm_swiglu_nvfp4_quant.py，基于 CUTLASS DSL 实现一个 persistent 内核，执行 gate_up_gemm → silu → FP4 quantize。
2. 预处理融合权重：在 modelopt_quant.py 的 process_weights_after_loading 中，当检测到 _interleave_for_swiglu_fusion 标志时：先快照原始缩放因子，然后将 up/gate 分支的权重和缩放因子交错排列，并执行 blockscale 2D swizzle，保存为 weight_swiglu_interleaved 和 weight_scale_swiglu_interleaved，并释放原始权重以节省显存。
3. 添加前向快速路径：在 deepseek_v2.py 的 DeepseekV2MLP.forward 中，当满足条件（融合开关开启、非 SWA 路径、输入未预量化）时：将输入量化为 FP4，调用融合内核，输出 FP4 结果和缩放直接传递给已标记 _accepts_prequantized_fp4 的 down_proj，跳过内部重量化。
4. 环境变量控制：在 environ.py 添加 SGLANG_ENABLE_NVFP4_GEMM_SWIGLU_FUSION（默认开启），并在 __init__ 中检查条件（SM100 支持、量化方法为 ModelOptFp4LinearMethod、CUDA Graph 禁用）设置标志。
5. 配置调整：修改 .codespellrc 添加新词汇避免拼写检查失败。

关键文件：

- python/sglang/srt/layers/quantization/nvfp4_gemm_swiglu_nvfp4_quant.py（模块 量化层；类别 source；类型 core-logic）：新增融合内核实现，基于 CUTLASS DSL 提供 gate_up_gemm+silu+FP4 量化融合操作，是性能提升的核心。

- python/sglang/srt/layers/quantization/modelopt_quant.py（模块 量化层；类别 source；类型 data-contract）：修改 process_weights_after_loading 和 apply 方法，为融合路径预处理权重和缩放因子，并支持接受预量化输入。
- python/sglang/srt/models/deepseek_v2.py（模块 模型层；类别 source；类型 data-contract）：在 DeepseekV2MLP 中添加融合路径的 init 标志和 forward 快速分支。
- python/sglang/srt/environ.py（模块 配置层；类别 source；类型 configuration）：添加 SGLANG_ENABLE_NVFP4_GEMM_SWIGLU_FUSION 环境变量控制融合开关。
- .codespellrc（模块 拼写检查；类别 other；类型 configuration）：添加新词避免拼写检查失败。

关键符号：process_weights_after_loading, apply, DeepseekV2MLP.init, DeepseekV2MLP.forward

关键源码片段

python/sglang/srt/layers/quantization/modelopt_quant.py

修改 process_weights_after_loading 和 apply 方法，为融合路径预处理权重和缩放因子，并支持接受预量化输入。

```
# 在 process_weights_after_loading 中，常规 padding 和 interleaving 之后（
# padded_scales 已通过 alias_or_bind_derived_param 保存为 weight_scale_interleaved），
# 如果 layer 标记了 _interleave_for_swiglu_fusion，则额外构造融合所需的交错权重

# 快照原始 scale（在 swizzle 之前），因为 alias_or_bind_derived_param 会原地覆盖
raw_scale_snapshot = (
    (scales.squeeze(0) if scale_ndim == 2 else scales).detach().clone()
)

# 标准 blockscale 交错（原有代码）...
# [ 省略常规 padding 和 intleave 部分，与先前相同 ]

if getattr(layer, "_interleave_for_swiglu_fusion", False):
    from sglang.srt.layers.quantization.nvfp4_gemm_swiglu_nvfp4_quant import (
        interleave_linear_and_gate,
        swizzle_blockscale_2d,
    )

    w = layer.weight.data
    # 当前融合路径要求无 K-padding 和无 N-padding
    assert weights_padding_cols == 0
    assert raw_scale_snapshot.shape[0] == w.shape[0]
    assert w.shape[0] % 128 == 0 # group_size=64，gate 和 up 各占一半

    # 将权重沿 N 维度切分为 gate 和 up，然后按 [up, gate] 顺序 concat 并交错行
    gate_w, up_w = w.chunk(2, dim=0)
    w_swiglu = interleave_linear_and_gate(
        torch.cat((up_w, gate_w), dim=0), group_size=64, dim=0
```

```
    )

    # 对 scale 做同样处理（使用未 swizzle 的快照）
    gate_s, up_s = raw_scale_snapshot.chunk(2, dim=0)
    w_scale_swiglu = swizzle_blockscale_2d(
        interleave_linear_and_gate(
            torch.cat((up_s, gate_s), dim=0), group_size=64, dim=0
        )
    )

    # 保存交错后的 tensor 作为属性
    layer.weight_swiglu_interleaved = w_swiglu
    layer.weight_scale_swiglu_interleaved = w_scale_swiglu

    # 释放原始权重和 scale 的存储，但保留 Parameter 对象以支持重载
    layer.weight.data = torch.empty(0, dtype=layer.weight.dtype, device=layer.weight.device)
    layer.weight_scale_interleaved.data = torch.empty(
        0, dtype=layer.weight_scale_interleaved.dtype, device=layer.weight_scale_interleaved.
        device
    )
```

python/sglang/srt/models/deepseek_v2.py

在 DeepseekV2MLP 中添加融合路径的 init 标志和 forward 快速分支。

```
  # DeepseekV2MLP.forward 中新增的快速路径：当满足条件时，
  # 将输入量化、调用融合内核、直接传预量化结果给 down_proj

  if (
      getattr(self, "_enable_nvfp4_gemm_swiglu_fusion", False)
      and self.swiglu_limit is None # 不使用 SWA 路径
      and not isinstance(x, tuple) # 输入尚未预量化
  ):
      # 延迟导入避免循环依赖
      from flashinfer import fp4_quantize
      from sglang.srt.layers.quantization.nvfp4_gemm_swiglu_nvfp4_quant import (
          nvfp4_gemm_swiglu_nvfp4_quant,
      )

      # 将 BF16/FP16 输入量化为 FP4（复用 gate_up_proj 的 input_scale_inv）
      x_fp4, x_scale = fp4_quantize(
          x, self.gate_up_proj.input_scale_inv, enable_pdl=True
      )

      # 调用融合内核：gate_up_gemm → silu → FP4 量化输出
      out_fp4, out_scale = nvfp4_gemm_swiglu_nvfp4_quant(
          x_fp4,
          x_scale,
          self.gate_up_proj.weight_swiglu_interleaved,
          self.gate_up_proj.weight_scale_swiglu_interleaved,
```

```
        self.gate_up_proj.alpha,
        self.down_proj.input_scale_inv,
        enable_pdl=True,
    )

    # 将预量化的 FP4 结果传给 down_proj（它已设置 _accepts_prequantized_fp4）
    out, _ = self.down_proj(
        (out_fp4, out_scale),
        skip_all_reduce=should_allreduce_fusion or use_reduce_scatter,
    )
    return out
```

评论区精华

Reviewer b8zhong 表示 LGTM，但指出 DSL 代码是移植的未详细审查，并询问是否存在某些 token 阈值可能导致性能回退。作者回应未来可测试。

- Token 阈值对融合内核性能的影响 (performance): 作者接受，未来可测试。

风险与影响

- 风险：
  - 新增融合内核仅在 NVIDIA Blackwell (SM100) 上支持，通过 is_sm100_supported() 保护，但在其他架构上误启用可能 Crush。
  - 权重预处理释放原始权重，一旦运行模型则无法回退到非融合路径（需重新加载权重）。
  - 融合快速路径与 SWA 路径互斥（通过 swiglu_limit is None 条件），若用户同时启用 SWA 则融合不生效，逻辑正确但可能混淆用户。
  - .codespellrc 修改可能遗漏其他拼写错误。
- 影响：
  - 用户：仅在使用 NVFP4 量化的 DeepSeek 模型且运行在 Blackwell 硬件上的用户受益，BS=1 吞吐提升约 2%。其他用户无影响。
  - 系统：模型加载时额外执行权重交错预处理，但推理时融合路径减少 DRAM 访问，延迟更低。
  - 团队：需维护一个约 3000 行的 CUTLASS DSL 内核，但该内核基于官方示例，风险可控。
  - 风险标记：Blackwell 专属，权重预处理不可逆，未测试 Token 阈值

关联脉络

- 暂无明显关联 PR