PR #37075 完整报告

sgl-project/sglang

[Diffusion][Kernel] Fuse Wan2.2 NVFP4 bias + GELU on Blackwell

合并时间：2026-08-31 01:37

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/37075

执行摘要

- 一句话：融合 Wan2.2 NVFP4 bias+GELU，GB300 去噪提速约 3%
- 推荐动作：值得精读。重点关注三点：一是 bias_gelu_tanh_kernel 如何通过 PDL 与 occupancy 限制在 memory-bound 场景下取得近 3 倍微基准加速；二是 nvfp4_bias_gelu_site.py 如何复用 QualityGatedFusion 实现请求级门控，避免影响 lossless 路径；三是测试设计，用 torch.equal 做 bit-exact 断言并覆盖非法 width/dtype 的拒绝行为，这种精度验证方式可用于其他融合 kernel。

功能与动机

Wan2.2 ModelOpt NVFP4 MLP 的 fc_in 在 GB300 上以「NVFP4 GEMM -> BF16 bias add -> tanh GELU」三步执行，后两步是 memory-bound 操作，占去噪时间相当比例。PR body 明确目标是融合后两步，同时保持 eager 半精度舍入边界（bias 加法后先舍入回 BF16/FP16 再做 GELU），使默认 quality=lossless 路径完全不变，只在 quality=high 下为合格 NVFP4 站点启用加速。

实现拆解

该 PR 的实现按以下步骤展开：

1. 新增 JIT CUDA 融合内核：在 python/sglang/kernels/jit/csrc/elementwise/bias_gelu.cuh 中实现 bias_gelu_tanh_kernel，按向量化宽度（Blackwell 上 32 字节，其余 16 字节）逐元素执行「FP32 相加 -> 舍入回原 dtype -> tanh GELU」，保证与 eager F.gelu(input + bias, approximate="tanh") 的舍入边界一致；并支持 occupancy 上限的 persistent launch 几何与 PDL wait/trigger。
2. JIT 封装与参数校验：新增 python/sglang/kernels/ops/elementwise/bias_gelu.py，用 cache_once 缓存按 dtype 编译的 TVM 模块，仅接受 FP16/BF16；对外暴露 bias_gelu_tanh(input, bias)，内部用注册的 custom op _bias_gelu_tanh 写入预先分配的 output。
3. 请求级融合站点：新增 python/sglang/kernels/ops/diffusion/sites/nvfp4_bias_gelu_site.py，复用现有 QualityGatedFusion 基础设施，提供 mark_nvfp4_bias_gelu_site、mount_nvfp4_bias_gelu、unmount_nvfp4_bias_gelu、nvfp4_bias_gelu_active。mount 时校验站点满足 fuse_bias_gelu_tanh 且 fc_in 是 ModelOptFp4LinearMethod 且带 bias，然后把 fc_in.skip_bias_add 置为 True，使 GEMM 返回延迟 bias。

4. MLP 激活分发改造：在 python/sglang/multimodal_gen/runtime/layers/mlp.py 中新增 fuse_bias_gelu_tanh 构造参数和 _apply_activation(x, bias, use_fused_bias_gelu) 方法。forward 改为接收 fc_in 返回的延迟 bias，在满足 CUDA、FP16/BF16 且门控激活时调用 bias_gelu_tanh，否则回退到 self.act(x + bias)。
5. Wan 模型接入与门控注册：在 python/sglang/multimodal_gen/runtime/models/dits/wanvideo.py 的 _WanGELUMLP 中根据 fc_in.quant_method 是否为 ModelOptFp4LinearMethod 决定是否标记 NVFP4 融合站点；在 python/sglang/multimodal_gen/runtime/pipelines_core/stages/denoising.py 的 _QUALITY_FUSION_HANDLERS 中注册 mount/unmount 回调，使 quality=high 生命周期自动启用与恢复。
6. 测试与 Benchmark 配套：新增 test/registered/kernels/ops/elementwise/test_bias_gelu.py（bit-exact、非法宽度 / 类型拒绝）、test/registered/kernels/benchmark/elementwise/bench_bias_gelu.py（JIT vs PyTorch 对比），扩展 python/sglang/multimodal_gen/test/unit/test_wan_gelu_mlp.py 覆盖 NVFP4 站点的 mount/unmount 与 *excluded linear* 回归。

关键文件：

- python/sglang/kernels/jit/csrc/elementwise/bias_gelu.cuh（模块 融合核函数；类别 other；类型 dependency-wiring；符号 gelu_tanh, bias_gelu_tanh_kernel, bias_gelu_tanh）：融合 kernel 的核心实现，包含向量化、PDL 支持与半精度舍入边界控制，是整个 PR 的性能来源。
- python/sglang/kernels/ops/diffusion/sites/nvfp4_bias_gelu_site.py（模块 融合门控；类别 infra；类型 infrastructure；符号 mark_nvfp4_bias_gelu_site, nvfp4_bias_gelu_active, _site_reject_reason, mount_nvfp4_bias_gelu）：请求级融合门控的实现，决定哪些站点、何时启用融合，并负责 skip_bias_add 的挂载与恢复。
- python/sglang/multimodal_gen/runtime/layers/mlp.py（模块 激活层；类别 source；类型 core-logic；符号 _apply_activation）：MLP 前向逻辑的改动点，新增 _apply_activation 统一分发 *eager* 与融合路径，是模型侧唯一的行为入口。
- python/sglang/multimodal_gen/runtime/models/dits/wanvideo.py（模块 Wan 模型；类别 source；类型 data-contract；符号 _WanGELUMLP）：_WanGELUMLP 根据量化方法决定是否标记 NVFP4 站点，并接入 forward 分发。
- python/sglang/kernels/ops/elementwise/bias_gelu.py（模块 JIT 封装；类别 infra；类型 infrastructure；符号 _jit_bias_gelu_tanh_module, _bias_gelu_tanh, bias_gelu_tanh）：JIT 模块封装：按 dtype 缓存编译结果，负责类型校验与 custom op 注册。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/denoising.py（模块 去噪编排；类别 source；类型 core-logic）：将新增门控注册进质量融合处理器列表，使 quality=high 生命周期自动 mount/unmount。
- test/registered/kernels/ops/elementwise/test_bias_gelu.py（模块 核函数测试；类别 test；类型 test-coverage；符号 test_bias_gelu_tanh_is_bit_exact, test_bias_gelu_tanh_rejects_unsupported_width, test_bias_gelu_tanh_rejects_unsupported_dtype）：核心 kernel 的 bit-exact 与拒绝路径测试，覆盖 FP16/BF16 与生产 shape。

- python/sglang/multimodal_gen/test/unit/test_wan_gelu_mlp.py（模块 集成测试；类别 test；类型 test-coverage；符号 test_wan_nvfp4_mlp_defers_bias_for_gelu_fusion, test_wan_nvfp4_mlp_does_not_mark_excluded_linear）：验证 NVFP4 站点挂载 / 卸载与 excluded linear 不标记的集成行为。
- test/registered/kernels/benchmark/elementwise/bench_bias_gelu.py（模块 性能基准；类别 test；类型 test-coverage；符号 torch_bias_gelu, benchmark）：提供 JIT 融合 kernel 与 PyTorch 基线在同一 shape 下的性能对比数据。
- python/sglang/kernels/ops/diffusion/__init__.py（模块 导出配置；类别 infra；类型 infrastructure）：导出新站点符号，供 diffusion 运行时统一使用。

关键符号：bias_gelu_tanh_kernel, gelu_tanh, bias_gelu_tanh, _jit_bias_gelu_tanh_module, _bias_gelu_tanh, _apply_activation, mark_nvfp4_bias_gelu_site, nvfp4_bias_gelu_active, mount_nvfp4_bias_gelu, unmount_nvfp4_bias_gelu, _WanGELUMLP.forward

关键源码片段

python/sglang/kernels/jit/csrc/elementwise/bias_gelu.cuh

融合 kernel 的核心实现，包含向量化、PDL 支持与半精度舍入边界控制，是整个 PR 的性能来源。

```
// 融合 bias 加法与 tanh GELU 的 CUDA kernel 核心片段
// 关键设计：bias 加法后立即舍回原 dtype，再转 FP32 计算 GELU，
// 以复刻 eager F.gelu(input + bias, approximate="tanh") 的舍入边界。

template <typename T, int kVecN, bool kUsePDL>
__global__ void bias_gelu_tanh_kernel(
    const T* __restrict__ input,
    const T* __restrict__ bias,
    T* __restrict__ output,
    int64_t num_vecs,
    int64_t row_vecs) {
  using vec_t = device::AlignedVector<T, kVecN>;

  device::PDLWaitPrimary<kUsePDL>();
  const int64_t stride = static_cast<int64_t>(blockDim.x) * gridDim.x;

  // 每个线程负责多个向量，stride 循环覆盖整个张量
  for (int64_t vec_id = static_cast<int64_t>(blockIdx.x) * blockDim.x + threadIdx.x;
       vec_id < num_vecs; vec_id += stride) {
    vec_t x;
    vec_t b;
    x.load(input, vec_id);
    b.load(bias, vec_id % row_vecs);

    vec_t result;
#pragma unroll
```

```
    for (int i = 0; i < kVecN; ++i) {
      const float x_f32 = device::cast<fp32_t>(x[i]);
      const float bias_f32 = device::cast<fp32_t>(b[i]);
      const T biased = device::cast<T>(x_f32 + bias_f32); // 半精度舍入边界
      result[i] = device::cast<T>(gelu_tanh(device::cast<fp32_t>(biased)));
    }
    result.store(output, vec_id);
  }
  device::PDLTriggerSecondary<kUsePDL>();
}

// 宿主编排：校验输入并计算 occupancy 上限的 grid 尺寸
// grid = min(num_sms * occupancy, ceil(num_vecs / block_size))
// 避免过度占用，同时保证小块也能被填满。
template <typename T, bool kUsePDL>
void bias_gelu_tanh(tvm::ffi::TensorView input, tvm::ffi::TensorView bias,
                    tvm::ffi::TensorView output) {
  using namespace host;

  auto num_rows = SymbolicSize{"num_rows"};
  auto hidden_dim = SymbolicSize{"hidden_dim"};
  auto device_ = SymbolicDevice{};
  device_.set_options<kDLCUDA>();

  TensorMatcher({num_rows, hidden_dim}).with_dtype<T>().with_device(device_).verify(input);
  TensorMatcher({hidden_dim}).with_dtype<T>().with_device(device_).verify(bias);
  TensorMatcher({num_rows, hidden_dim}).with_dtype<T>().with_device(device_).verify(output);

  constexpr int kVecN = device::kMaxVecBytes / sizeof(T);
  const int64_t rows = num_rows.unwrap();
  const int64_t width = hidden_dim.unwrap();
  CHECK_HOST(rows > 0) << "bias_gelu_tanh: num_rows must be positive";
  CHECK_HOST(width > 0 && width % kVecN == 0)
      << "bias_gelu_tanh: hidden_dim must be positive and divisible by " << kVecN;

  const int64_t row_vecs = width / kVecN;
  const int64_t num_vecs = rows * row_vecs;
  constexpr int64_t kBlockSize = 256;
  const auto kernel = bias_gelu_tanh_kernel<T, kVecN, kUsePDL>;
  const int64_t occupancy = runtime::get_blocks_per_sm(kernel, kBlockSize);
  const int64_t num_sms = runtime::get_sm_count(device_.unwrap().device_id);
  const int64_t grid = std::min(num_sms * occupancy, div_ceil(num_vecs, kBlockSize));

  LaunchKernel(grid, kBlockSize, device_.unwrap())
      .enable_pdl(kUsePDL)(kernel,
                           static_cast<const T*>(input.data_ptr()),
                           static_cast<const T*>(bias.data_ptr()),
                           static_cast<T*>(output.data_ptr()),
                           num_vecs, row_vecs);
```

```
}
```

python/sglang/kernels/ops/diffusion/sites/nvfp4_bias_gelu_site.py

请求级融合门控的实现，决定哪些站点、何时启用融合，并负责 skip_bias_add 的挂载与恢复。

```
"""请求级 Wan NVFP4 bias+GELU 融合站点。"""
from __future__ import annotations

import logging

import torch.nn as nn

from sglang.kernels.ops.diffusion.sites.quality_gate import QualityGatedFusion

logger = logging.getLogger(__name__)

# 复用通用 QualityGatedFusion，以 marker/enabled 属性区分不同融合站点
_FUSION = QualityGatedFusion(
    name="Wan NVFP4 bias+GELU",
    marker_attr="_sgl_nvfp4_bias_gelu_site",
    enabled_attr="_sgl_nvfp4_bias_gelu_enabled",
)


def mark_nvfp4_bias_gelu_site(module: nn.Module) -> None:
    """将 module 标记为可延迟 fc_in bias 的融合站点。"""
    _FUSION.mark(module)


def nvfp4_bias_gelu_active(module: nn.Module) -> bool:
    """查询该站点当前是否处于挂载（激活）状态。"""
    return _FUSION.is_enabled(module)


def _site_reject_reason(site: nn.Module) -> str | None:
    """返回拒绝挂载的原因；返回 None 表示可挂载。"""
    if not getattr(site, "fuse_bias_gelu_tanh", False):
        return "site is not an NVFP4 fused-GELU target"
    linear = getattr(site, "fc_in", None)
    if linear is None:
        return "missing fc_in"

    from sglang.multimodal_gen.runtime.layers.quantization.modelopt_quant import (
        ModelOptFp4LinearMethod,
    )

    if not isinstance(getattr(linear, "quant_method", None), ModelOptFp4LinearMethod):
        return "fc_in is not ModelOpt NVFP4"
    if getattr(linear, "bias", None) is None:
```

```
            return "fc_in has no bias"
        return None


    def mount_nvfp4_bias_gelu(root: nn.Module) -> bool:
        """挂载所有合格站点：把 fc_in 的 bias 延迟到激活阶段。"""
        sites = list(_FUSION.iter_sites(root))
        mounted = _FUSION.mount(root, reject_reason=_site_reject_reason, logger=logger)
        for site in sites:
            site.fc_in.skip_bias_add = mounted # 让 GEMM 返回 (output, bias)
        return mounted


    def unmount_nvfp4_bias_gelu(root: nn.Module) -> None:
        """卸载并恢复所有标记站点的原始 eager 路径。"""
        _FUSION.unmount(root)
        for site in _FUSION.iter_sites(root):
            site.fc_in.skip_bias_add = False
```

python/sglang/multimodal_gen/runtime/layers/mlp.py

MLP 前向逻辑的改动点，新增 _apply_activation 统一分发 eager 与融合路径，是模型侧唯一的行为入口。

```
    class MLP(nn.Module):
        """DiT 块的 MLP，无门控线性单元。"""

        def __init__(self, input_dim, mlp_hidden_dim, output_dim=None, bias=True,
                     act_type="gelu_pytorch_tanh", dtype=None, prefix="",
                     quant_config=None, fuse_bias_gelu_tanh=False):
            super().__init__()
            self.fuse_bias_gelu_tanh = fuse_bias_gelu_tanh
            self.fc_in = ColumnParallelLinear(
                input_dim, mlp_hidden_dim, bias=True,
                skip_bias_add=False, # 默认 eager 路径；挂载融合门控时置 True
                gather_output=False, quant_config=quant_config,
                prefix=add_prefix("fc_in", prefix),
            )
            self.act = get_act_fn(act_type)
            if output_dim is None:
                output_dim = input_dim
            self.fc_out = RowParallelLinear(
                mlp_hidden_dim, output_dim, bias=True,
                input_is_parallel=True, quant_config=quant_config,
                prefix=add_prefix("fc_out", prefix),
            )

        def _apply_activation(self, x, bias, *, use_fused_bias_gelu=False):
            """统一激活入口：优先走 JIT 融合 kernel，否则回退 eager。"""
            if self.fuse_bias_gelu_tanh and bias is not None:
```

```
            if (use_fused_bias_gelu and x.is_cuda
                    and x.dtype in (torch.float16, torch.bfloat16)):
                from sglang.kernels.ops.elementwise.bias_gelu import bias_gelu_tanh
                return bias_gelu_tanh(x, bias)
            return self.act(x + bias)
        return self.act(x)

    def forward(self, x):
        x, bias = self.fc_in(x) # 挂载融合门控时 bias 非 None
        x = self._apply_activation(x, bias)
        x, _ = self.fc_out(x)
        return x
```

评论区精华

该 PR 没有实质 review 讨论线程，只有两条作者触发的 CI 命令评论（/tag-and-rerun-ci 与 /tag-run-ci-label extra）。技术上的关键决策都体现在 PR body 中：作者用 MP4 SHA256 完全一致、81 帧 SSIM=1.0、PSNR=inf、LPIPS=0 证明了融合路径与 baseline 视觉无损；同时给出 quality=high 与 lossless 的固有差异（SSIM 0.9866）作为已知边界。

- 暂无高价值评论线程

风险与影响

- 风险：主要风险集中在以下几点：

1. 内核适用性限制：bias_gelu_tanh 要求 hidden_dim 能被向量宽度整除（FP16 下 16 或 32 字节对应 8/16 个元素），且仅支持 FP16/BF16。不满足时 bias_gelu_tanh 直接抛错，但调用方 _apply_activation 在 use_fused_bias_gelu=False 或 dtype 不符合时会优雅回退 eager，因此只有 Blackwell 上 NVFP4 的合格站点才会走融合路径。
2. 与既有 cublasLt GELU epilogue 的交互：_WanGELUMLP.forward 中 fused_gelu_active 优先于 NVFP4 融合。当 fused_linear_gelu 挂载时走 fused_linear_gelu_tanh，此时 fc_in.skip_bias_add 也会被 NVFP4 门控设为 True，但由于 epilogue 直接消费 weight+bias，不受 skip_bias_add 影响。两个门控同时挂载时的行为需要后续留意，现有单元测试覆盖了常见顺序。
3. AMD ROCm CI 失败：PR 的 ROCm 7.2 测试结果为失败，PR body 未说明原因。该 kernel 是 CUDA JIT，可能只注册了 CUDA CI；需确认是否会影响 AMD 平台回归。
4. bit-exact 依赖 FP32 中间计算：内核先将 x 和 bias 各自转 FP32 相加，再舍入回原 dtype，最后转 FP32 算 GELU。该顺序必须与 eager PyTorch 行为严格一致，否则会引入精度漂移；测试已覆盖主要 shape，但极端数值（如 NaN/Inf）未覆盖。 - 影响：影响范围集中在 diffusion 推理路径：Wan2.2 NVFP4 模型在 Blackwell（GB300/B200）上、quality=high 模式下，fc_in 的 bias+GELU 被融合，端到端延迟降低约 2.95%（去噪约 2.98%），峰值显存不变。默认 quality=lossless 路径与未量化模型完全不受影响。对团队而言，该 PR 展示并复用了 quality-gated fusion 的模式，为后续更多融合内核（如 Qwen-Image 的 bias 吸收）提供了可参考的 mount/unmount 与 bit-exact 验证范式。 - 风险标记：AMD ROCm CI 失败，kernel 仅支持 16/32 字节对齐维度，与 cublasLt

GELU epilogue 门控交互需关注，bit-exact 依赖 FP32 中间计算顺序

关联脉络

- PR #37116 [diffusion] perf: absorb Qwen-Image output projection biases: 同为 diffusion 性能优化 PR，将 bias 延迟 / 吸收到后续融合算子中，与本 PR 的 bias 延迟思路一致，可对比不同融合策略。
- PR #36991 [Diffusion] Add exact component precision overrides: 涉及 diffusion 组件精度与 quality 生命周期管理，与本 PR 的 quality=high 门控机制有共同基础。
- PR #36907 [Diffusion] Enforce component attention backend application: 同为 diffusion 组件级开关的强制应用逻辑，与 quality gate 的 mount/unmount 模式相关。