PR #37116 完整报告

sgl-project/sglang

[diffusion] perf: absorb Qwen-Image output projection biases

合并时间：2026-08-31 08:25

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/37116

执行摘要

- 一句话：Qwen-Image 输出 bias 延迟融合，GB300 端到端提速约 6%
- 推荐动作：值得精读。三个设计点尤其值得关注：其一，能力门控 + 静默回退的窄 fast path 模式（_can_defer_modelopt_output_bias 将量化类型、算力、dtype、形状约束集中一处，回退路径与原实现语义完全一致）；其二，用原生 BF16 __hfma 保持舍入点语义的细节，这是 pin 到生产 MulAdd 逐位一致的工程狠活；其三，PR body 用 profiling 数据驱动优化决策（673 次 launch → 224 次可消除），度量方法本身可复用。建议阅读时把 qwen_image.py 的 _bias_mul_add/_modulate 与 norm_scale_shift.cuh 的 bias_mul_add_kernel 对照看，能完整理解融合路径与回退路径的切换逻辑。

功能与动机

PR body 明确指出性能瓶颈来源：Qwen-Image 的 ModelOpt FP8/NVFP4 路径在注意力与 FFN 残差更新之前会先 materialize 输出投影 bias。在 GB300 上 profile 发现每个去噪步启动 673 个 BF16 add kernel，其中 224 个启动可以在 bias 被后续残差操作消费后消失。该优化参考 Baseten 的 "Agentic Kernels in Production" 博客中描述的 Qwen-Image bias-absorption 方案，本质是把零散的 x + bias 小 kernel 合并进必做的残差 /LayerNorm kernel，减少 kernel 启动次数与全局内存往返。

实现拆解

整个实现分为 5 个步骤：

1. 能力门控与 skip_bias_add 契约（python/sglang/multimodal_gen/runtime/models/dits/qwen_image.py） - 新增 _can_defer_modelopt_output_bias(quant_config, capability) 与 _defer_modelopt_output_bias(quant_config)：仅当量化名属于 {"modelopt_fp8", "modelopt_fp4"} 且设备 capability 为 (10, 3) 时返回 True。代码注释明确说明，延迟 bias 会移动 BF16 舍入点，端到端图像质量只在 SM103 上验证过。 - QwenImageCrossAttention.__init__ 计算 self.defer_output_bias，并将其作为 skip_bias_add 传入 to_out.0 与 to_add_out 两个 RowParallelLinear；QwenImageFeedForward.__init__ 对 FFN 的 net.2 同样传入。skip_bias_add=True 后 GEMM 不再在 epilogue 加 bias，而是返回 (output, bias) 二元组。
2. forward 签名扩展（同文件） - QwenImageCrossAttention.forward 从返回 (img_attn_output, txt_attn_output) 扩展为 (img_attn_output, txt_attn_output, img_attn_bias, txt_attn_bias) 四元组。 - QwenImageFeedForward 新增

forward_with_bias 返回 (hidden_states, bias)；原 forward 在 bias 非空时补加 hidden_states + bias，保证非融合路径语义不变。

3. JIT 内核扩展（python/sglang/kernels/ops/diffusion/norm/norm_scale_shift_jit.py + python/sglang/kernels/jit/csrc/diffusion/norm_scale_shift.cuh） - 新增 try_fused_bias_scale_residual_norm_scale_shift：把 x + bias、门控残差、LayerNorm、scale/shift一次性融合，约束为norm_type=="layer"、无affine参数、BF16、SM10.3、行向量广播。 - 新增 try_fused_bias_mul_add：实现 y = (x + bias) * gate + residual，对应最终残差。 - CUDA 侧 NormScaleShiftParams 增加 input_bias 字段，norm_scale_shift_kernel 模板增加 kHasInputBias 编译期分支；新增 BiasMulAddKernel/bias_mul_add_kernel。
4. TransformerBlock 前向整合（同文件） - 图片流 norm2 的 _modulate 新增 x_bias 参数：优先调用融合 norm kernel，失败则 x = x + x_bias 后走原 _scale_residual_norm_scale_shift；FFN 改用 forward_with_bias 拿到 img_mlp_bias，经新辅助方法 _bias_mul_add 完成最终残差（先试 try_fused_bias_mul_add，失败则 a + bias 后走 fuse_mul_add）。 - 文本流非 BCG 路径把 txt_attn_bias 传入 _modulate，BCG 路径手动 txt_attn_output + txt_attn_bias。Nunchaku、非 ModelOpt、torch.compile/BCG 路径全部保持原实现。
5. 测试与导出配套 - 新增 test/registered/kernels/ops/diffusion/test_qwen_output_bias_absorption.py：覆盖 SM 10.3-only 门控（含 None capability 与 (10, 0)/(12, 0) 反例）、对 ScaleResidualLayerNormScaleShift/MulAdd 的 bit-exact 对比、unsupported shape 与 torch.compiler.is_compiling() 下的回退拒绝。 - python/sglang/kernels/ops/diffusion/__init__.py 为两个新函数注册 lazy import 映射。 - 提交历史显示测试经历两轮调整："Fix Qwen-Image bias fusion CI coverage" 与 "isolate qwen output bias B200 coverage"，把必须跑在 SM 10.3 上的用例从 B200 常规 CI 隔离出来。

关键文件：

- python/sglang/multimodal_gen/runtime/models/dits/qwen_image.py（模块 扩散模型；类别 source；类型 core-logic；符号 _can_defer_modelopt_output_bias, _defer_modelopt_output_bias, forward, forward_with_bias）：PR 主战场：新增 SM 10.3 + ModelOpt 量化门控、skip_bias_add 契约接入、forward 签名扩展、_bias_mul_add/_modulate 融合与回退逻辑，四个 bias 的延迟消费全部在此编排。
- python/sglang/kernels/jit/csrc/diffusion/norm_scale_shift.cuh（模块 内核层；类别 other；类型 core-logic；符号 bias_mul_add_kernel, BiasScaleResidualNormScaleShiftKernel, BiasMulAddKernel, norm_scale_shift_kernel）：CUDA 内核本体：norm_scale_shift_kernel 模板新增 kHasInputBias 分支，新增 bias_mul_add_kernel 与两个 kernel 封装结构体，是融合收益的底层来源。
- python/sglang/kernels/ops/diffusion/norm/norm_scale_shift_jit.py（模块 内核封装；类别 infra；类型 infrastructure；符号 _sm103, try_fused_bias_scale_residual_norm_scale_shift, try_fused_bias_mul_add）：JIT 内核入口层：新增 _sm103 设备判定、两个 try_fused_* 尝试函数，负责形状 / 设备 /dtype 校验与 kernel 名称注册，是模型代码与 CUDA 内核之间的适配层。

- test/registered/kernels/ops/diffusion/test_qwen_output_bias_absorption.py（模块 单元测试；类别 test；类型 test-coverage；符号 _seed_cuda, test_qwen_output_bias_absorption_is_sm103_only, test_qwen_output_bias_absorption_is_bit_exact, test_qwen_output_bias_absorption_rejects_unsupported_inputs）：新增测试：覆盖 SM 10.3-only 门控决策矩阵、对生产算子的 bit-exact 对比（含极端数值下的 BF16 tie-break 守卫）、unsupported 输入与编译场景的回退拒绝，是验证该优化正确性与边界约束的关键配套。
- python/sglang/kernels/ops/diffusion/__init__.py（模块 导出映射；类别 infra；类型 infrastructure）：为两个新 JIT 函数补充 lazy import 注册，保持 diffusion kernel 门面统一的按需导入模式。

关键符号：_can_defer_modelopt_output_bias, _defer_modelopt_output_bias, forward_with_bias, _bias_mul_add, _modulate, try_fused_bias_scale_residual_norm_scale_shift, try_fused_bias_mul_add, bias_mul_add_kernel, BiasScaleResidualNormScaleShiftKernel, BiasMulAddKernel

关键源码片段

python/sglang/multimodal_gen/runtime/models/dits/qwen_image.py

PR 主战场：新增 SM 10.3 + ModelOpt 量化门控、skip_bias_add 契约接入、forward 签名扩展、_bias_mul_add/_modulate 融合与回退逻辑，四个 bias 的延迟消费全部在此编排。

```
# 门控函数：只有 ModelOpt FP8/NVFP4 量化 + SM 10.3 才允许延迟输出投影 bias。
# 延迟 bias 会移动 BF16 舍入点，端到端图像质量只在 SM103 上验证过，
# 其他 GPU 必须继续走 GEMM bias epilogue 原路径，保证输出字节不变。
def _can_defer_modelopt_output_bias(
    quant_config: Optional[QuantizationConfig], capability: Any
) -> bool:
    return (
        quant_config is not None
        and hasattr(quant_config, "get_name")
        and quant_config.get_name() in {"modelopt_fp8", "modelopt_fp4"}
        and capability is not None
        and (capability.major, capability.minor) == (10, 3)
    )


def _defer_modelopt_output_bias(quant_config: Optional[QuantizationConfig]) -> bool:
    return _can_defer_modelopt_output_bias(
        quant_config, current_platform.get_device_capability()
    )


class QwenImageTransformerBlock(nn.Module):
    # 最终残差入口：bias + gate * output + residual 一次完成。
    # 优先尝试融合内核，失败则回退到 " 先加 bias 再 MulAdd" 的分步实现，
```

```
    # 回退语义与原路径完全一致，只是多一次 BF16 add kernel 启动。
    def _bias_mul_add(
        self,
        a: torch.Tensor,
        bias: Optional[torch.Tensor],
        b: torch.Tensor,
        c: torch.Tensor,
        *,
        use_bcg_helpers: bool,
    ) -> torch.Tensor:
        if bias is not None and not use_bcg_helpers:
            fused = try_fused_bias_mul_add(a, bias, b, c)
            if fused is not None:
                return fused
        if bias is not None:
            a = a + bias
        if use_bcg_helpers:
            return self._mul_add(a, b, c)
        return self.fuse_mul_add(a, b, c)
```

python/sglang/kernels/jit/csrc/diffusion/norm_scale_shift.cuh

CUDA 内核本体：norm_scale_shift_kernel 模板新增 kHasInputBias 分支，新增 bias_mul_add_kernel 与两个 kernel 封装结构体，是融合收益的底层来源。

```
// 最终残差融合内核：y = (x + bias) * gate + residual，逐行处理。
// 关键点：bias 加法先做一次 BF16 中间舍入，再用原生 __hfma 完成乘加。
// 若经 FP32 中转再存 BF16，大数值下会丢失 product-rounding tie 信息，
// 与生产 MulAdd 内核的舍入语义不再一致。
__global__ void bias_mul_add_kernel(const NormScaleShiftParams __grid_constant__ params) {
    using namespace device;
    using Vec = AlignedVector<bf16_t, kVecElems>;

    const int row_offset = blockIdx.x * kHidden;
    const int elem_offset = threadIdx.x * kVecElems;

    Vec xv;
    Vec bv;
    Vec gv;
    Vec rv;
    Vec yv;
    xv.load(static_cast<const bf16_t*>(params.x) + row_offset + elem_offset);
    bv.load(static_cast<const bf16_t*>(params.input_bias) + elem_offset);
    gv.load(static_cast<const bf16_t*>(params.gate) + elem_offset);
    rv.load(static_cast<const bf16_t*>(params.residual) + row_offset + elem_offset);

#pragma unroll
    for (int i = 0; i < kVecElems; ++i) {
        // 先做 BF16 舍入的 bias 加法，再调用 BF16 原生 FMA
        const bf16_t biased = static_cast<bf16_t>(static_cast<float>(xv[i]) + static_
```

```
            cast<float>(bv[i]));
            yv[i] = __hfma(biased, gv[i], rv[i]);
        }
        yv.store(static_cast<bf16_t*>(params.y) + row_offset + elem_offset);
    }
```

python/sglang/kernels/ops/diffusion/norm/norm_scale_shift_jit.py

JIT 内核入口层：新增 _sm103 设备判定、两个 try_fused_* 尝试函数，负责形状 / 设备 /dtype 校验与 kernel 名称注册，是模型代码与 CUDA 内核之间的适配层。

```
def try_fused_bias_mul_add(x, input_bias, gate, residual):
    # torch.compile 场景必须回退，JIT kernel 不参与图捕获
    if torch.compiler.is_compiling():
        return None
    # 仅支持 SM 10.3 + BF16 激活值；其他设备走原路径
    if not (_nss_activation(x) and _nss_activation(residual, x) and _sm103(x.device)):
        return None

    input_bias = _row_bf16(input_bias, x.device)
    gate = _row_bf16(gate, x.device)
    if input_bias is None or gate is None:
        return None

    y = torch.empty_like(x)
    _module().bias_mul_add_bf16_row(
        y.view(-1, _HIDDEN),
        x.view(-1, _HIDDEN),
        input_bias,
        gate,
        residual.view(-1, _HIDDEN),
    )
    return y
```

评论区精华

本 PR 没有任何 GitHub review 评论（唯一的 comments 是作者 BBuf 触发的 /tag-run-ci-label extra），核心设计取舍沉淀在提交历史、代码注释与测试注释中，可提炼出两条关键讨论线：

1. 架构收窄决策（对应提交 "Restrict Qwen-Image bias absorption to SM103"）：最初实现未限制架构，后来刻意收窄到 SM 10.3。PR body 明确解释原因——把 BF16 舍入点从 GEMM epilogue 移到后续算子后，端到端图像质量只在 GB300 上验证过，B200/SM 10.0 保留原 epilogue 以保证输出字节不变。
2. BF16 舍入语义保全：测试注释与 PR body 均强调，大数值下经 FP32 中转再存 BF16 会丢失 product-rounding tie 信息，因此 bias_mul_add_kernel 必须用原生 __hfma，并用专门的极端数值用例（x=-24576.0、bias=-0.01055908203125、gate=206.0、residual=-0.2080078125）守卫舍入行为。

- 为何将 bias absorption 限制在 SM 10.3 而非所有 Blackwell (design): 采用 _can_defer_modelopt_output_bias 能力门控（量化名 + capability == (10, 3)），其他架构静默回退原路径。
- 原生 BF16 __hfma 与舍入点语义保全 (correctness): bias_mul_add_kernel 使用 __hfma 保持 BF16 舍入语义，并通过 bit-exact 回归用例长期锁定。

风险与影响

- 风险:
  1. BF16 舍入点移动导致输出非字节一致（核心风险）：PR body 给出端到端指标 SSIM 0.950545、PSNR 29.262 dB、LPIPS 0.042544，图像质量差异可接受，但 main 与 PR 的 SHA256 不同。该差异仅在 SM 10.3 上被验证过，若未来在 other GPU 上放开门控，需重新做图像质量验收。
  2. skip_bias_add 数据契约跨组件变更：QwenImageCrossAttention.forward 返回元组从 2 个元素扩为 4 个，依赖方（TransformerBlock）同步解包；任何遗漏的调用方会因 tuple 解包错误或漏加 bias 而引入静默错误。当前校验覆盖有限。
  3. 静默回退语义：_bias_mul_add 与 _modulate 在融合失败时回退为 a + bias 分步实现，依赖 try_fused_* 返回 None 的判定。若未来新增 shape/ 设备分支而遗漏回退条件，可能出现双重加 bias。
  4. CI 覆盖盲区：CI 状态中 AMD ROCm 7.2 测试失败（Run #33328819277）。新测试文件通过 torch.cuda.is_available() 与 requires_sm103 做跳转，ROCm 上可能执行门控测试，失败是否与本次变更相关需确认。
  5. fast path 极窄：仅 batch 1、hidden 3072、连续对齐张量、行广播 bias/gate/scale/shift；业务侧任何形状变化都会掉回原路径，性能收益随之消失，但正确性不受影响。
- 影响:
  1. 用户侧：GB300（SM 10.3）上运行 Qwen-Image-2512 ModelOpt FP8/NVFP4 checkpoint 的用户可获得约 6% 端到端推理加速，且输出图像质量差异在 SSIM 0.95/PSNR 29.26 dB 量级；其他 GPU 用户行为完全不变。
  2. 系统侧：新增的 BiasScaleResidualNormScaleShiftKernel 与 BiasMulAddKernel 是可复用的 JIT 内核原语，后续其他 Diffusion 模型的 bias 吸收优化可直接复用 norm_scale_shift_jit.py 入口与 norm_scale_shift.cuh 模板；norm_scale_shift_kernel 的 kHasInputBias 模板参数也为同类融合扩展铺路。
  3. 团队 / 工程侧：skip_bias_add 成为 Diffusion 线性层更正式的数据契约，测试文件沉淀了 SM 10.3 专属验证模式与 BF16 舍入守卫用例，对未来做类似精度敏感优化有参考价值。 - 风险标记：SM 10.3 专属优化路径，BF16 舍入点移动导致输出非字节一致，skip_bias_add 契约跨组件变更，AMD ROCm CI 失败待确认，静默回退需防双重加 bias

关联脉络

- PR #36916 [Diffusion] Detect quantized transformer replacements: 同属 Qwen-Image transformer 量化加载路径，量化配置名称检测与回退语义与本 PR 的 modelopt 量化门控处于同一功能线。
- PR #36907 [Diffusion] Enforce component attention backend application: 直接修改同一文件 python/sglang/multimodal_gen/runtime/models/dits/qwen_image.py，涉及注意力路径的组件化配置。
- PR #36991 [Diffusion] Add exact component precision overrides: diffusion 量化精度体系的另一条演进线，与本 PR 的 ModelOpt FP8/NVFP4 精度相关路径有交叉，共同完善量化 diffusion 模型的加载与执行语义。
- PR #36832 [Diffusion] Avoid direct GPU parameter copies: 同为 diffusion 运行时性能优化，涉及 transformer_loader 与 FSDP 加载路径，与本 PR 的 JIT 融合优化形成性能优化家族。