PR #37123 完整报告

sgl-project/sglang

[Diffusion] Fuse Qwen-Image FP8 QKV projection and Blackwell epilogue

合并时间：2026-09-01 08:34

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/37123

执行摘要

- 一句话：Qwen-Image FP8 QKV 融合，端到端提速约 21%
- 推荐动作：值得精读。核心看点是：1）" 先原型、被质量门禁否决、再保精度实现 " 的迭代路径，说明性能优化必须以输出质量为准入条件；2）CUTLASS per-output-channel scale 作为同时满足精度与速度的方案，比统一 requantize 更优；3）try_fused_qwen_qkv_epilogue 的守卫 + 静默回退设计，让所有 unsupported 场景行为零变化，是 kernel 融合的稳健范式；4）零拷贝 strided view 直接喂给 JIT kernel，避免了 6 次 contiguous 拷贝。建议对照 test_qwen_qkv_epilogue.py 的 bit-exact 测试理解内核契约。

功能与动机

PR body 明确点出动机：扩散模型每个去噪步要发起数千次小 kernel 启动，Qwen-Image 的 ModelOpt 注意力投影管线存在大量可消除的碎片化开销。FP8 路径把每步 6 个投影 GEMM 和 6 次静态激活量化调用合并为 2 个 packed GEMM 和 2 次量化调用，随后被融合 epilogue 直接消费，省掉 6 次 contiguous 拷贝。PR 还记录了一个关键约束：第一个 FP8 原型把所有 checkpoint shard 重量化到最大 scale，速度虽快但质量门禁失败（SSIM 0.883575、LPIPS 0.075531），因此最终方案必须通过 CUTLASS per-output-channel scale 保留每个 shard 的原始 FP8 权重与 scale，做到与 main 像素级一致。

实现拆解

1. 变更入口：qwen_image.py 新增 _MODEL_OPT_FP8_QKV_PARAM_NAMES_MAPPING 正则映射表，把 transformer_blocks.N.attn.to_q/to_k/to_v 及 add_q_proj/add_k_proj/add_v_proj 的 weight/bias/weight_scale/input_scale 映射到 to_qkv / to_added_qkv（带 merge index 与总分片数）；get_param_names_mapping_for_quant_config() 仅对 modelopt_fp8 叠加该映射，且落到实例级 param_names_mapping，确保 eager 与 NVFP4 checkpoint 仍走分离投影参数名。构造器中新增 use_fused_qkv_epilogue（modelopt_fp4/fp8 均为 True），并把 use_fused_qkv 扩展至 modelopt_fp8，使 FP8 也进入 MergedColumnParallelLinear。
2. 权重保真处理：modelopt_quant.py 的 process_weights_after_loading 新增 complete_shard_scales 分支——CUTLASS 可用、多 shard 且所有 weight_scale 完整（大于 fp8 min）时，保留 checkpoint 原始 FP8 权重，用 convert_to_channelwise 把 per-shard scale 展开为 per-channel；否则回退 requantize_with_max_scale。

_get_quant_method 同步用新的 _is_packed_layer_excluded 替换 is_layer_skipped，能在 fused 层部分 shard 被排除时抛出 ValueError，防止同一 packed layer 内精度不一致。

3. 新 JIT epilogue：新增 qwen_qkv_epilogue_jit.py 与 CUDA 内核 qwen_qkv_epilogue.cuh。try_fused_qwen_qkv_epilogue 是唯一入口：先拒绝 torch.compiler.is_compiling() 模式，再逐项校验 QKV 输入（BF16、batch=1、head_dim=128、data_ptr 32 字节对齐、head stride 128、同源 Q/K/V 布局一致）、RMSNorm weight、FP32 RoPE cache 行数；全部满足才加载 JIT module 启动一次 kernel。内核按 Q/K/V 三种 work kind 做 grid-stride 循环，Q/K 做 warp 归约 RMS + RoPE，V 直接拷贝，并按 [text, image] 顺序写入 joint 缓冲。diffusion/__init__.py 注册该内核并声明 _CUDA_SM100_PLUS 能力要求。
4. 前向接线：common.py 的 get_qkv_projections 增加 make_contiguous 关键字参数，默认 True 保持 FLUX 等模块现状；Qwen 在 use_fused_qkv_epilogue 时传 False，保留 packed GEMM 输出的零拷贝 strided view。qwen_image.py forward 仅在 use_fused_qkv_epilogue && qk_norm && 双 cache 存在 && 非 SP sharded && 无尾部 padding 时尝试融合；失败时对 QKV 做 contiguous 后走原 QKNorm/RoPE 组合路径，行为与 main 完全一致。
5. 测试与验证配套：新增 test_qwen_qkv_epilogue.py，覆盖 contiguous 与 packed-stride 两种输入的 bit-exact 校验、compiling 拒绝、head_dim=64 拒绝；test_transformer_quant.py 覆盖 CUTLASS shard scale 保真、不完整 scale 的 requantize 回退、packed 层 shard 精度一致性、FP8 参数合并映射与非 FP8 不合并映射；test_modelopt_fp8_layerwise_offload_load.py 参数化 CUTLASS 支持与否两条加载路径。端到端验证 FP8/NVFP4 均像素级一致（SSIM 1.0、PSNR infinity、LPIPS 0）。

关键文件：

- python/sglang/multimodal_gen/runtime/models/dits/qwen_image.py（模块 模型前向；类别 source；类型 data-contract；符号 _modelopt_quant_name, _MODEL_OPT_FP8_QKV_PARAM_NAMES_MAPPING, get_param_names_mapping_for_quant_config, use_fused_qkv_epilogue）：PR 主战场：新增 FP8 专用 QKV 参数映射表、use_fused_qkv_epilogue / use_fused_qkv 标志、forward 中融合 epilogue 调用与回退逻辑，以及按量化配置切换的实例级参数名映射。
- python/sglang/kernels/ops/diffusion/rope/qwen_qkv_epilogue_jit.py（模块 融合内核；类别 infra；类型 infrastructure；符号 qwen_qkv_epilogue_module, _qkv_tensor, try_fused_qwen_qkv_epilogue）：新增 JIT epilogue 的唯一入口，承担全部守卫条件（编译模式、SM100+、布局对齐、dtype 校验）与 kernel 启动，是回退设计的关键。
- python/sglang/multimodal_gen/runtime/layers/quantization/modelopt_quant.py（模块 量化层；类别 source；类型 data-contract；符号 _is_packed_layer_excluded, process_weights_after_loading, complete_shard_scales）：实现 FP8 packed 权重后处理的精度保真分支：完整 shard scale 走 CUTLASS channelwise 保留，否则回退 requantize；并新增 _is_packed_layer_excluded 防止 fused 层 shard 精度不一致。
- test/registered/kernels/ops/diffusion/test_qwen_qkv_epilogue.py（模块 内核测试；类别 test；类型 test-coverage；符号 _seed_cuda, test_qwen_qkv_epilogue_is_bit_exact, test_qwen_qkv_epilogue_rejects_compile, test_qwen_qkv_epilogue_rejects_unsupporte

d_head_dim）：新增融合 epilogue 的 bit-exact 测试，覆盖 contiguous 与 packed-stride 两种输入布局，并验证编译模式与不支持 head_dim 时的拒绝行为。

- python/sglang/multimodal_gen/test/unit/test_transformer_quant.py（模块 量化测试；类别 test；类型 test-coverage；符号 test_modelopt_fp8_packed_cutlass_preserves_checkpoint_shard_scales, test_modelopt_fp8_packed_cutlass_requantizes_incomplete_shard_scales, test_modelopt_packed_layer_requires_consistent_shard_precision, test_qwen_modelopt_fp8_qkv_checkpoint_tensors_are_merged）：覆盖新的 FP8 packed CUTLASS 权重处理契约：shard scale 保真、不完整 scale 回退重量化、fused 层 shard 精度一致性校验，以及 Qwen FP8 参数合并映射。
- python/sglang/multimodal_gen/runtime/models/dits/common.py（模块 投影助手；类别 source；类型 data-contract；符号 get_qkv_projections）：get_qkv_projections 新增 make_contiguous 开关，使 FP8 packed GEMM 的零拷贝 strided view 能直接喂给融合 epilogue，同时保持 FLUX 等模块默认行为不变。
- python/sglang/multimodal_gen/test/unit/test_modelopt_fp8_layerwise_offload_load.py（模块 加载测试；类别 test；类型 test-coverage）：参数化 CUTLASS 支持与否两条加载路径，验证 layerwise offload 下 FP8 packed 权重后处理（保真 vs 重量化）与 CPU 落盘契约。
- python/sglang/kernels/jit/csrc/diffusion/qwen_qkv_epilogue.cuh（模块 CUDA 内核；类别 other；类型 core-logic；符号 qwen_qkv_epilogue_kernel, QwenQKVEpilogueKernel::run）：新增融合 CUDA 内核：单 kernel 完成 Q/K RMSNorm、interleaved RoPE、V 拷贝与 joint [text, image] 输出写入，是性能收益的核心载体。
- python/sglang/kernels/ops/diffusion/__init__.py（模块 内核注册；类别 infra；类型 configuration；符号 _CUDA_SM100_PLUS, try_fused_qwen_qkv_epilogue）：注册 qwen_qkv_epilogue JIT 内核与 try_fused_qwen_qkv_epilogue 符号，并声明 SM100+ 能力要求。

关键符号：try_fused_qwen_qkv_epilogue, get_param_names_mapping_for_quant_config, _modelopt_quant_name, _qkv_tensor, qwen_qkv_epilogue_module, process_weights_after_loading, _is_packed_layer_excluded, get_qkv_projections, QwenQKVEpilogueKernel::run

关键源码片段

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

PR 主战场：新增 FP8 专用 QKV 参数映射表、use_fused_qkv_epilogue / use_fused_qkv 标志、forward 中融合 epilogue 调用与回退逻辑，以及按量化配置切换的实例级参数名映射。

```
# 融合入口：ModelOpt FP8/NVFP4 下尝试用单个 JIT kernel 完成
# Q/K RMSNorm、RoPE、V 拷贝与 joint QKV 打包，减少 kernel 启动。
joint_qkv = None
if (
    self.use_fused_qkv_epilogue
    and self.qk_norm
    and img_cache is not None
    and txt_cache is not None
```

```
    and not sp_text_sharded
    and sp_txt_pad == 0
):
    # 文本流被 SP 切分或存在尾部 padding 时，joint 序列无法由
    # [text, image] 简单拼接得到，必须回退到原路径。
    joint_qkv = try_fused_qwen_qkv_epilogue(
        img_query,
        img_key,
        img_value,
        txt_query,
        txt_key,
        txt_value,
        self.norm_q.weight,
        self.norm_k.weight,
        self.norm_added_q.weight,
        self.norm_added_k.weight,
        img_cache,
        txt_cache,
        self.norm_q.variance_epsilon,
        self.norm_added_q.variance_epsilon,
    )

if joint_qkv is None:
    # 融合不可用时，ModelOpt FP8 的 QKV 是 packed GEMM 的零拷贝
    # strided view，先转回 contiguous，再走原 QKNorm/RoPE 组合，
    # 保证所有 unsupported 场景与 main 行为完全一致。
    img_query, img_key, img_value = [
        tensor.contiguous() for tensor in (img_query, img_key, img_value)
    ]
    txt_query, txt_key, txt_value = [
        tensor.contiguous() for tensor in (txt_query, txt_key, txt_value)
    ]
    if self.qk_norm:
        img_query, img_key = apply_qk_norm_with_optional_rope(
            q=img_query,
            k=img_key,
            q_norm=self.norm_q,
            k_norm=self.norm_k,
            head_dim=self.head_dim,
            cos_sin_cache=img_cache,
            is_neox=False,
            allow_inplace=True,
        )
        txt_query, txt_key = apply_qk_norm_with_optional_rope(
            q=txt_query,
            k=txt_key,
            q_norm=self.norm_added_q,
            k_norm=self.norm_added_k,
            head_dim=self.head_dim,
```

```
                cos_sin_cache=txt_cache,
                is_neox=False,
                allow_inplace=True,
            )

        # joint_qkv 命中时直接使用融合输出，否则按 [text, image] 顺序
        # 拼接两条流的 Q/K/V（join_seqs 会处理 SP 尾部 padding 搬移）。
        if joint_qkv is not None:
            joint_query, joint_key, joint_value = joint_qkv
        elif seg_qkv is not None:
            joint_query, joint_key, joint_value = seg_qkv
        else:
            joint_query = join_seqs(txt_query, img_query, sp_txt_pad)
            joint_key = join_seqs(txt_key, img_key, sp_txt_pad)
            joint_value = join_seqs(txt_value, img_value, sp_txt_pad)
```

python/sglang/multimodal_gen/runtime/layers/quantization/modelopt_quant.py

实现 FP8 packed 权重后处理的精度保真分支：完整 shard scale 走 CUTLASS channelwise 保留，否则回退 requantize；并新增 _is_packed_layer_excluded 防止 fused 层 shard 精度不一致。

```
    def process_weights_after_loading(self, layer: torch.nn.Module) -> None:
        # ROCm gfx942（MI300X/MI325X）原生 fp8 为 e4m3fnuz，而 ModelOpt
        # checkpoint 是 e4m3fn；这里先做 fp8 类型归一化，保证后续 GEMM
        # 落在原生 fp8 路径上（数值等价，仅 reinterpret + scale 加倍）。
        weight = layer.weight
        if is_fp8_fnuz():
            weight, weight_scale, input_scale = normalize_e4m3fn_to_e4m3fnuz(
                weight=layer.weight,
                weight_scale=layer.weight_scale,
                input_scale=layer.input_scale,
            )
            copy_or_rebind_param(layer, "weight_scale", weight_scale)
            if input_scale is not None:
                copy_or_rebind_param(layer, "input_scale", input_scale)

        # 判定“shard scale 完整”：CUTLASS 可用、确为多 shard fused 层、
        # 且所有 shard 的 weight_scale 都被真实写入（占位初始值为 fp8 min）。
        complete_shard_scales = (
            self.cutlass_fp8_supported
            and len(layer.logical_widths) > 1
            and bool(
                torch.all(
                    layer.weight_scale
                    > torch.finfo(torch.float8_e4m3fn).min
                ).item()
            )
        )
```

```
if complete_shard_scales:
    # CUTLASS 支持 per-output-channel scale：保留 checkpoint 每个
    # shard 的原始 FP8 权重与 scale，避免重量化到最大 scale 引入
    # 精度损失（首个原型因质量门禁失败而被否决的根因）。
    quantized_weight = weight
    processed_weight_scale = convert_to_channelwise(
        layer.weight_scale, layer.logical_widths
    )
else:
    # 存在缺失 scale 的 shard，或当前平台不支持 channelwise：
    # 仍走原有"统一重量化到最大 scale"的回退逻辑。
    processed_weight_scale, quantized_weight = requantize_with_max_scale(
        weight, layer.weight_scale, layer.logical_widths
    )
    if self.cutlass_fp8_supported:
        processed_weight_scale = convert_to_channelwise(
            processed_weight_scale, layer.logical_widths
        )
# 运行时 kernel 需要转置的 FP8 view，同时保留 Parameter 子类元数据。
layer.weight.data = quantized_weight.t().detach()
layer.weight.requires_grad_(False)
copy_or_rebind_param(layer, "weight_scale", processed_weight_scale)
copy_or_rebind_param(layer, "input_scale", layer.input_scale.max())
```

评论区精华

本 PR 没有内联 review 评论（review_comments_count=0），issue 评论仅为 CI 触发命令；核心设计权衡记录在 PR body 中。最重要的交锋是：第一个 FP8 原型把三个 checkpoint shard 重量化到最大 scale，速度达标但质量门禁失败（SSIM 0.883575、LPIPS 0.075531），实现被否决；最终改用 CUTLASS 已有的 per-output-channel scale 支持，保留每个 shard 原始 FP8 权重与 scale，才达到像素级一致。另一个明确决策是 NVFP4 投影不合并，因为激活与权重 scale 布局不可互换。此外 FP8 torch.compile + breakable CUDA graph 在 main 与 PR 两侧同样失败于 _static_quant_fp8: PassManager::run failed，PR 的 compile 守卫与通用回退保证该场景行为不变，属既有编译栈限制而非回归。

- 首个 FP8 原型因质量门禁被否决 (correctness): 最终改用 CUTLASS per-output-channel scale 保留各 shard 原始 FP8 权重与 scale，50 步去噪后与 main 像素级一致（SSIM 1.0、PSNR infinity、LPIPS 0）。
- CUTLASS channelwise scale 保真替代统一 requantize (design): 以"完整 shard scale 则保真、否则回退"的双分支同时满足精度与兼容性，是质量门禁失败后的关键设计修正。
- NVFP4 投影为何不合并 (design): NVFP4 仅受益于 epilogue 融合（约 4% 提速），不冒险合并投影。
- FP8 torch.compile + BCG 不可用的边界 (performance): 判定为既有 FP8 编译栈限制而非本 PR 回归，编译场景保持原行为，融合优化暂不覆盖编译部署形态。

风险与影响

- 风险：
  1. Checkpoint 加载契约变更：modelopt_fp8 下 QKV 参数名从 to_q/to_k/to_v 合并为 to_qkv，依赖分离参数名的自定义 loader 或脚本会受影响；_is_packed_layer_excluded 对部分排除的 fused 层直接抛 ValueError，可能让原本容忍的配置变成启动失败。
  2. 平台与布局限定：融合 epilogue 仅 SM100+、BF16、batch=1、head_dim=128、32 字节对齐且文本流未 SP 分片 / 无 padding 时生效，真实 shape 不满足时静默回退，性能预期可能与基准不一致。
  3. 编译模式不覆盖：is_compiling() 守卫使 torch.compile + BCG 下的 FP8 完全不走融合路径，优化无法覆盖编译部署形态。
  4. AMD ROCm 7.2 CI 失败（Run #33386190728 为 x）：modelopt_quant.py 中 normalize_e4m3fn_to_e4m3fnuz 与 cutlass_fp8_supported 分支存在 ROCm 平台行为差异风险，需确认失败原因与本 PR 的关系。
  5. 质量验证覆盖有限：像素级一致仅在固定 seed、1024x1024、50 步、CFG 4.0 下验证，其他采样配置的数值等价性依赖 CUTLASS 数值行为稳定。 - 影响：性能影响：Qwen-Image ModelOpt FP8 在 GB300 上去噪每步从 310.168ms 降至 245.247ms（-20.93%），端到端 -20.75%；NVFP4 去噪每步 -4.02%；每步 GPU kernel 启动从 4989 次降至 3309 次。功能影响：仅改动 Qwen-Image 且仅 modelopt_fp8 与 modelopt_fp4（部分受益）路径，其他模型与量化配置不受影响，不支持场景自动回退。工程影响：为 diffusion JIT 融合确立了守卫 + 回退 + 像素级验证的流程范式，并建立了 FP8 packed 参数映射与 CUTLASS channelwise scale 保真的新契约，后续 Qwen-Image 融合可直接复用。 - 风险标记：核心前向路径变更，Checkpoint 加载契约变更，平台限定 Blackwell+，编译模式不覆盖，AMD CI 失败待确认，大量守卫条件分支

关联脉络

- PR #37129 [Diffusion] Fuse Qwen-Image residual norm and NVFP4 quantization: 同属 Qwen-Image diffusion 内核融合系列，上一步融合 residual norm 与 NVFP4 量化，与本 PR 共用 diffusion JIT kernel 注册与量化融合基础设施。
- PR #37156 [Diffusion] Fuse Qwen-Image FP8 norm and activation quantization: 同属 Qwen-Image FP8 融合系列，已融合 FP8 norm 与激活量化；本 PR 在其基础上进一步合并 QKV 投影并新增 epilogue，端到端提速约 21% 与之一致。
- PR #37144 [Diffusion] Fuse Qwen-Image final adaptive LayerNorm: 同属 Qwen-Image 融合系列并引入 bit-exact 自校验回退模式，本 PR 的像素级一致验证与回退设计延续该模式；PR body 也声明 rebase 包含了 Qwen CFG modulation cache 相关改动。
- PR #37141 [Diffusion] Fuse FLUX.2 token concatenation and NVFP4 quantization: FLUX.2 token 拼接与 NVFP4 量化融合，与本次 QKV 融合共享 diffusion JIT kernel 注册、CUTLASS FP8 通道 scale 与打包策略。