PR #35006 完整报告

sgl-project/sglang

[Diffusion] Reuse SRT Qwen vision and text modules

合并时间：2026-08-19 10:12

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35006

执行摘要

- 一句话：Qwen 视觉与文本模块复用 SRT 原语，去重千行
- 推荐动作：值得精读。该 PR 展示了跨 runtime 复用的边界设计：通用原语下沉到 SRT，而 attention 与量化这类与 runtime 契约强耦合的模块留在 MMGen；同时以数值保真测试和两卡折叠校验保障重构安全。建议重点关注三处：Qwen2_5_VLMLP 的 TP 三段式选型、RMSNorm 高阶 residual 的 flatten/restore 逻辑、以及 TP group 绑定 / 恢复的生命周期管理。

功能与动机

PR body 明确指出：The multimodal generation Qwen2.5-VL, Qwen3, and Qwen3-VL encoders duplicated pure modules already implemented in SRT. Maintaining both copies also left the MMGen paths without reusable SRT tensor-parallel projections, embeddings, activation kernels, and checkpoint loaders. 核心问题是 MMGen 与 SRT 各维护一份 Qwen 系列模块，既增加双份维护成本，又让 diffusion 路径无法使用 SRT 成熟的 TP 线性层、嵌入、激活 kernel 与 checkpoint loader。本 PR 的目标是在不破坏 diffusion 特有 attention/ 量化契约的前提下，把通用原语下沉到 SRT。

实现拆解

1. 建立共享视觉注意力中枢：新增 qwen_vl_vision.py，定义 PackedSequenceMetadata 打包 cu_seqlens、cu_seqlens_host、max_seqlen，新增 QwenVLVisionAttention 把原来的 nn.Linear 换成 SRT 的 QKVParallelLinear / RowParallelLinear，并按 get_parallel().tp_size 切分头数；backend 支持 packed varlen 时走自定义 kernel，否则逐段回退 torch SDPA。
2. 视觉塔与文本塔去重：qwen2_5vl_vision.py、qwen3vl_vision.py 删除本地实现的 Attention、MLP、RMSNorm、PatchEmbed，改为从 SRT 的 qwen2_5_vl.py / qwen3_vl.py 导入；qwen2_5vl.py 删除自实现 Qwen2_5_VLTextMLP，改用 fuse_gate_up=False 的 SRT Qwen2_5_VLMLP 保持 act(gate) * up 顺序，qwen3vl.py 的 q_norm / k_norm 与 norm 换成 SRT RMSNorm；保留 LocalAttention、量化线性层与 model.visual.blocks 层归属。
3. 扩展 SRT 公共组件：srt/models/qwen2_5_vl.py 新增本地 Qwen2_5_VisionPatchEmbed（Conv3dLayer + disable_linear）与 Qwen2_5_VisionRotaryEmbedding，Qwen2_5_VLMLP 增加 fuse_gate_up、tp_size、tp_rank 参数且默认值保持旧行为；

srt/models/qwen3_vl.py 新增 _resolve_vision_tp；srt/layers/layernorm.py 支持高阶 residual 输入；srt/layers/activation.py 让 SiluAndMul 可在无 server context 时工作。

4. TP 组绑定与 checkpoint 名称映射：encoder root 通过 EncoderTensorParallelMixin 绑定到构造 / 加载 /forward 所用的 TP group，嵌套 context 同时切换 MMGen TP、SRT TP 与 SRT attention TP 并在结束后独立恢复；qwen3vl.py 的 load_weights 对 visual. 前缀做 .attn.qkv. → .attn.qkv_proj. 映射，兼容 MiniMax-H3 checkpoint。
5. 测试与验证配套：新增 test_qwen_vl_rope.py、test_encoder_fold_srt_linear_2_gpu.py、test_component_accuracy_weight_transfer.py，扩展 test_layernorm_fusion.py、test_qwen3vl_text.py、test_qwen2_5vl_generation.py 等，覆盖 Transformers v5 配置、GQA 布局、TP=1/TP=2、fold/replicate 数值与 E2E 图像质量指标。

关键文件：

- python/sglang/multimodal_gen/runtime/models/encoders/qwen_vl_vision.py（模块 视觉注意力；类别 source；类型 data-contract；符号 PackedSequenceMetadata, from_cu_seqlens, QwenVLVisionAttention, _packed_attention）：新增的共享视觉注意力中枢，Qwen2.5-VL 与 Qwen3-VL 视觉塔都改为引用此文件；它把 SRT 的 QKVParallelLinear/RowParallelLinear 与 MMGen 的 packed attention 契约焊接在一起，是本 PR 复用设计的核心接缝。
- python/sglang/multimodal_gen/runtime/models/encoders/qwen2_5vl_vision.py（模块 视觉编码器；类别 source；类型 data-contract；符号 Qwen2_5VLVisionBlock, Qwen2_5VLVisionTransformer, PackedSequenceMetadata）：从 285 行本地实现缩减到 75 行左右，删除 Qwen2_5VLVisionAttention、Qwen2_5VLVisionMLP、Qwen2_5VLVisionRMSNorm 等重复代码，改为复用 SRT 组件，是本次去重收益最大的文件之一。
- python/sglang/multimodal_gen/runtime/models/encoders/qwen3vl_vision.py（模块 视觉编码器；类别 source；类型 data-contract；符号 Qwen3VLVisionBlock, Qwen3VLVisionTransformer, Qwen3VLVisionOutput）：与 qwen2_5vl_vision.py 对称的大规模去重，Qwen3-VL 视觉塔复用 QwenVLVisionAttention、SRT Qwen3_VisionMLP、PatchEmbed 与 MoE PatchMerger，同时保留 deepstack 特征与 position embedding 插值等模型特有逻辑。
- python/sglang/multimodal_gen/runtime/models/encoders/qwen2_5vl.py（模块 文本编码器；类别 source；类型 data-contract；符号 _tp_rank, Qwen2_5_VLDecoderLayer, Qwen2_5_VLMLP）：Qwen2.5-VL 文本路径的复用中枢：删除自实现 Qwen2_5_VLTextMLP 与 HF Qwen2RMSNorm，改用 SRT Qwen2_5_VLMLP、RMSNorm 与新增 MRoPE 适配器，并处理 MLP 非 TP-divisible 时的单 rank 回退。
- python/sglang/srt/models/qwen2_5_vl.py（模块 模型实现；类别 source；类型 data-contract；符号 Qwen2_5_VisionPatchEmbed, Qwen2_5_VisionRotaryEmbedding, Qwen2_5_VLMLP, Qwen2_5_VisionPatchMerger）：SRT 侧为支持复用而做的公共组件扩展：新增本地 Qwen2_5_VisionPatchEmbed 与 Qwen2_5_VisionRotaryEmbedding，Qwen2_5_VLMLP 增加 fuse_gate_up/tp_size/tp_rank 参数，默认保持旧行为。
- python/sglang/multimodal_gen/runtime/models/encoders/qwen_vl_rope.py（模块 旋转编码；类别 source；类型 data-contract；符号 build_qwen_vl_text_rope,

apply_qwen_vl_text_rope）：新增的 MRoPE 适配器，把 SRT get_rope 封装成 MMGen可独立构造的 Qwen-VL 文本 RoPE，并强制走 forward_native 以保持 HF 的 bf16 算术顺序。

- python/sglang/srt/models/qwen3_vl.py（模块 模型实现；类别 source；类型 data-contract；符号 _resolve_vision_tp, Qwen3_VisionMLP, Qwen3VLVisionPatchEmbed）：SRT 侧 Qwen3-VL 组件增加显式 TP 参数支持，_resolve_vision_tp 统一处理 use_data_parallel、显式 TP 与默认 attn_tp 的回退逻辑。
- python/sglang/srt/layers/layernorm.py（模块 归一化层；类别 source；类型 core-logic；符号 RMSNorm）：扩展 SRT 核心 RMSNorm 以支持更高 rank 的 residual 输入，是 diffusion 文本编码器 [batch, sequence, hidden] 状态可复用 SRT 归一化的前置条件。
- python/sglang/multimodal_gen/test/unit/test_qwen_vl_rope.py（模块 单元测试；类别 test；类型 test-coverage；符号 build_qwen_vl_text_rope, apply_qwen_vl_text_rope, _RecordingRotaryEmbedding）：新增 MRoPE 适配器的专项测试，覆盖 Transformers v5 配置、interleaved 布局、GQA 布局与形状校验，并断言不依赖 SRT runtime context。
- python/sglang/multimodal_gen/test/single_test_file/test_encoder_fold_srt_linear_2_gpu.py（模块 集成测试；类别 test；类型 test-coverage；符号 FoldedEncoder, test_folded_srt_linear_matches_unsharded_reference）：两 GPU 折叠校验：用真实 SRT RowParallelLinear 验证 folded encoder 在 DiT TP=1 / encoder TP=2 下输出与完整 unsharded 线性层一致，且三类 TP group 全部恢复为 1。

关键符号：PackedSequenceMetadata.from_cu_seqlens, QwenVLVisionAttention.init, QwenVLVisionAttention.forward, QwenVLVisionAttention._packed_attention, build_qwen_vl_text_rope, apply_qwen_vl_text_rope, Qwen2_5_VLMLP.init, Qwen2_5_VLMLP.forward, Qwen2_5_VisionPatchEmbed.forward, _resolve_vision_tp, _make_text_rms_norm, Qwen2_5VLVisionBlock.forward, Qwen3VLVisionBlock.forward

关键源码片段

python/sglang/multimodal_gen/runtime/models/encoders/qwen2_5vl_vision.py

从 285 行本地实现缩减到 75 行左右，删除 Qwen2_5VLVisionAttention、Qwen2_5VLVisionMLP、Qwen2_5VLVisionRMSNorm 等重复代码，改为复用 SRT 组件，是本次去重收益最大的文件之一。

```
class Qwen2_5VLVisionBlock(nn.Module):
    def __init__(self, config: Any, layer_idx: int) -> None:
        super().__init__()
        # 直接复用 SRT 的 RMSNorm 与共享注意力，视觉塔由此获得 SRT 的 TP 线性层与 checkpoint
        loader
        self.norm1 = RMSNorm(
            config.hidden_size, eps=1e-6,
            cast_x_before_out_mul=True, force_native=True,
        )
        self.norm2 = RMSNorm(
            config.hidden_size, eps=1e-6,
            cast_x_before_out_mul=True, force_native=True,
```

```
        )
        self.attn = QwenVLVisionAttention(
            config, prefix=f"visual.blocks.{layer_idx}.attn", model_name="Qwen2.5-VL",
        )
        self.mlp = Qwen2_5_VLMLP(
            config.hidden_size, config.intermediate_size, bias=True,
            hidden_act=config.hidden_act,
            prefix=f"visual.blocks.{layer_idx}.mlp", fuse_gate_up=False,
        )

    def forward(
        self, hidden_states: torch.Tensor, *, metadata: PackedSequenceMetadata,
        position_embeddings: tuple[torch.Tensor, torch.Tensor],
    ) -> torch.Tensor:
        hidden_states = hidden_states + self.attn(
            self.norm1(hidden_states), metadata=metadata,
            position_embeddings=position_embeddings,
        )
        return hidden_states + self.mlp(self.norm2(hidden_states))


class Qwen2_5VLVisionTransformer(nn.Module):
    # forward 中构造 full 与 window 两套 packed 元数据并逐层选择
    def forward(self, hidden_states: torch.Tensor, grid_thw: torch.Tensor) -> torch.Tensor:
        # ... 前置 patch embed、窗口重排与位置编码构造省略，核心是两套 metadata 的生成
        cu_seqlens = torch.repeat_interleave(
            grid_thw[:, 1] * grid_thw[:, 2], grid_thw[:, 0]
        ).cumsum(dim=0, dtype=torch.int32)
        cu_seqlens = F.pad(cu_seqlens, (1, 0), value=0)
        full_metadata = PackedSequenceMetadata.from_cu_seqlens(cu_seqlens)
        window_metadata = PackedSequenceMetadata.from_cu_seqlens(cu_window_seqlens)

        for layer_idx, block in enumerate(self.blocks):
            # 整图注意力层用 full 序列，其余层用窗口序列，契约由 MMGen 持有
            metadata = (
                full_metadata if layer_idx in self.full_attention_layers
                else window_metadata
            )
            hidden_states = block(
                hidden_states, metadata=metadata,
                position_embeddings=position_embeddings,
            )
        merged = self.merger(hidden_states)
        return merged[torch.argsort(window_index)]
```

python/sglang/multimodal_gen/runtime/models/encoders/qwen_vl_rope.py

新增的 MRoPE 适配器，把 SRT get_rope 封装成 MMGen 可独立构造的 Qwen-VL 文本 RoPE，并强制走 forward_native 以保持 HF 的 bf16 算术顺序。

```
def build_qwen_vl_text_rope(
    config: Any, *, mrope_interleaved: bool = False
) -> RotaryEmbedding:
    # 统一走 SRT 的 get_rope，但允许在无 SRT runtime context 的 MMGen 环境独立构造
    head_dim = getattr(config, "head_dim", None) or (
        config.hidden_size // config.num_attention_heads
    )
    rope_theta, rope_scaling = get_rope_config(config)
    rope_scaling = dict(rope_scaling or {})
    rope_scaling["mrope_interleaved"] = mrope_interleaved
    return get_rope(
        head_size=head_dim,
        rotary_dim=head_dim,
        max_position=config.max_position_embeddings,
        base=rope_theta,
        is_neox_style=True,
        rope_scaling=rope_scaling,
    )


def apply_qwen_vl_text_rope(
    rotary_emb: RotaryEmbedding, position_ids: torch.Tensor,
    query: torch.Tensor, key: torch.Tensor,
) -> tuple[torch.Tensor, torch.Tensor]:
    """对 4D 的 [batch, heads, sequence, head_dim] 张量应用三轴 MRoPE。"""
    # ... 形状校验省略：query/key 必须为 4D，position_ids 必须为 [3, batch, sequence]
    query = query.transpose(1, 2).reshape(-1, num_query_heads * head_dim)
    key = key.transpose(1, 2).reshape(-1, num_key_value_heads * head_dim)
    # 保持 HF 的 bf16 算术顺序，融合 MRoPE 会改变生成图像
    query, key = rotary_emb.forward_native(position_ids.reshape(3, -1), query, key)
    query = query.view(batch_size, sequence_length, num_query_heads, head_dim)
    key = key.view(batch_size, sequence_length, num_key_value_heads, head_dim)
    return query.transpose(1, 2), key.transpose(1, 2)
```

评论区精华

本 PR 没有任何 reviewer 评论（review_comments_count = 0），核心权衡集中在 PR body 与 commit 序列中：

- 作者明确强调 Keep the Qwen2.5 split gate/up execution order as act(gate) * up; this avoids a concatenation and preserves HF numerical behavior.，即在复用 SRT MLP 时坚持保留 split 执行顺序以保证数值一致。
- TP 选型上采用 Use replicated SRT linears at TP=1 and explicit TP rank/size at TP>1, while retaining the existing replicated fallback when a dimension is not TP-divisible. 的三段式策略，对应 commit Avoid TP wrappers in single-rank Qwen MLPs 与 Unify single-rank Qwen MLP projections 的反复打磨。
- 对 attention 边界，作者说明共享的 packed-sequence 适配器 remains MMGen-owned because diffusion backend selection and segmented SDPA fallback differ from SRT

serving's attention contract.，明确区分了“可复用原语”与“runtime 契约绑定模块”。

- SRT 复用边界：attention 与量化线性层为何保留 MMGen 实现 (design): 通用原语（PatchEmbed、MLP、RMSNorm、MRoPE、TP 线性层）统一到 SRT；attention 与量化这类依赖 runtime 契约的模块继续留在 MMGen，避免破坏 diffusion 的 backend 选择与量化语义。
- TP=1 与 TP>1 的线性层选型：ReplicatedLinear vs 显式 TP + 非整除 fallback (design): 最终采用 TP=1 时 ReplicatedLinear、TP>1 且可整除时显式 TP、不可整除时回退 replicated 的三段式策略，并新增 SRT Qwen2_5_VLMLP 的 fuse_gate_up 分支兼容两种路径。
- MRoPE 数值保真：为什么用 forward_native 而不用融合实现 (correctness): 对 Qwen-VL 文本 attention 复用 SRT rotary embedding 但强制走 native 路径，避免数值漂移；单元测试验证最大误差 2.38e-7（FP32）与 1-2 ULP（BF16）。

风险与影响

- 风险：
  - 跨模块耦合加深：MMGen 与 SRT 从此共享核心层实现，未来任何对 RMSNorm、Qwen2_5_VLMLP、SRT RoPE 的改动都会同时影响 serving 与 diffusion 两条路径，回归面扩大。
  - SRT 基础模块行为变化：srt/layers/layernorm.py 的高阶 residual 支持改动影响所有使用 residual 的模型；Qwen2_5_VLMLP 在 TP=1 时将 down_proj 从 RowParallelLinear 换成 ReplicatedLinear，需关注量化装载与自定义 weight loader 的兼容性。
  - 数值保真依赖：apply_qwen_vl_text_rope 刻意走 forward_native 以保持 HF 的 bf16 算术顺序，如果未来 SRT 融合 MRoPE 被默认启用，会再次出现图像漂移。
  - TP 覆盖不足：两卡折叠测试只覆盖 TP=2 单实例；更高 rank（TP=4/8）与多实例并发下的 group 恢复未验证。
  - 工程流程风险：0 条 review 评论、作者自行合并，26 个文件 1300+ 行改动的评审深度有限。
- 影响：
  - 用户侧：无公开 API 与 CLI 变化，默认行为保持兼容；但任何数值偏差都会直接影响生成图像质量，因此本 PR 用 SSIM、PSNR、CLIP、余弦相似度等多维指标做了严格回归对比。
  - 系统侧：MMGen 的 Qwen2.5-VL、Qwen3-VL、Qwen3、MiniMax-H3 路径全部切换到 SRT 原语，视觉塔与文本塔获得 SRT 的 TP 线性层、嵌入、激活 kernel 与 checkpoint loader；SRT 侧新增本地 Qwen2.5 视觉模块并扩展 MLP/Layernorm 能力。
  - 团队侧：消除两套重复实现，降低长期维护成本，但 MMGen 与 SRT 的耦合关系需要形成稳定的模块边界约定，避免后续改动互相踩踏。
  - 性能侧：Qwen-Image E2E 12697.67ms 对 12799.17ms、MiniMax-H3 2GPU 47120.89ms 对 47538.57ms，视觉塔保持 1.08x/1.19x 加速，均无回归。

- 风险标记：跨模块重构：MMGen 与 SRT 耦合加深，核心基础模块变更：SRT RMSNorm 与 Qwen2_5_VLMLP，数值保真依赖 HF 算术顺序，TP 折叠路径覆盖不足（仅 2 GPU），无 reviewer 评论，作者自合

关联脉络

- PR #34713 [diffusion] Decouple encoder parallelism from the DiT parallel layout: 同为 diffusion encoder 并行布局改造，本 PR 的 TP group 绑定、fold/replicate 行为是该 PR 引入的 encoder 并行解耦的延续。
- PR #34581 [Diffusion] Optimizing MiniMax-H3 for consumer-level GPUs: INT8 Linear + pluggable DiT attention backends: MiniMax-H3 的注意力与量化路径，本 PR 新增 .attn.qkv. 到 .attn.qkv_proj. 权重映射和共享视觉注意力，需要与其注意力后端机制对齐。
- PR #35339 [diffusion] Per-request lossy accelerations: Cache-DiT, CFG gating, attention backend override: attention backend override 使注意力后端选择按请求动态化，与本 PR 共享的 QwenVLVisionAttention 的 backend 选择逻辑在同一契约上演进。
- PR #35004 Native pipeline encoder fold vs replicate e2e guard: PR body 明确说明完整的 fold 与 replicate e2e guard 在 #35004，本 PR 的两卡单测是其前置校验。