PR #34949 完整报告

sgl-project/sglang

[Diffusion] Route MiniMax H3 VAE attention through native backends

合并时间：2026-08-16 10:07

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34949

执行摘要

- 一句话：H3 VAE 注意力统一走原生 USP 后端，删除本地 SDPA 包装
- 推荐动作：值得精读：attention.py 展示了如何将封闭的 VAE bundle 接入组件化后端选择并保持精度契约；audio_vae.py 的 dtype 处理（显式 fused 才 cast）是值得参考的精度策略；layer.py 的 skip_sequence_parallel 与 ring 准入解耦值得关注。建议在合入后跟踪 ROCm gfx95 与 MPS 路径的回归测试结果，并确认与 #34891 的注意力后端选择改动协调一致。

功能与动机

PR body 明确指出根因：旧代码暴露了名为 flash_attn 的助手，但其密集路径总是调用 PyTorch SDPA，绕过 SGLang 的注意力后端机制。同时，发布版 H3 视频解码器是非因果的（causal_decoder=False），其 block-causal mask、varlen、valid-padding 和 pack_info 分支没有运行时调用者，使实际行为难以判断；而 H3 DiT/token refiner 已使用原生 packed-varlen 后端分派，Qwen 文本路径已使用 LocalAttention。因此需要让两个 VAE 注意力模块通过 USPAttention 走原生组件 / 后端选择，并保留显式 FP32、MPS、ROCm 正确性回退。

实现拆解

本 PR 的改造按以下步骤展开：

1. 删除本地 SDPA 包装器（flash.py）：整体移除 minimax_h3_video_vae/flash.py 中 190 行代码，包括 _sdpa_attention、_sdpa_kernel_context、_mask_mod_to_dense、make_block_causal_mask_mod 等 mask 转 dense 与 block-causal 辅助逻辑。这些代码服务于无运行时调用者的因果 / 掩码 /varlen 分支；同时同步清理 vae_vit.py 的 t_causal 分支与 base_module.py 中 TransformerBlock.forward 的 pack_info 参数，并在 ViT3DDecoder.__init__ 显式拒绝 t_causal=True，防止未来误用。
2. 视频 VAE 注意力模块改造（attention.py）：Attention.__init__ 在 CUDA 平台创建 USPAttention(num_heads, head_size, causal=False, skip_sequence_parallel=True)，CUDA 下 FP16/BF16 输入直接走 self.attn(query, key, value)，让组件级覆盖（如 video_vae=torch_sdpa）生效；FP32 与非 CUDA（如 MPS）路径保留 _sdpa_attention 直连 SDPA，避免 FlashAttention 不接受 FP32 破坏既有精度契约；ROCm gfx95 上继续用 sdpa_kernel([MATH]) 回退。同时删除 _perform_attention/perform_attention 的 pack_info 接口，Attention.forward 签名简化。

3. 音频 VAE 注意力改造（audio_vae.py）：CausalAttention 接入 USPAttention(causal=True, supported_attention_backends={FA, TORCH_SDPA}, default_attention_backend=TORCH_SDPA, skip_sequence_parallel=True)，默认仍为 FP32 causal SDPA；仅当显式选择 FA 等 fused 后端时才将 QKV cast 到后端 dtype，并在输出恢复原 dtype。q/k/v 张量布局从 (2,0,3,1,4) 调整为 (2,0,1,3,4)，对应后续 mean(dim=2) / reshape(B,N,-1)。
4. 原生注意力层扩展（layer.py / selector.py）：USPAttention.__init__ 和 get_attn_backend 新增 default_attention_backend 参数，作为无全局或组件覆盖时的回退；USPAttention 在 skip_sequence_parallel=True 时跳过 get_ring_parallel_world_size() > 1 的 ring 能力检查，使 tile-local VAE 解码无需 ring 能力。
5. 配套上下文与测试：minimax_h3/stages/decoding.py 在 video_decode 外套上 set_forward_context(current_timestep=0, attn_metadata=None)，满足原生注意力在 VAE 解码期间对 forward context 的要求。test_minimax_h3_vae_parallel_modes.py 新增三个契约测试（test_vit_attention_uses_local_usp_backend_dispatch、test_vit_qk_norm_supports_affine_free_rmsnorm、test_audio_vae_attention_defaults_to_local_sdpa_and_allows_fa），分别锁定 skip_sequence_parallel、affine-free RMSNorm 和音频 VAE 的 SDPA 默认 /FA 准入 /QKV cast/FP32 恢复；test_ring_admission.py 新增 test_local_usp_backend_does_not_require_ring_capability，验证 local USP 后端在 ring world size 大于 1 时也能正常创建。

关键文件：

- python/sglang/multimodal_gen/runtime/models/vaes/minimax_h3_video_vae/attention.py（模块 视频 VAE；类别 source；类型 core-logic；符号 _sdpa_attention, _perform_attention, perform_attention, Attention.forward）：视频 VAE 注意力入口：用 USPAttention 替换本地 flash_attn 包装，CUDA FP16/BF16 走原生后端分派，FP32/MPS/ROCm 保留 SDPA 回退，是 PR 核心变更点。
- python/sglang/multimodal_gen/runtime/models/vaes/minimax_h3_video_vae/flash.py（模块 视频 VAE；类别 source；类型 deletion；符号 _auto_sdpa_backend_name, _as_bool_mask, _ensure_nonempty_rows, _sdpa_kernel_context）：被整体删除的 190 行本地 SDPA 包装，包含 mask_mod/block-causal/varlen 等无运行时调用者的分支，是本 PR 消除重复与死代码的核心对象。
- python/sglang/multimodal_gen/runtime/models/vaes/minimax_h3_audio_vae/audio_vae.py（模块 音频 VAE；类别 source；类型 core-logic；符号 CausalAttention.init, CausalAttention.forward）：音频 VAE 的 CausalAttention 接入 USPAttention，新增 FA/TORCH_SDPA 后端准入与 FP32 精度契约管理，是音频侧的核心变更。
- python/sglang/multimodal_gen/runtime/layers/attention/layer.py（模块 注意力层；类别 source；类型 core-logic；符号 USPAttention.init）：USPAttention 核心层：新增 default_attention_backend 参数，并让 skip_sequence_parallel 跳过 ring 能力检查，是支撑 VAE tile-local 解码的关键底层改动。

- python/sglang/multimodal_gen/runtime/models/vaes/minimax_h3_video_vae/vae_vit.py（模块 视频 VAE；类别 source；类型 data-contract；符号 forward_transformer_blocks）：移除 t_causal/block-causal mask 死代码，并在 __init__ 显式拒绝 t_causal=True，明确 released 解码器的非因果契约。
- python/sglang/multimodal_gen/test/unit/test_minimax_h3_vae_parallel_modes.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_vit_attention_uses_local_usp_backend_dispatch, test_vit_qk_norm_supports_affine_free_rmsnorm, test_audio_vae_attention_defaults_to_local_sdpa_and_allows_fa）：新增三个 VAE 注意力契约测试，锁定 skip_sequence_parallel、affine-free RMSNorm 与音频 VAE 的 SDPA 默认 /FA 准入 / 精度恢复，是本次改造的回归保障。
- python/sglang/multimodal_gen/runtime/layers/attention/selector.py（模块 后端选择；类别 source；类型 core-logic；符号 get_attn_backend）：get_attn_backend 新增 default_attention_backend 回退选择，与 layer.py 的改动共同构成后端选择回退链。
- python/sglang/multimodal_gen/test/unit/test_ring_admission.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_local_usp_backend_does_not_require_ring_capability）：新增 local USP 后端在 ring world size 大于 1 时无需 ring 能力的测试，验证 skip_sequence_parallel 与 ring 准入解耦。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/minimax_h3/stages/decoding.py（模块 解码管线；类别 source；类型 data-contract）：在 VAE 解码调用外新增 set_forward_context，满足原生注意力在 VAE 解码期间对 forward context 的要求。
- python/sglang/multimodal_gen/runtime/models/vaes/minimax_h3_video_vae/base_module.py（模块 VAE 通用；类别 source；类型 data-contract；符号 TransformerBlock.forward）：TransformerBlock.forward 去掉 pack_info 参数，与新的无掩码 Attention 接口对齐。

关键符号：Attention.forward, Attention.init, CausalAttention.forward, CausalAttention.init, USPAttention.init, get_attn_backend, forward_transformer_blocks, TransformerBlock.forward, _sdpa_attention

关键源码片段

python/sglang/multimodal_gen/runtime/models/vaes/minimax_h3_video_vae/attention.py

视频 VAE 注意力入口：用 USPAttention 替换本地 flash_attn 包装，CUDA FP16/BF16 走原生后端分派，FP32/MPS/ROCm 保留 SDPA 回退，是 PR 核心变更点。

```
# MiniMax H3 视频 VAE 注意力模块（推理专用）
# 关键路径：CUDA + FP16/BF16 走 USPAttention 原生后端分派，
# FP32 与 MPS 等路径保留直连 SDPA，避免改变发布版精度契约

# gfx95 上 fused ROCm SDPA 会破坏密集 ViT 解码，强制回退 math 后端
_FORCE_ROCM_MATH_SDPA = current_platform.is_rocm() and "gfx95" in str(
    torch.cuda.get_device_properties(0).gcnArchName
```

```
)


def _sdpa_attention(query, key, value):
    # 非 gfx95 平台交由 PyTorch 自动选择 SDPA 后端；gfx95 强制 MATH
    context = sdpa_kernel([SDPBackend.MATH]) if _FORCE_ROCM_MATH_SDPA else nullcontext()
    with context:
        return F.scaled_dot_product_attention(
            query.transpose(1, 2),
            key.transpose(1, 2),
            value.transpose(1, 2),
            dropout_p=0.0,
        ).transpose(1, 2)


class Attention(nn.Module):
    def __init__(self, heads, dim_head, embed_dim=None, qk_norm_type=None, **kwargs):
        # ... 此处省略 to_qkv / to_out / qk_norm 的构建 ...
        # 解码各 rank 处理独立完整的 tile：复用 USPAttention 的后端分派，
        # 但显式绕过其序列并行 collectives（skip_sequence_parallel=True）
        self.attn = (
            USPAttention(
                num_heads=heads,
                head_size=dim_head,
                causal=False,
                skip_sequence_parallel=True,
            )
            if current_platform.is_cuda()
            else None
        )

    def forward(self, hidden_states, rotary_pos_emb=None):
        batch_size, seq_len, _ = hidden_states.shape
        qkv = self.to_qkv(hidden_states)
        qkv = qkv.view(batch_size, seq_len, -1, 3 * self.dim_head)
        query, key, value = torch.chunk(qkv, 3, dim=-1)
        # ... 此处省略 qk_norm 与 rotary_pos_emb 的应用 ...
        if self.attn is not None and query.dtype in (torch.float16, torch.bfloat16):
            hidden_states = self.attn(query, key, value)
        else:
            # FlashAttention 内核不接受 FP32；
            # 保留无 autocast 与 MPS 路径，避免后端选择改变精度契约
            hidden_states = _sdpa_attention(query, key, value)
        hidden_states = hidden_states.reshape(batch_size, seq_len, -1)
        return self.to_out(hidden_states)
```

python/sglang/multimodal_gen/runtime/models/vaes/minimax_h3_audio_vae/audio_vae.py

音频 VAE 的 CausalAttention 接入 USPAttention，新增 FA/TORCH_SDPA 后端准入与 FP32 精度契约管理，是音频侧的核心变更。

```
# MiniMax H3 音频 VAE 的因果注意力（DAC 谱系）
# 默认保持发布版 FP32 causal SDPA 契约；
# 仅当显式选择 fused 后端（如 audio_vae=fa）时才转换注意力计算精度

class CausalAttention(nn.Module):
    def __init__(self, in_dim, out_dim, num_heads):
        # … 省略 qkv 投影与 bias 参数构建 …
        self.attn = (
            USPAttention(
                num_heads=num_heads,
                head_size=self.head_dim,
                causal=True,
                # 音频 VAE 只接受 FA 与 TORCH_SDPA 两个后端
                supported_attention_backends={
                    AttentionBackendEnum.FA,
                    AttentionBackendEnum.TORCH_SDPA,
                },
                # 默认仍是 TORCH_SDPA，保证 FP32 行为不漂移
                default_attention_backend=AttentionBackendEnum.TORCH_SDPA,
                # 参考音频编码 rank-local，不参与序列并行
                skip_sequence_parallel=True,
            )
            if current_platform.is_cuda()
            else None
        )

    def forward(self, x):
        B, N, C = x.shape
        qkv = F.linear(
            input=x,
            weight=self.qkv.weight,
            bias=torch.cat((self.q_bias, self.zero_k_bias, self.v_bias)),
        )
        q, k, v = (
            qkv.reshape(B, N, 3, self.num_heads, self.head_dim)
            .permute(2, 0, 1, 3, 4)
            .unbind(0)
        )

        if self.attn is None:
            x = F.scaled_dot_product_attention(
                q.transpose(1, 2),
                k.transpose(1, 2),
                v.transpose(1, 2),
                attn_mask=None,
                dropout_p=0.0,
```

```
                is_causal=True,
            ).transpose(1, 2)
        else:
            input_dtype = q.dtype
            if self.attn.backend != AttentionBackendEnum.TORCH_SDPA:
                # 显式 fused 后端只负责注意力计算精度；输入输出仍按 FP32 交付
                q, k, v = (tensor.to(self.attn.dtype) for tensor in (q, k, v))
            x = self.attn(q, k, v).to(input_dtype)

        if self.in_dim > self.out_dim:
            x = torch.mean(x, dim=2)
            if self.in_dim // self.num_heads != self.out_dim:
                x = nn.functional.adaptive_avg_pool1d(x, self.out_dim)
        else:
            x = x.reshape(B, N, -1)
        x = self.proj(x)
        return x
```

python/sglang/multimodal_gen/runtime/layers/attention/layer.py

USPAttention 核心层：新增 default_attention_backend 参数，并让 skip_sequence_parallel 跳过 ring 能力检查，是支撑 VAE tile-local 解码的关键底层改动。

```
# USPAttention.__init__ 的关键片段：默认后端回退与 ring 准入
# default_attention_backend：仅在没有全局或组件覆盖时使用的回退
attn_backend = get_attn_backend(
    head_size,
    dtype,
    supported_attention_backends=supported_attention_backends,
    default_attention_backend=default_attention_backend,
    is_cross_attention=is_cross_attention,
)

# skip_sequence_parallel=True 表示各 rank 本地持有完整 KV（tile-local 解码），
# 不参与 Ulysses/Ring 序列并行，因此跳过全局 ring 能力准入检查
if not skip_sequence_parallel and get_ring_parallel_world_size() > 1:
    if not attn_backend.supports_ring_rotation():
        raise RuntimeError(
            "Ring Attention requires a backend whose kernel exposes the "
            "rotation primitive"
        )
```

评论区精华

该 PR 没有实质性的 code review 评论，评论区只有作者两次 /tag-and-rerun-ci 触发 CI 重跑。核心设计决策记录在 PR body 中：

> H3 decode 各 rank 处理独立完整的 tile，若在此应用全局 Ulysses/Ring collectives 会混合无关序列。

对应实现为 skip_sequence_parallel=True，并在 layer.py 中让该参数跳过 ring 能力准入检查。

> 发布版音频 VAE 保持 FP32 causal SDPA，只有显式 fused 后端才 cast QKV 计算精度。

对应实现为音频 VAE 默认 TORCH_SDPA、仅 FA 覆盖时转换并恢复 FP32 输出。

> H100 上 FA3 内核 1.57-1.75x 加速，但 E2E 解码 14.800 s vs 14.801 s，因此不宣称端到端加速。

作者明确接受该 PR 的收益在架构统一而非 H100 性能。

- VAE 注意力是否参与序列并行 collectives (design): 采用 skip_sequence_parallel=True，并在 ring world size > 1 时仅对非 skip 的实例做 ring 准入校验。
- 音频 VAE 的 FP32 精度契约 (correctness): 默认 SDPA，显式 fused 才 cast，输出恢复 FP32；配套测试验证 input_dtype 与 output.dtype。
- 删除 flash.py 中 unreachable 的 mask/varlen 分支 (design): 删除 190 行死代码，并在 __init__拒绝 t_causal，防止未来误用。
- E2E 性能收益 (performance): 接受：收益在架构统一与可配置性，而非 H100 上的 E2E 性能。

风险与影响

- 风险：具体风险点：
- 删除死代码的兼容性风险：flash.py 的 mask/varlen/pack_info 路径被整体删除，TransformerBlock.forward、forward_transformer_blocks 签名均去掉 pack_info。若仓库内仍存在依赖这些接口的未发布或训练路径，将直接破坏；PR 声称 released 解码器非因果，但调用方清理的完整性值得在合入后通过全量搜索确认。
- 精度契约依赖后端选择：视频 VAE 的 FP32 输入仍走 SDPA、CUDA FP16/BF16 走原生后端；音频 VAE 默认 FP32 SDPA，但显式 audio_vae=fa 时 QKV 会被 cast 到 attn.dtype 再恢复，可能存在数值漂移。测试只覆盖了 dtype 恢复，未验证数值等价。
- ROCm 回退逻辑变化：_FORCE_ROCM_MATH_SDPA 在 attention.py 模块导入时基于 torch.cuda.get_device_properties(0).gcnArchName 判断，若平台初始化时机不同（如 device 尚未就绪）可能误判；且该逻辑从原环境变量控制退化为硬编码 gfx95 判断，灵活性下降。
- 核心层接口变更的合并冲突：layer.py、selector.py 在 merge 提交 3343cafd 中出现冲突，说明这两个文件正被其他 PR（如 #34891）并行修改，存在后续合并或行为分叉风险。
- 跨平台性能不确定性：H100 上 E2E 无提升；在短序列、Blackwell、ROCm 等平台上 FA3/ 原生后端的选择可能产生不同性能或数值行为，缺少跨平台基准。
- 影响：影响范围：
- 用户 / 功能：MiniMax H3 视频与音频 VAE 现在可通过组件覆盖（如 audio_vae=fa）选择原生注意力后端；音频 VAE 默认仍为 FP32 SDPA，行为向后兼容；视频 VAE 在 CUDA FP16/BF16 下不再强行走 Torch SDPA，而是跟随全局 / 组件后端选择，H100 服务日志已确认 Attention backends for video_vae: fa。

- 系统：删除 190 行本地实现，减少重复代码；为后续其他 VAE 模块接入原生注意力提供可复用模式。
- 团队：新增 default_attention_backend 参数和 ring 准入规则变更，影响 diffusion 子系统所有使用 USPAttention 的模型（如 Wan、LTX），需要关注后端选择行为的一致性。
- 影响程度：中低；主要限于 diffusion 子系统的 MiniMax H3 路径，不涉及 SRT 核心推理路径。
- 风险标记：删除死代码的兼容性风险，FP32 精度契约依赖后端选择，ROCm gfx95 强制 math 回退，核心注意力层接口变更

关联脉络

- PR #34891 fix(diffusion): scope attention backend fallback: 与本 PR 修改同一个 diffusion 注意力后端选择机制（selector.py、layer.py），本 PR 的 merge 提交还出现了与 main 的冲突，两个改动需协同评审。
- PR #34264 config: decisions keyed on the attention backend read the configured pair: 注意力后端决策从 server_args 迁移到配置对，与本 PR 的 default_attention_backend 回退链设计相关。