PR #27531 完整报告

sgl-project/sglang

[Diffusion] Add SANA-WM with streaming support

合并时间：2026-06-09 01:10

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27531

执行摘要

- 一句话：为 SANA-WM 世界模型添加扩散、流式与实时推理支持
- 推荐动作：建议精读。该 PR 设计具有较高参考价值：自强迫去噪的流式框架、GDN/Softmax 混合注意力实现、实时链式状态管理。重点关注 streaming.py、self_forcing.py 和 realtime_chain.py 中的模式，便于后续集成其他世界模型。

功能与动机

为了在 SGLang 中支持 NVIDIA 发布的 SANA-WM 世界模型（Camera-Controlled Text-Image-to-Video），使系统能够运行密集双向推理和流式实时生成，满足交互式视频生成场景的需求。PR 描述虽未详细说明动机，但从实现看这是一项核心新能力的引入。

实现拆解

1. 模型定义与加载：在 python/sglang/multimodal_gen/runtime/models/dits/sana_wm.py 中定义 SanaWMBlock 和 SanaWMTransformer3DModel，实现 GDN（Gated DeltaNet）/ Softmax 混合注意力机制，支持双向和流式前向（forward/forward_long）。同时 sana_wm_components.py 导出大量复用组件。
2. Pipeline 配置与预处理：新增 python/sglang/multimodal_gen/configs/pipeline_configs/sana_wm.py 配置类 SanaWMPipelineConfig，处理文本后处理、latent shape 调整、帧数适配。python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/sana_wm/base.py 实现图像预处理、摄像机动作解析、旋转矩阵生成等辅助函数。
3. 流式去噪阶段：streaming.py 实现 SanaWMStreamingDenoisingStage，基于自强迫（Self-Forcing）机制逐块去噪，维护滚动 KV 缓存和 GDN 状态。核心采样逻辑在 self_forcing.py 的 SanaWMSelfForcingSampler 中，包括 create_autoregressive_segments 和 accumulate_kv_cache。
4. Refiner 流式封装：streaming_refiner.py 实现 LTX-2 Refiner 的分块流式推理，通过 KV 前缀和捕获机制解决块间依赖。sana_wm_refiner_transformer.py 定义 SanaWMRefinerBlock 和 SanaWMLTX2VideoRefiner，包含 pack_latents/unpack_latents 等工具。
5. 实时交互链：realtime_chain.py 实现完整的实时 Tick Pipeline（条件帧编码 → Latent 准备 → 摄像机条件 → 流式去噪 → Refiner → 因果解码），会话状态由多个 BaseRealtimeState 子类管理。realtime_stage.py 提供公共基类和 VAE 编码帮助。

sana_wm_realtime_adapter.py 处理 WebSocket 事件和状态维护。

6. JIT Kernel 优化：sana_wm_gdn_chunkwise.py 实现了 Triton 手写的逐块 GDN 扫描 Kernel（前向 / 双向），用于加速流式注意力中的 GDN 计算。
7. 测试与 CI：添加了 GPU 测试用例（后因太重被移除）、精度测试框架适配（accuracy_harness 跳过规则）、批处理客户端脚本。后续合入的提交增加了实时一致性测试和预览修复。

关键文件：

- python/sglang/multimodal_gen/runtime/models/dits/sana_wm.py（模块 模型定义；类别 source；类型 core-logic；符号 SanaWMBlock, SanaWMTransformer3DModel, forward, forward_long）：核心模型文件，定义 SanaWMBlock（GDN/Softmax 混合注意力）和 SanaWMTransformer3DModel，是整个 PR 的模型基础。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/sana_wm/base.py（模块 Pipeline 基础；类别 source；类型 data-contract；符号 sana_wm_pil_to_model_tensor, _sana_wm_rot_x, _sana_wm_rot_y, sana_wm_compute_resize_crop_geometry）：提供所有 SANA-WM pipeline 阶段共享的基础函数，包括图像预处理、摄像机动作解析、旋转矩阵、VAE 配置等，是整个 pipeline 的基石。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/sana_wm/streaming.py（模块 流式去噪；类别 source；类型 core-logic；符号 self_forcing_denoise_chunk, SanaWMStreamCacheState, SanaWMStreamingDenoisingStage, component_uses）：实现 SANA-WM 流式去噪阶段（SanaWMStreamingDenoisingStage），包含自强迫逐块去噪核心逻辑，是流式推理的关键。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/sana_wm/streaming_refiner.py（模块 Refiner 流式；类别 source；类型 core-logic；符号 set_kv_prefix_on_blocks, clear_kv_prefix_on_blocks, set_capture_flag_on_blocks, collect_captured_kv_from_blocks）：实现 LTX-2 Refiner 流式分块推理，通过 KV 前缀和捕获机制在块间传递隐藏状态，是质量增强的关键组件。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/sana_wm/self_forcing.py（模块 自强迫采样；类别 source；类型 core-logic；符号 SanaWMSelfForcingSamplerConfig, SanaWMSelfForcingSampler, create_autoregressive_segments, accumulate_kv_cache）：核心采样策略，封装 SanaWMSelfForcingSampler，管理流式去噪的自回归分段、KV 缓存累积与淘汰。
- python/sglang/jit_kernel/diffusion/triton/sana_wm_gdn_chunkwise.py（模块 JIT Kernel；类别 source；类型 performance；符号 _PhaseCfg, _ChunkwiseCfg, _phase_a_kv_kernel, _auto_config）：Triton 手写 GDN 分块 Kernel，优化流式注意力中的 GDN 计算，对性能至关重要。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/sana_wm/realtime_chain.py（模块 实时链；类别 source；类型 core-logic；符号 SanaWMSessionInputsState, SanaWMNoiseState, SanaWMRefinerChainState, SanaWMCondFrameEncodeStage）：实现完整的实时 Tick Pipeline，将离线流式阶段串

联为实时处理链，包括状态管理和 chunk 规划。

关键符号：SanaWMBlock.init, SanaWMBlock.forward, SanaWMBlock.forward_long, SanaWMTransformer3DModel.post_load_weights, self_forcing_denoise_chunk, SanaWMStreamingDenoisingStage.component_uses, SanaWMSelfForcingSampler.create_autoregressive_segments, SanaWMSelfForcingSampler.accumulate_kv_cache, SanaWMRealtimeAdapter.receive_camera_event_payload, set_kv_prefix_on_blocks, collect_captured_kv_from_blocks, pack_latents, unpack_latents

关键源码片段

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

核心模型文件，定义 SanaWMBlock（GDN/Softmax 混合注意力）和 SanaWMTransformer3DModel，是整个 PR 的模型基础。

```
# SPDX-License-Identifier: Apache-2.2
# Note: This is an excerpt from the newly added SANA-WM model file.
# The block implements a GDN (Gated DeltaNet) / Softmax hybrid attention.

class SanaWMBlock(nn.Module):
    """One transformer block of SANA-WM."""

    def __init__(
        self,
        hidden_size: int,
        num_heads: int,
        head_dim: int,
        mlp_ratio: float,
        t_kernel_size: int,
        qk_norm: bool,
        cross_norm: bool,
        conv_kernel_size: int,
        k_conv_only: bool,
        softmax_main: bool,
        use_chunk_plucker_post_attn: bool,
        chunk_size: Optional[int] = None,
        chunk_split_strategy: str = "uniform",
        update_rule: str = "torch_chunk",
        cam_update_rule: str = "torch_chunk",
        chunk_gdn_chunk_size: int = 21,
        use_chunked_softmax_attention: bool = False,
        gdn_backend: str = "auto",
    ) -> None:
        super().__init__()
        self.softmax_main = softmax_main
        self.chunk_size = chunk_size
        self.chunk_split_strategy = chunk_split_strategy
```

```
# Layer norm without learnable affine (handled by modulation)
self.norm1 = nn.LayerNorm(hidden_size, elementwise_affine=False, eps=1e-6)
self.norm2 = nn.LayerNorm(hidden_size, elementwise_affine=False, eps=1e-6)

# GDN/Softmax hybrid attention (BidirectionalGDNUCPESinglePathLiteLA)
self.attn = BidirectionalGDNUCPESinglePathLiteLA(
    in_dim=hidden_size,
    heads=num_heads,
    head_dim=head_dim,
    qk_norm=qk_norm,
    conv_kernel_size=conv_kernel_size,
    k_conv_only=k_conv_only,
    softmax_main=softmax_main,
    update_rule=update_rule,
    cam_update_rule=cam_update_rule,
    chunk_gdn_chunk_size=chunk_gdn_chunk_size,
    use_chunked_softmax_attention=use_chunked_softmax_attention,
    gdn_backend=gdn_backend,
)

self.cross_attn = MultiHeadCrossAttention(
    d_model=hidden_size,
    num_heads=num_heads,
    qk_norm=cross_norm,
)

self.mlp = GLUMBConvTemp(
    in_features=hidden_size,
    hidden_features=int(hidden_size * mlp_ratio),
    t_kernel_size=t_kernel_size,
)

# Scale-shift modulation table (6 params: scale_shift for norm1/attn/norm2/cross/mlp/
norm3)
self.scale_shift_table = nn.Parameter(
    torch.randn(6, hidden_size) / hidden_size**0.5
)

# Optional chunk plucker projection for post-attention summarization
if use_chunk_plucker_post_attn:
    self.plucker_proj = nn.Linear(hidden_size, hidden_size, bias=True)
    nn.init.zeros_(self.plucker_proj.weight)
    if self.plucker_proj.bias is not None:
        nn.init.zeros_(self.plucker_proj.bias)
else:
    self.plucker_proj = None

# ... (rest of __init__ )
```

python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/sana_wm/base.py

提供所有 SANA-WM pipeline 阶段共享的基础函数，包括图像预处理、摄像机动作解析、旋转矩阵、VAE 配置等，是整个 pipeline 的基石。

```
# SPDX-License-Identifier: Apache-2.0
# Note: This is an excerpt from the base stage module for SANA-WM.
# It contains camera action parsing, geometry computation, and utility functions.

# Constants for SANA-WM preprocessing
SANA_WM_TARGET_HEIGHT = 704
SANA_WM_TARGET_WIDTH = 1280
_SANA_WM_DEFAULT_TRANSLATION_SPEED = 0.04 # match official streaming
_SANA_WM_DEFAULT_ROTATION_SPEED_DEG = 1.2
_SANA_WM_ALLOWED_ACTION_KEYS = frozenset("wasdijkl") # movement keys


def sana_wm_pil_to_model_tensor(
    image: Image.Image, *, device: torch.device, dtype: torch.dtype
) -> torch.Tensor:
    """Convert PIL image to normalized model tensor (B, C, 1, H, W) in [-1, 1]."""
    arr = np.asarray(image, dtype=np.float32) / 255.0
    tensor = torch.from_numpy(arr).permute(2, 0, 1)
    return (tensor * 2.0 - 1.0).unsqueeze(0).unsqueeze(2).to(device=device, dtype=dtype)


def _sana_wm_rot_x(angle_rad: float) -> np.ndarray:
    """Rotation matrix around x-axis."""
    c, s = np.cos(angle_rad), np.sin(angle_rad)
    return np.array([[1.0, 0.0, 0.0], [0.0, c, -s], [0.0, s, c]], dtype=np.float64)


def _sana_wm_rot_y(angle_rad: float) -> np.ndarray:
    """Rotation matrix around y-axis."""
    c, s = np.cos(angle_rad), np.sin(angle_rad)
    return np.array([[c, 0.0, s], [0.0, 1.0, 0.0], [-s, 0.0, c]], dtype=np.float64)


def sana_wm_compute_resize_crop_geometry(
    src_w: int, src_h: int, target_h: int, target_w: int
) -> tuple[int, int, int, int]:
    """Compute resize/crop parameters to fit source into target aspect."""
    scale = max(target_h / float(src_h), target_w / float(src_w))
    resized_w = max(target_w, int(round(src_w * scale)))
    resized_h = max(target_h, int(round(src_h * scale)))
    left = (resized_w - target_w) // 2
    top = (resized_h - target_h) // 2
    return resized_w, resized_h, left, top
```

```
def parse_sana_wm_action_string(action: str) -> list[list[str]]:
    """Parse action string like 'w-5,i-3' into per-frame key lists."""
    cleaned = "".join(action.replace(",  ", ",").split())
    if not cleaned:
        raise ValueError("action string is empty")
    per_frame: list[list[str]] = []
    for segment in cleaned.split(","):
        if not segment or "-" not in segment:
            raise ValueError(f"invalid action segment {segment!r}; expected '<keys>-<frames>'")
        keys_part, duration_str = segment.rsplit("-", 1)
        duration = int(duration_str)
        if duration <= 0:
            raise ValueError(f"duration must be positive, got {duration}")
        # Normalize keys: keep only allowed characters
        keys = "".join(ch for ch in keys_part.lower() if ch in _SANA_WM_ALLOWED_ACTION_KEYS)
        for _ in range(duration):
            per_frame.append(list(keys))
    return per_frame
```

评论区精华

核心设计讨论集中在张量并行（TP）支持：

- sjmshsh询问 "TP 还不支持 "，认为需要支持 TP 策略。
- AgainstEntropy承认当时紧急添加流式与实时功能，尚未支持 TP，但认为需要，并请求 sjmshsh 移植后续优化。
- sjmshsh回复 OK 并提供了 PR #29513。 该讨论已达成结论：后续通过 #29513 添加 TP 支持。
- 是否支持张量并行（TP） (design): 决定后续通过 PR #29513 添加 TP 支持。

风险与影响

- 风险：
  1. 代码量大且审查不足：新增 17k+ 行代码，review 评论仅 6 条，核心逻辑可能未经充分审查，存在隐藏缺陷。
  2. TP 支持缺失：当前不支持张量并行，大规模部署时显存瓶颈明显，扩展性受限。
  3. Triton Kernel 兼容性：GDN 分块 kernel 依赖特定 GPU 架构（SM90+），在较老 GPU 或 AMD/Intel 平台上可能回退或报错。
  4. 实时状态管理复杂度：SanaWMRealtimeChain 涉及多阶段状态同步和内存管理，高并发下可能出现内存泄漏或延迟抖动。
  5. 测试覆盖有限：最初的 GPU CI 测试因太重被移除，目前仅保留轻量级单元测试和实时一致性测试，缺少端到端长期运行测试。 - 影响：用户：新增 SANA-WM 世界模型支持，用户可通过离线批处理或实时 WebSocket 接口生成摄像机可控视频，但需要至少 80GB+ 显存（1600M 模型）。系统：扩展了 multimodal_gen 模块的模型和 pipeline 生态，新增 sana_wm 子包，对框架可扩展性有积极贡献。JIT kernel 目录增加扩散专用

Triton kernel。团队：需维护大量新代码，尤其是 GDN 混合注意力和实时链逻辑。后续 TP 支持和集成测试需跟进。

- 风险标记：新模型集成未充分测试，TP 支持缺失，Triton Kernel 兼容性，实时状态管理复杂度，缺乏端到端长期运行测试

关联脉络

- PR #26153 Unknown (co-authored with sjmshsh, mentioned in discussion): 作者提到此 PR 基于 #26153 的一些先前更改（与 sjmshsh 共同创作）。
- PR #29513 [SANA-WM] TP support (provided by sjmshsh): sjmshsh 在讨论中提供的后续 PR，用于添加 TP 支持。
- PR #28624 [diffusion] optimize LTX2.3 CFG/SP paths: 同为 diffusion 模块的性能 / 重构 PR，部分优化思路可能与此 PR 的 Refiner 相关。