# PR #26954 完整报告

sgl-project/sglang

[diffusion] misc

合并时间：2026-06-02 13:52

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26954

## 执行摘要

- 一句话：新增 LingBot World 实时扩散管道与 WebUI 支持
- 推荐动作：此 PR 属于里程碑级功能合并，架构设计值得精读：特别是 *CausalSelfAttentionKVCache* 的抽象和 *CausalDMDDenoisingStage* 的通用化处理。对于参与 diffusion 或实时推理的开发者，建议重点阅读 *causal_denoising.py*、*causal_attention_cache.py* 和 lingbot_world.py，了解因果视频生成的流水线设计。对于仅使用批处理推理的用户，此 PR 无直接影响，但未来可复用其基础架构。

## 功能与动机

根据关联 Issue #19159（实时 diffusion）和 #26756（实时视频会话与适配器抽象），本 PR 的目标是支持低延迟因果视频生成。关键需求包括：因果 DiT 支持、滚动 KV 缓存、KV 重计算、sink、可变首块、实时会话运行时（流式输入输出）、条件管理（条件队列、插值、状态缓存）以及编码器边界处理。此 PR 是这些需求的初始实现，聚焦于 LingBot World 模型。

## 实现拆解

1. 实时运行时代理与状态管理：在 python/sglang/multimodal_gen/runtime/entrypoints/openai/realtime/ 下新增 realtime_video_api.py，定义 WebSocket 路由 /v1/realtime_video，以及 _generate_loop、_send_output_and_log、_listen_events 等异步协程。同时通过 BaseRealtimeState 派生子类（如 RealtimeInputValidationState、RealtimeTextState、RealtimeVAEState）实现跨 Chunk 的会话级缓存。
2. 因果 DMD 去噪流水线：在 python/sglang/multimodal_gen/runtime/pipelines_core/stages/ 下新增 causal_denoising.py（大幅修改，新增 945 行），提供 CausalDMDDenoisingStage 基类，管理因果 KV 缓存策略、序列分片、Cache 更新等。专属子类 LingBotWorldCausalDMDDenoisingStage（lingbot_world_causal_denoising.py）实现了 LingBot World 特有的 36 通道输入拼接、KV 缓存大小计算、序列分片开关等功能。
3. 因果注意力与 KV 缓存层：新增 python/sglang/multimodal_gen/runtime/layers/kvcache/causal_attention_cache.py，定义 CausalSelfAttentionKVCache 和 CrossAttentionKVCache，支持滚动窗口、整数索引读写、grow-to-fit 分配，为因果视频生成提供高效缓存。同时新增模型定义 lingbot_world.py（1360 行），包含 LingBotWorldCamConditioner、LingBotWorldCausalSelfAttention、LingBotWorldTransformerBlock 等核心模块，利用序列分片（_compute_sequence_splits、_sequence_shard_tensor）支持分布式推理。

4. 实时 VAE 与文本编码：新增 realtime_vae.py，包含 RealtimeImageVAEEncodingStage（复用首块条件潜变量）、CausalVaeDecodingStage（支持 WAN 解码器持久缓存）。新增 realtime_text_encoding.py，通过 RealtimeTextEncodingStage 缓存文本编码器输出，避免重复计算。新增 realtime_input_validation.py，缓存验证后的条件图像和随机数生成器。
5. 条件事件与输出适配：新增 runtime/realtime/condition_events.py，定义 ConditionEventQueue、ControlSignal 等，管理条件采样参数和事件队列。新增 realtime_output_adapter.py，实现帧分批、头打包、原始 /JPEG 传输等。同时添加实时 WebUI（apps/realtime_webui/）作为演示接口。
6. 测试与配置：新增大量单元测试和集成测试（至少 16 个测试文件），覆盖因果 VAE、实时会话、帧传输等场景。在 python/sglang/multimodal_gen/configs/pipeline_configs/lingbot_world.py 中添加 LingBot World 专属流水线配置，包括相机状态管理、动作验证等。

关键文件：

- python/sglang/multimodal_gen/runtime/models/dits/lingbot_world.py（模块 模型定义；类别 source；类型 data-contract；符号 _compute_sequence_splits, _sequence_shard_tensor, _sequence_all_gather_varlen, LingBotWorldCamConditioner）：新增的 LingBot World 模型定义，包含因果自注意力、相机条件注入、序列分片等核心模块，是整个实时流水线的计算核心。
- python/sglang/multimodal_gen/runtime/entrypoints/openai/realtime/realtime_video_api.py（模块 实时接口；类别 source；类型 entrypoint；符号 _transport_ms, _wait_for_active_session_slot, _log_realtime_chunk_timing, _send_realtime_chunk_stats）：实时视频生成的主入口，定义 WebSocket 路由、会话管理、块统计日志、帧发送循环等。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/lingbot_world/lingbot_world_causal_denoising.py（模块 去噪流水线；类别 source；类型 data-contract；符号 LingBotWorldCausalDMDDenoisingStage, _get_causal_kv_cache_size, _causal_sequence_shard_enabled, _num_causal_cache_attention_heads）：LingBot World 的因果 DMD 去噪阶段，继承基类并定制了 KV 缓存大小、序列分片、注意力头数等关键参数。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/realtime_vae.py（模块 VAE 解码；类别 source；类型 dependency-wiring；符号 RealtimeVAEState, init, dispose, RealtimeVAEDecodeState）：实现了实时 VAE 编码 / 解码的状态管理和因果缓存，首块条件潜变量复用以提升效率。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/causal_denoising.py（模块 因果去噪基类；类别 source；类型 dependency-wiring；符号 CausalDMDForwardContext, CausalDMDCachePolicy, CausalDMDRealtimeCacheContext, forward）：因果 DMD 去噪的基类，本 PR 对其进行了重大重构（+945/-276），提取了通用缓存策略、序列分片接口、前向准备逻辑等，供所有实时模型复用。
- python/sglang/multimodal_gen/runtime/entrypoints/openai/realtime/realtime_output_adapter.py（模块 输出适配；类别 source；类型 core-logic；符号 RealtimeFrameBatchHeader, RealtimeFrameBatchMessage,

RealtimeFrameSendStats, empty_frame_send_stats）：实时帧输出适配器，负责将模型输出打包为帧批次、构造传输头和统计信息，支持多种内容类型。

关键符号：_compute_sequence_splits, _sequence_shard_tensor, _sequence_all_gather_varlen, LingBotWorldCamConditioner.forward, LingBotWorldCausalSelfAttention.forward, CausalVaeDecodingStage.decode_causal, CausalVaeDecodingStage._decode_wan_with_persistent_cache, RealtimeTextEncodingStage._restore_cached_outputs, RealtimeInputValidationStage._can_reuse_cached_image, ConditionEventQueue.init, CausalDMDDenoisingStage.forward

## 关键源码片段

### python/sglang/multimodal_gen/runtime/entrypoints/openai/realtime/realtime_video_api.py

实时视频生成的主入口，定义 WebSocket 路由、会话管理、块统计日志、帧发送循环等。

```
import asyncio
import time
import msgspec.msgpack
from fastapi import APIRouter, WebSocket

# 将毫秒值转换为整数，保证非负
def _transport_ms(value: float) -> int:
    return max(0, int(value + 0.5))

# 等待活跃会话槽释放，最多等 15 秒
async def _wait_for_active_session_slot(
    *,
    timeout_s: float = 15.0,
    interval_s: float = 0.1,
) -> bool:
    deadline = time.monotonic() + timeout_s
    while _ACTIVE_SESSION_IDS and time.monotonic() < deadline:
        await asyncio.sleep(interval_s)
    return not _ACTIVE_SESSION_IDS

# 记录每个 Chunk 的详细时序到日志
def _log_realtime_chunk_timing(
    session: GenerateSession,
    chunk: RealtimeChunkContext,
    batch: "Req",
    request_prepare_ms: float,
    scheduler_forward_ms: float,
    chunk_total_ms: float,
    send_stats: RealtimeFrameSendStats,
) -> None:
    logger.info(
```

```
        "realtime chunk timing: session_id=%s ... chunk_total=%.2fms ...",
        session.id, chunk.request_id, batch.block_idx,
        getattr(batch, "realtime_event_id", None),
        sorted(batch.condition_inputs or []),
        request_prepare_ms, scheduler_forward_ms,
        send_stats["header_pack_ms"], send_stats["header_write_ms"],
        send_stats["raw_payload_build_ms"], send_stats["raw_write_ms"],
        send_stats["ws_write_ms"], chunk_total_ms,
        send_stats["num_batches"], send_stats["num_frames"],
        send_stats["frame_shape"], send_stats["raw_bytes"],
        send_stats["ws_payload_bytes"], send_stats["content_type"],
    )
```

## python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/lingbot_world/lingbot_world_causal_denoising.py

LingBot World 的因果 DMD 去噪阶段，继承基类并定制了 KV 缓存大小、序列分片、注意力头数等关键参数。

```
class LingBotWorldCausalDMDDenoisingStage(CausalDMDDenoisingStage):
    """
    LingBot-World 因果 DMD 去噪阶段
    输入为 [noise(16ch), condition(20ch)] 拼接的 36 通道张量
    """
    # 计算因果 KV 缓存大小：local_attn_size 或滑动窗口帧数
    def _get_causal_kv_cache_size(self, *, sequence_shard_enabled: bool = False) -> int:
        if self.local_attn_size != -1:
            return self.local_attn_size * self.num_token_per_frame
        return self.sliding_window_num_frames * self.num_token_per_frame

    # 判断是否启用序列分片（Ulysses > 1 且 batch 标记允许）
    def _causal_sequence_shard_enabled(self, batch: Req) -> bool:
        return bool(
            getattr(batch, "enable_sequence_shard", False)
            and get_ulysses_parallel_world_size() > 1
        )

    # 当序列分片启用时，注意力头数根据 Ulysses 世界大小整除
    def _num_causal_cache_attention_heads(self, *, sequence_shard_enabled: bool) -> int:
        if not sequence_shard_enabled:
            return self.transformer.num_attention_heads
        ulysses_world_size = get_ulysses_parallel_world_size()
        if get_ring_parallel_world_size() > 1:
            raise NotImplementedError("ring_degree > 1 暂不支持")
        if ulysses_world_size <= 1:
            raise ValueError("需要 ulysses_degree > 1")
        if self.transformer.num_attention_heads % ulysses_world_size != 0:
            raise ValueError("注意力头数必须能被 ulysses_degree 整除")
        return self.transformer.num_attention_heads // ulysses_world_size
```

```
    # 输入验证：检查 image_latent, latents, timesteps, scheduler, prompt_embeds
    def verify_input(self, batch: Req, server_args: ServerArgs) -> VerificationResult:
        result = VerificationResult()
        result.add_check("image_latent", batch.image_latent, [V.is_tensor, V.with_dims(5)])
        result.add_check("latents", batch.latents, [V.is_tensor, V.with_dims(5)])
        result.add_check("timesteps", batch.timesteps, [V.is_tensor, V.with_dims(1)])
        result.add_check("scheduler", batch.scheduler, V.not_none)
        result.add_check("prompt_embeds", batch.prompt_embeds, V.list_not_empty)
        return result
```

## 评论区精华

本 PR 的 Review 讨论极少（仅 Gemini Code Assist 的配额警告和作者 /tag-and-rerun-ci 命令），不存在实质性的技术辩论。鉴于作者同时是合并者且 PR 无 review comments，可能属于内部直接合并。因此没有可提炼的讨论交锋。

- 暂无高价值评论线程

## 风险与影响

- 风险：
  1. 性能风险：实时流水线涉及多次跨进程通信（WebSocket、IPC）和频繁的 KV 缓存读写，可能在高并发场景下出现延迟抖动或内存压力。特别是 CausalVaeDecodingStage._decode_wan_with_persistent_cache 逐帧解码存在计算瓶颈。
  2. 兼容性风险：新增的模型定义 lingbot_world.py 依赖特定版本的 aiter、fastvideo 等第三方库，若依赖不一致可能导致模型加载失败。配置键（如 enable_sequence_shard）需要与前端控制器严格同步。
  3. 稳定性风险：实时会话维护大量可变状态（RealtimeInputValidationState、RealtimeTextState、RealtimeVAEState），若 dispose 流程未正确清理可能导致内存泄漏。WebSocket 断开重连逻辑尚不够健壮（如 _wait_for_active_session_slot 可能死等）。
  4. 安全性风险：WebUI 和 WebSocket API 未添加身份验证，生产部署需前置认证层。 - 影响：用户影响：新增实时视频生成能力，用户可通过 WebSocket API /v1/realtime_video 或内置 WebUI 体验低延迟的视频流生成。需要部署支持 LingBot World 模型的推理服务。系统影响：增加了进程启动时间（加载新的 VAE、Transformer 模型）和运行时内存占用（因果 KV 缓存、VAE 缓存）。对现有 diffusion 批处理流程无影响，因为新代码位于独立模块和路由下。团队影响：需要维护新模块的代码质量、更新文档（cookbook 已新增 LingBot World 部分）、提供客户支持。对于后续实时 diffusion 模型（如 Cosmos3）的集成，本 PR 提供了可复用的基类（CausalDMDDenoisingStage、BaseRealtimeState）。
- 风险标记：新模块核心变更，缺少 review 讨论，大量配置整合，无生产认证

## 关联脉络

- PR #26959 [diffusion] add WebUI: 与本 PR 同属 diffusion 实时功能线，共享 WebUI 组件和实时接口设计。当前 PR 进一步扩展了 WebUI 与 LingBot 运行时集成。
- PR #26973 [diffusion] reduce Cosmos3 denoise overhead: Cosmos3 去噪优化与本 PR 的因果 DMD 架构使用相似的因果 KV 缓存技术，两者在缓存策略上可能互相借鉴。