# PR #24931 完整报告

sgl-project/sglang

feat(mimo-v2): add EPD disaggregation support

合并时间：2026-05-18 10:33

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24931

## 执行摘要

- 一句话：MiMo-V2 添加 EPD 编码器分离支持
- 推荐动作：PR 设计合理，关键讨论已解决，CI 通过。建议合并，并后续补充自动化测试以覆盖 EPD 端到端流程。值得关注的设计点包括编码服务器钩子模式和按模型类型属性拆分，可作为其他模型接入 EPD 的范例。

## 功能与动机

MiMo-V2 的视觉 / 音频编码器与 LLM 骨干计算特性差异大，物理部署在同一节点会导致 TP/DP 尺寸一刀切，浪费空闲组件 GPU 内存。EPD 让运营商在成本较低的 GPU 上运行编码器（自有 TP），在首选 TP/DP 拓扑上运行语言模型，并根据请求混合独立扩展两层。

## 实现拆解

1. 模型分离：在 MiMoV2ForCausalLM.__init__ 中根据 config.encoder_only / config.language_only 跳过语言模型（model、lm_head、LogitsProcessor），但始终保留视觉 / 音频编码器作为本地 fallback。添加权重路由前缀常量 _LANGUAGE_WEIGHT_PREFIXES 和 _VISION_AUDIO_WEIGHT_PREFIXES，load_weights 按角色过滤权重。
2. 编码器预处理：MiMoProcessor 增加 from_hf_config 类方法（无需 tokenizer）和 preprocess_for_encoder 方法，处理图像 / 视频 / 音频到模型输入张量。新增 _decode_frames_and_timestamps、_ffprobe_has_audio 等共享工具。
3. 编码服务器重构：将原先单一 _process_mm_items 拆分为 _process_image_items、_process_video_items、_process_audio_items 三个助手。若模型定义了 preprocess_mm_for_encoder 则委托模型预处理，否则回退到原有 per-modality 逻辑。新增 encode_video_audio 钩子处理视频中的音频轨道。
4. 视频元数据扩展：encode_receiver.py 中拆分通用（_GENERAL_VIDEO_META_ATTRS）和 MiMo 特有（_MIMO_VIDEO_AUDIO_META_ATTRS）视频元数据属性，通过 video_meta_attrs_for 按 model_type 动态解析，避免非 MiMo 模型引入额外字段。
5. SHM 包装器泛化：mm_utils.py 中提取 _wrap_tensor_or_list 和 _unwrap_tensor_or_list 以支持 precomputed_embeddings 的共享内存传输，使 EPD 数据流兼容既有 SHM 机制。

6. 启动配置调整：server_args.py 中跳过 MiMoV2 encoder_only 模式的 fused-QKV TP 检查，并将 MiMoV2ForCausalLM 加入编码器分离允许列表。 测试方面，PR 未包含自动化测试，但列出了详细的测试计划（单节点回归、端到端 EPD 验证等）。

关键文件：

- python/sglang/srt/models/mimo_v2.py（模块 模型定义；类别 source；类型 data-contract；符号 preprocess_mm_for_encoder, get_input_embeddings, encode_video_audio, _as_tensor）：实现 MiMo-V2 模型在 EPD 模式下的构造分离，新增编码器预处理钩子和权重路由过滤
- python/sglang/srt/multimodal/processors/mimo_v2.py（模块 预处理；类别 source；类型 dependency-wiring；符号 _decode_frames_and_timestamps, _ffprobe_has_audio, from_hf_config, _as_dict）：新增 from_hf_config 和 preprocess_for_encoder 方法，支持无 tokenizer 的编码器预处理
- python/sglang/srt/disaggregation/encode_server.py（模块 编码服务；类别 source；类型 dependency-wiring；符号 _build_mm_aux_data, _process_image_items, _process_video_items, _process_audio_items）：重构 _process_mm_items 为按模态划分的助手，引入模型侧钩子 preprocess_mm_for_encoder 和 encode_video_audio
- python/sglang/srt/disaggregation/encode_receiver.py（模块 接收逻辑；类别 source；类型 core-logic；符号 video_meta_attrs_for, from_embedding_data）：拆分通用和 MiMo 视频元数据属性，动态按 model_type 解析，确保非 MiMo 模型不引入额外字段
- python/sglang/srt/managers/mm_utils.py（模块 工具函数；类别 source；类型 core-logic；符号 _wrap_tensor_or_list, _unwrap_tensor_or_list）：提取 _wrap_tensor_or_list 和 _unwrap_tensor_or_list 以支持 precomputed_embeddings 的 SHM 传输，为 EPD 提供必要的数据传递工具
- python/sglang/srt/server_args.py（模块 启动配置；类别 source；类型 core-logic）：跳过 MiMo-V2 encoder_only 模式的 fused-QKV TP 检查，将 MiMoV2 加入 EPD 允许列表
- python/sglang/srt/managers/tokenizer_manager.py（模块 管理服务；类别 source；类型 core-logic）：少量修改以支持 EPD 相关的请求处理

关键符号：preprocess_mm_for_encoder, encode_video_audio, from_hf_config, preprocess_for_encoder, video_meta_attrs_for, _process_image_items, _process_video_items, _process_audio_items, _wrap_tensor_or_list, _unwrap_tensor_or_list

## 关键源码片段

### python/sglang/srt/disaggregation/encode_receiver.py

拆分通用和 MiMo 视频元数据属性，动态按 model_type 解析，确保非 MiMo 模型不引入额外字段

```
def video_meta_attrs_for(model_type: Optional[str]) -> tuple:
    # 根据 model_type 返回视频元数据属性元组
    # 所有模型共有的属性
    general = ("video_timestamps", "second_per_grid_ts")
```

```
    if model_type and "mimo" in model_type.lower():
        # MiMo 模型额外包含音频 - 视频对齐属性
        general += (
            "video_audio_feature_lens",
            "video_audio_segment_lens_flat",
            "video_audio_per_video_num_units",
            "video_audio_embedding",
        )
    return general
```

### python/sglang/srt/managers/mm_utils.py

提取 _wrap_tensor_or_list 和 _unwrap_tensor_or_list 以支持 precomputed_embeddings 的 SHM 传输，为 EPD 提供必要的数据传递工具

```
def _wrap_tensor_or_list(value):
    # 包装 CPU tensor 或 tensor 列表为 ShmPointerMMData
    # 用于跨进程共享内存传输
    if isinstance(value, torch.Tensor) and value.is_cpu:
        return ShmPointerMMData(value)
    elif isinstance(value, (list, tuple)):
        wrapped = [
            (ShmPointerMMData(t) if isinstance(t, torch.Tensor) and t.is_cpu else t)
            for t in value
        ]
        return type(value)(wrapped) if isinstance(value, tuple) else wrapped
    return value


def _unwrap_tensor_or_list(value):
    # 从 ShmPointerMMData 恢复为普通 tensor
    if isinstance(value, ShmPointerMMData):
        return value.materialize()
    elif isinstance(value, (list, tuple)):
        unwrapped = [
            t.materialize() if isinstance(t, ShmPointerMMData) else t
            for t in value
        ]
        return type(value)(unwrapped) if isinstance(value, tuple) else unwrapped
    return value
```

## 评论区精华

- 视频元数据属性拆分：ShangmingCai 指出原 _VIDEO_META_ATTRS 注释误导非 MiMo 模型，Abatom 将其拆分为通用和 MiMo 特有组，通过 video_meta_attrs_for 动态选择。
- 编码服务器中视频音频编码抽象：ShangmingCai 建议将视频中音频编码逻辑抽取为模型侧钩子，Abatom 实现 encode_video_audio 方法并重命名以符合现有约定。
- Embedding 拼接设备匹配：ShangmingCai 询问移除 .cuda() 和 .to("cpu") 是否引入设备问题，Abatom 解释所有 tensor 均为 CPU，直接拼接安全，原操作为性能优化非必需。

- 视频元数据属性命名与跨模型兼容性 (design): Abatom 将属性拆分为 _GENERAL_VIDEO_META_ATTRS 和 _MIMO_VIDEO_AUDIO_META_ATTRS，并通过 video_meta_attrs_for 函数按 model_type 动态拼接，解决混淆。
- 视频音频编码应抽取为模型侧钩子 (design): Abatom 将该部分抽出为 model-side 的 encode_video_audio 方法，服务器仅做调度，并重命名以符合现有约定。
- Embedding 拼接时的设备匹配问题 (correctness): Abatom 解释所有 embedding 均为 CPU tensor（来自 ZMQ/Mooncake 链路），因此直接 cat 安全；之前 .cuda() 是性能优化，非必要。

## 风险与影响

- 风险：权重路由可能因前缀匹配不完整导致加载错误，但已通过 is_vision_audio_weight 等辅助方法防御。MiMo-V2 特定视频元数据字段（video_audio*）若未在 decode 端正确重构，可能引发维度不匹配，动态属性机制降低了跨模型污染。编码服务器中 fallback 路径（Qwen2-VL、Kimi 等）依赖重构后的 per-modality 助手，回归风险较小但需验证。mm_utils.py 中 SHM 包装新增对 precomputed_embeddings 的处理，可能与其他模型交互时产生意料外的包装，但已有显式类型检查。整体缺少自动化测试覆盖，依赖手动验证计划。
- 影响：用户启用 --encoder-only 或 --language-only 即可部署 MiMo-V2 EPD，实现灵活资源配置。编码服务器新增模型侧钩子接口，未来模型无需修改编码服务器即可支持 EPD；但需确保模型实现 preprocess_mm_for_encoder 和 encode_video_audio 等契约。团队需维护和文档化钩子契约及 MiMo-V2 特有视频元数据属性。
- 风险标记：核心路径变更，缺少测试覆盖，模型兼容性，数据契约变更

## 关联脉络

- 暂无明显关联 PR