PR #33829 完整报告

sgl-project/sglang

[Model] Complete dots.note.omni support with native encoders, video preprocessing, and MTP decoding

合并时间：2026-08-22 14:19

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33829

执行摘要

- 一句话：完整集成 dots.note.omni 多模态模型与 MTP 推测解码
- 推荐动作：值得精读，尤其对维护多模态与推测解码框架的工程师。重点阅读 dots_hybrid_backend.py 的 DP padding 元数据 normalize 逻辑、pool_configurator.py 的 SWA 草稿层记账，以及 nextn.py 的全共享 MTP 实现。同时需要关注作者在评论中提到的 follow-up PR（#36195）对 DSA 默认后端兼容性的修复，当前合并状态下该配置存在已知缺陷。

功能与动机

dots.note.omni 在几个关键假设上不符合现有多模态与推测解码路径：视频编码必须匹配训练时的 flatten 算法，PR body 明确写到“Applying a generic uniform-frame video processor changes the modality ordering, sampling strategy, and token allocation, which causes an inference/training mismatch”；语言模型全注意力层与滑窗层（MLA/DSA 注意力门控）不能共享同一注意力路径与 KV 记账；MTP 结构也不同于标准单层 EAGLE/NextN。因此需要原生编码器、定制视频预处理与新的混合 SWA 注意力后端。

实现拆解

1. 原生多模态编码器迁移：新增 python/sglang/srt/models/dots3_common/dots_omni_audio.py 与 dots_omni_vision.py，分别移植 Whisper 派生语音编码器（DotsWhisperConfig、RotaryEmbedding、compute_audio_token_length）和 MoE ViT（DotsMoEVitConfig、VisionAttention、VisionRotaryEmbedding）；dots_omni_towers.py 提供 DotsNoteOmniVisionEncoder/DotsNoteOmniAudioEncoder 直接加载模型目录权重，并内置 CPU 图像预处理器 DotsNoteOmniImagePreprocessor。这些模块移除了对 mm_encoder_server 的运行时依赖，并遵循 --language-only 拆分编码器与 LLM。
2. 训练一致的视频预处理：新增 python/sglang/srt/multimodal/processors/dots_note_omni_video_core/ 包（preprocess.py、flatten_runner.py、v2core.py、video_qa_flattener.py），vendor 训练时的 flatten 算法，通过 torchcodec 在内存中解码远程 video_url，按 token 预算交织采样帧、时间戳与音频段；dots_note_omni.py 处理器与 OpenAI chat API 集成，支持 request-scoped 的 seq、audio_cap、audio_sr、k_mode 控制。

3. 混合注意力与模型实现：python/sglang/srt/configs/dots3.py 定义 Dots3Config 并负责 draft 配置重写与注意力后端包装；python/sglang/srt/layers/attention/dots_hybrid_backend.py 新增 DotsSWAMLAAttnBackend，处理 SWA 层的 prefill/decode 元数据并在 DP padding 后重建注意力元数据；python/sglang/srt/models/dots3_common/nextn.py 新增全共享 MTP 草稿模型 Dot3NoteModelNextN（单层递归复用的 Dots3MTPHead，约束为 sliding_attention 几何）。
4. 共享基础设施修正：pool_configurator.py 在混合池字节预算中区分 full-attention、普通 SWA 与 full-capacity SWA 草稿层；model_runner.py 修复 DP+MTP eager draft 多步中 mlp_sync_prepared 标志残留；spec_aux_hidden_state.py 为 DeepSeek V4 架构补充 is_hybrid_swa 属性保护；dsa_indexer.py 修复 K-only rotary 的输入别名；eagle_draft_cuda_graph_runner.py 在 CUDA graph 重放时使用 bucket-padded cache-location 缓冲区。
5. 测试、文档与部署：新增 / 更新 test_pool_configurator.py、test_dsa_indexer.py、dots 检测器 unittest 等测试；新增 RedNote cookbook 与 Dots3-Note.sh 启动脚本；dots_detector.py 实现 dots 工具调用与推理解析。

关键文件：

- python/sglang/srt/models/dots3_common/dots_omni_audio.py（模块 音频编码；类别 source；类型 data-contract；符号 DotsWhisperConfig, RMSNorm, RotaryEmbedding, compute_audio_token_length）：新增 1027 行的 Whisper 派生语音编码器，定义 DotsWhisperConfig 与音频 token 长度计算，是原生多模态编码器迁移的核心。
- python/sglang/srt/models/dots3_common/dots_omni_vision.py（模块 视觉编码；类别 source；类型 data-contract；符号 VisionRotaryEmbedding, VisionAttention, _map_qkv_weight, DotsMoEVitConfig）：新增 769 行的 MoE ViT 视觉编码器，实现 VisionAttention 包装、2D 视觉 RoPE 与 DotsMoEVitConfig，是原生视觉塔的核心。
- python/sglang/srt/models/dots3_common/nextn.py（模块 草稿模型；类别 source；类型 data-contract；符号 Dots3MTPHead, Dot3NoteModelNextN, forward, _embed_input_ids）：新增 204 行的全共享 MTP/NextN 草稿模型，是 dots3 推测解码的核心，约束为 sliding_attention 并复用 Dots3DecoderLayer。
- python/sglang/srt/layers/attention/dots_hybrid_backend.py（模块 注意力后端；类别 source；类型 dependency-wiring；符号 DotsSWAMLAAttnBackend, DotsSWAMLAPrefillMetadata, normalize_forward_metadata_for_dp_padding, _normalize_page_table_rows）：新增 571 行的 dots 专属混合 SWA/MLA 注意力后端，处理 DP padding 后的元数据重建与 SWA page table 对齐，是推测解码正确性的关键。
- python/sglang/srt/model_executor/pool_configurator.py（模块 池配置；类别 source；类型 core-logic；符号 fixed_swa_bytes, _draft_swa_layers_num, calculate_pool_sizes, calculate_pool_sizes_from_max_tokens）：修改混合池配置器，将 EAGLE/NextN 草稿层按 SWA 几何分别记账，避免目标 KV 容量浪费或不足，影响所有 hybrid-SWA 模型。
- python/sglang/srt/multimodal/processors/dots_note_omni_video_core/preprocess.py（模块 视频预处理；类别 source；类型 dependency-wiring；符号 tokenize_len, audio_block_tokens, conversation_tokens, _make_video_decoder）：新增 338 行的训练一致视频预处理入口，负责 token 预算估计、帧 / 音频抽取与解码，是 video_url 端到端处

理的核心。

- python/sglang/srt/multimodal/processors/dots_note_omni.py（模块 多模态处理；类别 source；类型 dependency-wiring；符号 DotsNoteOmniProcessor, preprocess_dots_video, _build_video_cfg, _flat_video_to_content）：新增 565 行的多模态处理器，将视频 flatten 管线接入 OpenAI chat API，生成训练一致的 interleaved 内容。
- python/sglang/srt/configs/dots3.py（模块 模型配置；类别 source；类型 dependency-wiring；符号 Dots3Config, configure_draft_model, wrap_attention_backend, DotsNoteOmniTokenizerProxy）：新增 243 行的 Dots3 配置与草稿模型重写逻辑，是模型注册、attention 后端包装和 draft loading 的入口。

关键符号：DotsWhisperConfig.init, RotaryEmbedding.get_cos_sin, VisionRotaryEmbedding._compute_freqs, VisionAttention._map_qkv_weight, DotsNoteOmniVisionEncoder.load_converted_state, DotsNoteOmniAudioEncoder.load_converted_state, DotsNoteOmniImagePreprocessor._resized_size, Dots3MTPHead.init, Dot3NoteModelNextN.forward, Dots3NoteForCausalLMNextN.load_weights, DotsSWAMLAAttnBackend.normalize_forward_metadata_for_dp_padding, load_omni_component_config, audio_block_tokens, conversation_tokens, extract_frames_v2, build_plan

关键源码片段

python/sglang/srt/models/dots3_common/dots_omni_audio.py

新增 1027 行的 Whisper 派生语音编码器，定义 DotsWhisperConfig 与音频 token 长度计算，是原生多模态编码器迁移的核心。

```
"""Dots 路径的语音编码器（仅推理，单 GPU）。

从 cybertron_alm 的 dots_audio_encoder/modeling_whisper.py 移植，
上行 WhisperEncoder 以 DotsSpeechEncoder 名义暴露。
"""

from functools import lru_cache
from typing import ClassVar
import math
import torch
from torch import nn
from transformers.models.whisper.configuration_whisper import WhisperConfig


class DotsWhisperConfig(WhisperConfig):
    """Dots 编码器所需字段的 Whisper 配置扩展。"""

    def __init__(self, *, use_causal=False, use_rms_norm=False,
                 use_latent_input=False, use_conv2d_stem=False,
                 latent_dim=None, downsample_hidden_size=480, use_rope=False,
                 rope_parameters=None, conv_chunksize=500, **kwargs):
```

```
        super().__init__(**kwargs)
        self.use_causal = use_causal
        self.use_rms_norm = use_rms_norm
        self.use_latent_input = use_latent_input
        self.use_conv2d_stem = use_conv2d_stem
        self.latent_dim = latent_dim
        self.downsample_hidden_size = downsample_hidden_size
        self.use_rope = use_rope
        self.rope_parameters = rope_parameters or {}
        self.conv_chunksize = conv_chunksize


class RotaryEmbedding(nn.Module):
    """带缓存的 RoPE 频率表，partial rotary 支持。"""

    def __init__(self, head_dim, rope_parameters, base_seq_len=0):
        super().__init__()
        self.partial_rotary_factor = float(
            rope_parameters.get("partial_rotary_factor", 1.0)
        )
        self.rope_theta = float(rope_parameters.get("rope_theta", 10000.0))
        self.rope_type = rope_parameters.get("rope_type", "default")
        # 保持偶数 rotary 维度，便于 cos/sin 配对
        rotary_dim = int(head_dim * self.partial_rotary_factor)
        self.rotary_dim = (rotary_dim // 2) * 2
        self.attention_scaling = 1.0
        if self.rotary_dim > 0:
            inv_freq = 1.0 / (
                self.rope_theta
                ** (torch.arange(0, self.rotary_dim, 2, dtype=torch.float)
                    / max(self.rotary_dim, 1))
            )
        else:
            inv_freq = torch.tensor([])
        self.register_buffer("inv_freq", inv_freq, persistent=False)
        self._cache = None

    @torch.no_grad()
    def get_cos_sin(self, position_ids, dtype, device):
        """返回 cos/sin 表格；rotary_dim 为 0 时返回 None。"""
        if self.rotary_dim == 0:
            return None, None
        seq_len = position_ids.shape[-1]
        if position_ids.shape[0] == 1 and self._cache is not None:
            cached_seq_len, cached_dtype, cached_device, cached_cos, cached_sin = self._cache
            if cached_seq_len >= seq_len and cached_dtype == dtype and cached_device == device:
                return cached_cos[:seq_len], cached_sin[:seq_len]
        # 无缓存命中时重新计算，并更新缓存
        cos, sin = self._compute_cos_sin(position_ids, dtype, device)
```

```
        self._cache = (seq_len, dtype, device, cos, sin)
        return cos, sin
```

评论区精华

Review 中最有价值的交锋集中在 DP 重叠 MTP 的元数据一致性与共享路径的侵入性上：

- alphabetc1 指出 dots_hybrid_backend 在默认 dsa/dsa 配置下会拿到 DeepseekSparseAttnBackend，而 DotsSWAMLAAttnBackend 期望 FA 风格的 page_size/swa_page_table 元数据，SWA forward 将失败；作者承认该问题并承诺在合并后的 follow-up PR 中单独创建 FA backend。
- alphabetc1 建议 cache_seqlens 重建时用 padding 追加 dummy 行而不是覆盖原值，因为 spec decoding 时两者不一定相等；作者接受并采用该辅助函数。
- yhyang201 报告 DeepSeek V4 架构 is_hybrid_swa=True 但没有 swa_attention_layer_ids 属性，MTP 启动会 AttributeError；作者确认修复。
- yhyang201 分析 draft_forward 多步复用 ForwardBatch 时 mlp_sync_prepared 保持 True 导致 geometry 停留在 step 1，破坏 DP+MTP(steps>1) 的 eager draft；作者在 post_forward_mlp_sync_batch 尾部加 unpad。
- Fridge003 质疑 dsa_indexer.py K-only rotary 的 dummy query 与 key 别名风险、以及 forward_batch_info.py/flashattention_backend.py 的侵入性改动；作者分别用 if 分支保护、并将公共改动收敛到 dots 专属路径。
- yhyang201 对 flashmla_ops 命名提出 rename 建议（swa_mla_fallback），作者未明确回应，属于遗留风格问题。
- DSA 默认 backend 与 DotsSWAMLAAttnBackend 元数据不兼容 (correctness): 作者承认问题，表示合并后发 follow-up PR 创建 SWA 专用的 FA backend，并简化 cookbook 以使用默认 target backend 配置。
- DP 重叠 MTP 下 cache_seqlens 规范化应保留原值 (correctness): 作者接受并采用建议的 padding 辅助函数，保留已有行。
- DeepSeek V4 MTP 启动 AttributeError (correctness): 作者回复 ok，确认修复。
- eager draft 中 mlp_sync_prepared 标志导致多步 DP+MTP 几何错误 (correctness): 作者在 post_forward_mlp_sync_batch 尾部增加 unpad，清除该标志。
- DSA K-only rotary 输入别名风险 (correctness): 已修复并保留分支。
- 公共 FA/ForwardBatch 路径的侵入性修改 (design): 已收敛到 dots 专属路径。
- EAGLE CUDA graph replay 的 bucket-padded cache-location 缓冲区 (correctness): 保留但缩小到必要分支。
- flashmla_ops 命名误导 (style): 无明确后续，属于遗留风格问题。

风险与影响

- 风险：
  1. DSA 默认后端不兼容（高风险）：dots_hybrid_backend.py 中 DotsSWAMLAAttnBackend 依赖 FA 风格元数据，但默认 dsa/dsa 配置下 SWA 层获得

的是 DeepseekSparseAttnBackend，用户按 cookbook 之外的方式启动可能直接运行失败（review 中已确认，合并后修复）。

2. 共享池配置回归：pool_configurator.py 的 SWA 草稿层记账影响所有 hybrid-SWA 模型（如 DeepSeek V4、Inkling），_draft_swa_layers_num 计算偏差会导致 KV 池容量不足或目标池容量浪费。
3. speculative 公共路径回归：model_runner.py、spec_aux_hidden_state.py、eagle_draft_cuda_graph_runner.py 的改动影响所有 EAGLE/NextN 路径，尤其 CUDA graph 重放几何与 DeepSeek V4 MTP 交互。
4. DSA 索引器行为变更：dsa_indexer.py 的 K-only rotary 防别名虽加 if 保护，但仍影响公共 kernel 路径，需关注 CUDA/HIP/XPU 之外的 backend。
5. 视频预处理环境依赖：preprocess.py 使用 torchcodec 内存解码远程视频，新增依赖的同时带来 CPU 峰值、解码器版本差异和音频 token 预算估计偏差的潜在风险。
6. 大规模新增回归面：9600+ 行代码横跨多模块，尽管配套 7+ 个测试文件，但模型整体端到端覆盖仍有限。 - 影响：用户侧：dots.note.omni 用户可从单一 SGLang 服务获得图像、音频、视频及混合输入支持，并可启用 MTP 快速解码与 DP/TP/EP 大规模部署；--language-only 支持编码器 /LLM 拆分部署。系统侧：新增 dots_hybrid_backend、全共享 MTP 草稿模型、视频预处理管线，并修改共享 KV 池配置与 speculative 执行路径；这些改动对其他 hybrid-SWA 模型自动生效，属于隐式行为变更。团队侧：50 个 commits、5 位 reviewer 深度参与的长期功能分支，模型主体 dots3.py 达 2881 行，后续维护需要模块化拆分（作者在 review 中已提及 follow-up 计划）。

- 风险标记：DSA 默认后端下 SWA 运行失败（合并后跟进），共享 attention/speculative 路径改动，DP 重叠 MTP 元数据一致性，新增 torchcodec 视频解码依赖，9600 行大规模新增回归面

关联脉络

- PR #36195 （作者在 review 中提及的 Dots SWA backend 选择修复 PR）：作者在回复 alphabetc1 时主动提及该 follow-up PR，用于修复 DotsSWAMLAAttnBackend 在默认 DSA backend 下元数据不兼容的问题，并简化 cookbook 配置。
- PR #36055 [Diffusion] Load MiniMax H3 GGUF text encoders: 同属多模态模型原生编码器加载方向，且均涉及 text/vision encoder 的 checkpoint 兼容与加载路径扩展，属于同一能力线。
- PR #36044 [Diffusion] Load Comfy NVFP4 MiniMax H3 checkpoints: 同为多模态模型量化 checkpoint 原生加载工作，与 dots.note.omni 的 FP8/ 量化 encoder 路径有共通的技术挑战。