PR #46535 完整报告

vllm-project/vllm

[Model Runner V2][MM] Support EVS

合并时间：2026-06-24 22:18

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/46535

执行摘要

- 一句话：V1 模型状态机增加多模态嵌入剪枝（EVS）支持
- 推荐动作：建议精读本 PR，特别是 MultiModalPruner 的设计——它将模型特定的嵌入后处理抽象为独立组件，避免在主路径中散落条件分支，提升了可扩展性。同时也展示了如何通过基类默认方法减少重复代码（gather_mm_embeddings 上提）。对于多模态推理工程师有参考价值。

功能与动机

支持多模态模型（如 Qwen2.5-VL / Qwen3-VL / Nemotron-Nano-VL）的 Efficient Video Sampling（EVS）功能，这些模型在 media embeddings 后附加 mrope 位置通道，需要在 GPU 上剥离并重新计算位置以进行正确的位置编码，避免将 input_ids 复制到 CPU 造成的额外开销。参考 PR body："This also changes the Qwen VL recompute_mrope_positions methods to accept input_ids as a device tensor rather than python list, to avoid cpu roundtrip."

实现拆解

1. 新增 MultiModalPruner 类（vllm/v1/worker/gpu/model_states/mm_pruning.py）：提供 strip 方法用于 draft forward 时剥离位置通道，recompute 方法用于 target forward 时按请求分割嵌入、调用模型的 recompute_mrope_positions 并将重新计算的位置写回 RopeState。__init__ 记录 inputs_embeds_size 作为裁剪宽度。
2. 修改抽象接口 ModelState（vllm/v1/worker/gpu/model_states/interface.py）：增加 gather_mm_embeddings 默认方法，将 encoder_runner.gather_mm_embeddings 的重复调用统一提升到基类；同时将 get_mm_embeddings 签名新增 req_states 参数以支持 pruner 访问请求状态。在类级别添加 encoder_runner 类型标注。
3. 集成到 DefaultModelState（vllm/v1/worker/gpu/model_states/default.py）：在 __init__ 中通过 maybe_create_mm_pruner 条件创建 pruner；在 get_mm_embeddings 中调用 pruner.recompute 并在之后调用 apply_staged_writes 刷新 staged rope 更新；新增 gather_mm_embeddings 覆盖方法，在其中调用 pruner.strip。
4. 更新EncoderDecoderModelState（vllm/v1/worker/gpu/model_states/encoder_decoder.py）：类似地引入 pruner 的创建和调用。

5. 优化 Qwen VL 模型（vllm/model_executor/models/qwen3_vl.py 和 qwen2_5_vl.py）：将 recompute_mrope_positions 的 input_ids 参数类型从 list[int] 扩展为 list[int] | torch.Tensor，内部根据类型结构进行转换或直接使用设备张量，避免 CPU 回环。同时更新 multimodal_embeddings 类型为 Sequence。 此外，vllm/model_executor/models/interfaces.py 增加了 supports_multimodal_pruning 判断接口，vllm/v1/worker/gpu/model_runner.py 和 vllm/v1/worker/gpu/mm/rope.py 分别调整了调用方式（增加 req_states 参数传递）和提供了 read_prefill_positions / update_prefill_positions 方法。

关键文件：

- vllm/v1/worker/gpu/model_states/mm_pruning.py（模块 剪枝器；类别 source；类型 data-contract；符号 MultiModalPruner, init, strip, recompute）：新增核心 MultiModalPruner 类，封装了嵌入剪枝逻辑
- vllm/v1/worker/gpu/model_states/default.py（模块 状态模型；类别 source；类型 data-contract；符号 gather_mm_embeddings）：DefaultModelState 集成 pruner，重写 get_mm_embeddings 和 gather_mm_embeddings
- vllm/v1/worker/gpu/model_states/interface.py（模块 接口；类别 source；类型 data-contract；符号 gather_mm_embeddings）：抽象接口增加 gather_mm_embeddings 默认方法，更新 get_mm_embeddings 签名
- vllm/v1/worker/gpu/mm/rope.py（模块 位置编码；类别 source；类型 core-logic；符号 read_prefill_positions, update_prefill_positions）：新增 read_prefill_positions 和 update_prefill_positions 方法供 pruner 使用
- vllm/model_executor/models/qwen3_vl.py（模块 模型；类别 source；类型 data-contract）：修改 recompute_mrope_positions 方法以接受设备端张量，避免 CPU 回环
- vllm/model_executor/models/qwen2_5_vl.py（模块 模型；类别 source；类型 data-contract）：同步修改 recompute_mrope_positions 方法以适应接口变化
- vllm/v1/worker/gpu/model_runner.py（模块 模型运行器；类别 source；类型 data-contract）：调整 get_mm_embeddings 调用传递 req_states 参数
- vllm/model_executor/models/diffusion_gemma.py（模块 模型；类别 source；类型 data-contract；符号 get_mm_embeddings）：适配 get_mm_embeddings 新签名并改用 self.gather_mm_embeddings
- vllm/model_executor/models/interfaces.py（模块 接口；类别 source；类型 data-contract）：新增 supports_multimodal_pruning 函数用于检测模型是否支持剪枝
- vllm/v1/worker/gpu/model_states/encoder_decoder.py（模块 状态模型；类别 source；类型 data-contract）：集成 pruner 支持，与 DefaultModelState 模式相同

关键符号：MultiModalPruner.init, MultiModalPruner.strip, MultiModalPruner.recompute, maybe_create_mm_pruner, DefaultModelState.get_mm_embeddings, DefaultModelState.gather_mm_embeddings, ModelState.gather_mm_embeddings, RopeState.read_prefill_positions, RopeState.update_prefill_positions, Qwen3VLChatModel.recompute_mrope_positions,

Qwen2_5VLChatModel.recompute_mrope_positions

关键源码片段

vllm/v1/worker/gpu/model_states/mm_pruning.py

新增核心 MultiModalPruner 类，封装了嵌入剪枝逻辑

```
# SPDX-License-Identifier: Apache-2.0
# SPDX-FileCopyrightText: Copyright contributors to the vLLM project
import torch
import torch.nn as nn

from vllm.config import ModelConfig
from vllm.model_executor.models.interfaces import supports_multimodal_pruning
from vllm.multimodal.utils import get_mm_features_in_window
from vllm.v1.worker.gpu.input_batch import InputBatch
from vllm.v1.worker.gpu.mm.encoder_cache import EncoderCache
from vllm.v1.worker.gpu.mm.rope import RopeState
from vllm.v1.worker.gpu.states import RequestState


class MultiModalPruner:
    """Recomputes M-RoPE positions for multimodal models that prune embeddings
    (e.g. Qwen2.5-VL / Qwen3-VL / Nemotron-Nano-VL Efficient Video Sampling).

    Pruning models append their mrope-position channels to the (variable-count)
    media embeddings from `embed_multimodal`. Those channels must be split off and
    used to recompute mrope positions before the embeddings are merged.
    """

    def __init__(
        self,
        model: nn.Module,
        rope_state: RopeState,
        encoder_cache: EncoderCache,
        inputs_embeds_size: int,
    ) -> None:
        self.model = model
        self.rope_state = rope_state
        self.encoder_cache = encoder_cache
        # 清理后的嵌入宽度：剪枝模型在其尾部附加 mrope 位置通道，
        # 因此 embeds[:, :inputs_embeds_size] 即可剥离它们。
        self.inputs_embeds_size = inputs_embeds_size

    def strip(self, mm_embeds: list[torch.Tensor]) -> list[torch.Tensor]:
        """Draft forward: 仅剥离附加的位置通道。

        剥离是逐个 embedding 进行的，不需要按请求分段。
        推测器重用目标已重新计算的位置，因此这里不需要回写位置。
```

```
    """
    return [mm[:, : self.inputs_embeds_size] for mm in mm_embeds]

def recompute(
    self,
    mm_embeds: list[torch.Tensor],
    input_batch: InputBatch,
    req_states: RequestState,
) -> list[torch.Tensor]:
    """Target forward: 将每个请求的媒体嵌入中附加的 mrope 位置通道分割出来,
    重新计算修正后的 mrope 位置，并将它们存回 RopeState。
    返回清理后的扁平嵌入列表。
    """
    cleaned: list[torch.Tensor] = []
    pos = 0
    req_idx_list = input_batch.idx_mapping_np.tolist()
    prefill_lens_list = input_batch.prefill_len_np.tolist()
    num_computed_list = input_batch.num_computed_prefill_tokens_np.tolist()
    num_scheduled_list = input_batch.num_scheduled_tokens.tolist()
    for batch_idx, req_id in enumerate(input_batch.req_ids):
        num_computed = num_computed_list[batch_idx]
        query_end = num_computed + num_scheduled_list[batch_idx]
        num_req_embeds = self._num_window_embeds(req_id, num_computed, query_end)
        if num_req_embeds == 0:
            continue
        req_embeds = mm_embeds[pos : pos + num_req_embeds]
        pos += num_req_embeds

        req_idx = req_idx_list[batch_idx]
        prefill_len = prefill_lens_list[batch_idx]
        # 从请求状态获取设备端的 input_ids
        input_ids = req_states.all_token_ids.gpu[req_idx, :prefill_len]
        mrope_positions = self.rope_state.read_prefill_positions(
            req_idx, prefill_len
        ).long()
        # 调用模型的 recompute_mrope_positions（已修改为接受设备张量）
        req_cleaned, new_positions, delta = self.model.recompute_mrope_positions(
            input_ids=input_ids,
            multimodal_embeddings=req_embeds,
            mrope_positions=mrope_positions,
            num_computed_tokens=num_computed,
        )
        # 将重新计算的位置写回 RopeState
        self.rope_state.update_prefill_positions(req_idx, new_positions, delta)
        cleaned.extend(req_cleaned)

    assert pos == len(mm_embeds)
    return cleaned
```

vllm/v1/worker/gpu/model_states/default.py

DefaultModelState 集成 pruner，重写 get_mm_embeddings 和 gather_mm_embeddings

```
    def get_mm_embeddings(
        self,
        scheduled_encoder_inputs: dict[str, list[int]],
        input_batch: InputBatch,
        req_states: RequestState,
    ) -> torch.Tensor:
        mm_hashes, mm_kwargs = self.encoder_runner.prepare_mm_inputs(
            scheduled_encoder_inputs
        )
        if mm_kwargs:
            encoder_outputs = self.encoder_runner.execute_mm_encoder(mm_kwargs)
            self.encoder_cache.encoder_outputs.update(zip(mm_hashes, encoder_outputs))

        mm_embeds, is_mm_embed = super().gather_mm_embeddings(input_batch)
        if self.mm_pruner is not None and mm_embeds:
            # EVS: 为剪枝的媒体重新计算 mrope 位置。
            mm_embeds = self.mm_pruner.recompute(mm_embeds, input_batch, req_states)
            # 必须刷新 staged rope 更新，以便 prepare_inputs() 拾取。
            self.apply_staged_writes()

        input_ids_unpadded = input_batch.input_ids[: input_batch.num_tokens]
        inputs_embeds = self.encoder_runner.get_inputs_embeds(
            input_ids_unpadded, mm_embeds, is_mm_embed
        )
        return inputs_embeds[: input_batch.num_tokens_after_padding]

    def gather_mm_embeddings(
        self, input_batch: InputBatch, draft_lookahead: int = 0
    ) -> tuple[list[torch.Tensor], torch.Tensor]:
        mm_embeds, is_mm_embed = super().gather_mm_embeddings(
            input_batch, draft_lookahead
        )
        if self.mm_pruner is not None:
            # EVS: 剥离附加的 mrope 位置通道。
            mm_embeds = self.mm_pruner.strip(mm_embeds)
        return mm_embeds, is_mm_embed
```

评论区精华

本 PR 获得了两名 review 的审阅，WoosukKwon 批准，yewentao256 表示“Thanks for the work!”，未发现实质性技术讨论或争议。

- 暂无高价值评论线程

风险与影响

- 风险：主要风险在于：
  - 核心路径变更：get_mm_embeddings 和 gather_mm_embeddings 是多模态推理的关键步骤，新增 pruner 流程可能影响非剪枝模型的正常运行，需确认 maybe_create_mm_pruner 仅在支持剪枝的模型上返回非空。
  - 缺少测试覆盖：本次变更未包含直接的测试文件，可能遗漏边界情况（如无多模态输入、pruner 为 None 时的回退路径）。
  - 接口契约变更：get_mm_embeddings 签名新增 req_states 参数，所有子类实现（如 DiffusionGemmaModelState、EncoderDecoderModelState）已相应修改，但未来自定义的 ModelState 子类若未同步更新则会出错。
  - 性能风险：recompute 方法中多次调用 tolist() 会将 GPU 张量同步到 CPU，可能引入额外开销，但此路径仅在剪枝模型启用时执行。
  - 影响：用户影响：启用后，Qwen2.5-VL / Qwen3-VL 等模型在 V1 框架下可正确进行 EVS 推理，提升视频处理效率并降低延迟。系统影响：新增模块 MultiModalPruner 和多个接口变更，但通过条件创建确保向后兼容（不启用时行为不变）。团队影响：维护者需要关注新接口对自定义 ModelState 的影响，未来可扩展支持更多剪枝模型。
  - 风险标记：核心路径变更，缺少测试覆盖，接口契约变更，多模态模型影响

关联脉络

- 暂无明显关联 PR