PR #46762 完整报告

vllm-project/vllm

[ModelRunner V2] Support realtime embeddings

合并时间：2026-06-27 10:42

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/46762

执行摘要

- 一句话：支持实时模型解码步嵌入收集
- 推荐动作：值得精读，特别是区分实时与非实时嵌入收集的设计、以及通过 dummy_inputs_embeds 解耦 CUDA 图捕获与编码器执行的方法。对理解 vLLM V2 多模态流程有参考价值。

功能与动机

实时模型（如 Voxtral）在解码步骤也需要多模态嵌入，原有方案只在预填充阶段收集，导致解码时嵌入缺失。PR body 明确说明：'Realtime models like voxtral require embeddings for decode steps too.'

实现拆解

1. 识别实时模型：在 EncoderRunner.__init__ 中通过 supports_realtime(model) 设置 self.is_realtime 标志（encoder_runner.py）。
2. 修改嵌入收集逻辑：在 EncoderRunner.gather_mm_embeddings 中根据 is_realtime 决定是否对所有请求（包括解码）收集嵌入；非实时模型行为不变（encoder_runner.py）。
3. 引入 dummy_inputs_embeds 接口：在 ModelState 抽象基类（interface.py）中添加方法（默认返回 None）；在 DefaultModelState（default.py）中实现为返回 self.encoder_runner.inputs_embeds[:num_tokens]。
4. 调整模型执行路径：在 ModelRunner.execute_model 中，若为 dummy run 则调用 dummy_inputs_embeds 而非 get_mm_embeddings，避免 dummy 阶段运行编码器（model_runner.py）。
5. 测试与类型修正：同步更新 test_encoder_runner.py 中的参数名；修正 diffusion_gemma.py 中 get_mm_embeddings 的返回类型为 torch.Tensor | None。

关键文件：

- vllm/v1/worker/gpu/mm/encoder_runner.py（模块 多模态编码器；类别 source；类型 core-logic；符号 EncoderRunner.init, EncoderRunner.gather_mm_embeddings, EncoderRunner.is_realtime）：核心变更：添加 is_realtime 标志，重构 gather_mm_embeddings 以支持解码步收集
- vllm/v1/worker/gpu/model_runner.py（模块 模型运行器；类别 source；类型 core-logic；符号 ModelRunner.execute_model）：修改 execute_model 中多模态嵌入准备路径，

dummy run 使用 dummy_inputs_embeds

- vllm/v1/worker/gpu/model_states/interface.py（模块 模型状态；类别 source；类型 data-contract；符号 ModelState.dummy_inputs_embeds, ModelState.gather_mm_embeddings）：添加 dummy_inputs_embeds 抽象方法，更新 gather_mm_embeddings 签名
- vllm/v1/worker/gpu/model_states/default.py（模块 模型状态；类别 source；类型 data-contract；符号 DefaultModelState.dummy_inputs_embeds）：实现 dummy_inputs_embeds 返回预分配缓冲区
- vllm/model_executor/models/diffusion_gemma.py（模块 模型实现；类别 source；类型 data-contract；符号 DiffusionGemmaModel.get_mm_embeddings）：修正 get_mm_embeddings 返回类型为 Optional
- tests/v1/worker/test_encoder_runner.py（模块 测试；类别 test；类型 test-coverage）：更新测试参数以匹配新接口

关键符号：EncoderRunner.init, EncoderRunner.gather_mm_embeddings, ModelState.dummy_inputs_embeds, DefaultModelState.dummy_inputs_embeds, ModelRunner.execute_model, DiffusionGemmaModel.get_mm_embeddings

关键源码片段

vllm/v1/worker/gpu/mm/encoder_runner.py

核心变更：添加 is_realtime 标志，重构 gather_mm_embeddings 以支持解码步收集

```
def gather_mm_embeddings(
    self,
    req_ids: list[str],
    total_num_scheduled_tokens: int,
    num_scheduled_tokens: np.ndarray,
    query_start_loc: np.ndarray,
    prefill_lens: np.ndarray,
    num_computed_tokens: np.ndarray,
    draft_lookahead: int = 0,
) -> tuple[list[torch.Tensor], torch.Tensor]:
    if draft_lookahead:
        num_computed_tokens = num_computed_tokens + draft_lookahead

    is_mm_embed = torch.zeros(
        total_num_scheduled_tokens, dtype=torch.bool, device="cpu"
    )

    # 判断是否需要排除嵌入收集（非实时模型在解码步跳过）
    exclude_embeddings: list[bool] | None = None
    if not self.is_realtime:
        # 非实时模型的媒体嵌入只出现在 prompt 中
        is_decode = num_computed_tokens >= prefill_lens
        if is_decode.all():
            # 所有请求都是解码，无需收集嵌入
```

```
            return [], is_mm_embed
        exclude_embeddings = is_decode.tolist()

    query_start = num_computed_tokens.tolist()
    query_end = (num_computed_tokens + num_scheduled_tokens).tolist()

    mm_embeds: list[torch.Tensor] = []
    for i, req_id in enumerate(req_ids):
        if exclude_embeddings is not None and exclude_embeddings[i]:
            # 非实时解码请求，跳过
            continue

        cur_query_start = query_start[i]
        cur_query_end = query_end[i]

        mm_features = self.encoder_cache.mm_features[req_id]
        lo, hi = get_mm_features_in_window(
            mm_features, start=cur_query_start, end=cur_query_end
        )
        for idx in range(lo, hi):
            mm_feature = mm_features[idx]
            pos_info = mm_feature.mm_position
            start_pos = pos_info.offset
            num_encoder_tokens = pos_info.length

            start_idx = max(cur_query_start - start_pos, 0)
            end_idx = min(cur_query_end - start_pos, num_encoder_tokens)
            assert start_idx < end_idx
            curr_embeds_start, curr_embeds_end = (
                pos_info.get_embeds_indices_in_range(start_idx, end_idx)
            )
            if curr_embeds_start == curr_embeds_end:
                continue

            mm_hash = mm_feature.identifier
            encoder_output = self.encoder_cache.encoder_outputs.get(mm_hash, None)
            if encoder_output is None:
                continue # 缓存未命中，通常不应发生
            mm_embeds.append(encoder_output[curr_embeds_start:curr_embeds_end])
            # 标记对应 token 位置为多模态嵌入
            is_mm_embed[cur_query_start + start_idx: cur_query_start + end_idx] = True

    return mm_embeds, is_mm_embed
```

vllm/v1/worker/gpu/model_runner.py

修改 execute_model 中多模态嵌入准备路径，dummy run 使用 dummy_inputs_embeds

```
  if self.supports_mm_inputs and self.is_first_pp_rank:
      # 对于 dummy runs（CUDA 图捕获阶段），直接从预分配缓冲区
```

```
    # 获取嵌入，避免调用实际编码器，同时保持正确的形状以匹配
    # 编译模型。
    if dummy_run:
        inputs_embeds = self.model_state.dummy_inputs_embeds(
            input_batch.num_tokens_after_padding
        )
    else:
        scheduled_encoder_inputs = scheduler_output.scheduled_encoder_inputs
        if self.lora_config is not None:
            set_active_mm_loras(
                model=self.model,
                lora_manager=self.lora_manager,
                encoder_cache=self.encoder_cache,
                req_id_to_index=self.req_states.req_id_to_index,
                lora_state=self.lora_state,
                scheduled_encoder_inputs=scheduled_encoder_inputs,
            )
        inputs_embeds = self.model_state.get_mm_embeddings(
            scheduled_encoder_inputs, input_batch, self.req_states
        )
    if inputs_embeds is not None and not self.model.requires_raw_input_tokens:
        input_ids = None
```

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

添加 dummy_inputs_embeds 抽象方法，更新 gather_mm_embeddings 签名

```
class ModelState(ABC):
    # ...
    def dummy_inputs_embeds(self, num_tokens: int) -> torch.Tensor | None:
        """Pre-allocated inputs_embeds buffer for dummy runs (contents unused)."""
        return None

    def gather_mm_embeddings(
        self, input_batch: InputBatch, draft_lookahead: int = 0
    ) -> tuple[list[torch.Tensor], torch.Tensor]:
        """Gather cached multimodal embeddings."""
        return self.encoder_runner.gather_mm_embeddings(
            input_batch.req_ids,
            input_batch.num_tokens,
            input_batch.num_scheduled_tokens,
            input_batch.query_start_loc_np,
            input_batch.prefill_len_np,
            input_batch.num_computed_tokens_np, # 由 computed_prefill_tokens_np 重命名而来
            draft_lookahead=draft_lookahead,
        )
```

评论区精华

reviewer yewentao256 直接批准（LGTM），无实质讨论。claude[bot] 自动提示该 PR 来自 fork，未启动审查。

- Overall approval (other): No changes requested.

风险与影响

- 风险：
  1. 回归风险：非实时模型的行为通过 is_decode.all() 和 exclude_embeddings 保持原逻辑，但 draft_lookahead 与 num_computed_tokens 的叠加计算需注意边界。
  2. 性能影响：实时模型每个步骤都执行嵌入收集循环与缓存查找，可能增加解码延迟；dummy run 不再调用编码器，略有改善。
  3. 兼容性：参数 computed_prefill_lens 重命名为 num_computed_tokens，内部调用已同步更新，外部直接调用 gather_mm_embeddings 的代码需适配。
  4. 测试覆盖：测试仅覆盖基本场景，缺少实时与非实时混合批处理及 dummy run 状态下实时模型的行为验证。 - 影响：用户影响：Voxtral 等实时模型可正常解码，不再因缺少嵌入报错；非实时模型无感知。系统影响：模型执行路径增加分支判断，开销极小。团队影响：自定义 ModelState 子类需实现 dummy_inputs_embeds 方法（默认返回 None 即可），其他外部接口无需改动。 - 风险标记：核心路径变更，参数重命名，新功能分支

关联脉络

- PR #46753 [ModelRunner V2] Fix cross-attention block table sizing: 同为 ModelRunner V2 的 bugfix，修改了同一层的 model_runner.py 和模型状态相关文件
- PR #46776 [ModelRunner V2] Deduplicate ModelState init logic: 同为 ModelRunner V2 的清理，涉及 model_states 接口和默认实现，与本 PR 有文件重叠