PR #41992 完整报告

vllm-project/vllm

[MM][Perf][CG] Support ViT full CUDA graph for Kimi-VL

合并时间：2026-06-17 20:14

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/41992

执行摘要

- 一句话：为 Kimi-VL 添加 ViT 全 CUDA 图支持
- 推荐动作：值得精读。重点阅读 moonvit.py 中 prepare_encoder_metadata 的设计（如何将 grid 相关计算外提）以及 kimi_vl.py 中协议方法的实现，可作为其他视觉模型启用 ViT CUDA 图的参考蓝本。

功能与动机

Kimi-VL 的 MoonVitPretrainedModel 在前向中包含 .tolist() 调用（位置嵌入插值、RoPE 频率计算、patch merging），与 CUDA 图捕获不兼容。本 PR 通过重构 moonvit.py 添加图安全路径（预计算元数据缓冲区），然后将 kimi_vl.py 接入 SupportsEncoderCudaGraph 协议，旨在消除 ViT 编码器的内核启动开销，提升多模态推理性能。

实现拆解

1. 重构 moonvit.py 编码器层：将 Learnable2DInterpPosEmb.forward 拆分为 get_pos_embeds，接受 grid_hws_list（Python 列表）而非原始 tensor，避免 CUDA 图内部的 .tolist() 循环；在 MoonVisionPatchEmbed 中添加 pos_embeds 参数支持跳过插值；为 Rope2DPosEmb 新增 get_freqs_cis_by_seqlens_list 方法，批量计算 RoPE 频率；新增 patch_merger_packed 和 _build_merge_gather_idx 实现图安全的 patch merging。
2. 预计算元数据入口：在 MoonVitPretrainedModel 中新增 prepare_encoder_metadata，接受 grid 列表，一次性返回所有与 grid 相关的元数据字典（pos_embeds、rope_freqs_cis、cu_seqlens、merge_gather_idx 等），此方法在图外调用。
3. 实现 SupportsEncoderCudaGraph 协议：KimiVLForConditionalGeneration 新增继承 SupportsEncoderCudaGraph，并实现以下方法：
   - get_encoder_cudagraph_config：定义 buffer_keys（pixel_values、pos_embeds、rope_freqs_cis 等）和配置。
   - get_encoder_cudagraph_budget_range：返回 token 预算范围（64 ~ max_num_batched_tokens）。
   - get_encoder_cudagraph_item_specs：根据 grid 返回每个 item 的 input_size 和 output_tokens。
   - select_encoder_cudagraph_items：根据 indices 从 mm_kwargs 中选择对应 buffer。
   - prepare_encoder_cudagraph_capture_inputs 等构建捕获 / 回放输入。

4. 测试覆盖：tests/models/multimodal/generation/test_vit_cudagraph.py 新增 kimi_vl 配置，使用 dummy 权重（dummy_hf_overrides）和单 bucket budget（encoder_cudagraph_token_budgets=[1024]），确保 CI 兼容。
5. 示例与文档：examples/generate/multimodal/vision_language_offline.py 在 MODELS_SUPPORT_VIT_CUDA_GRAPH 中添加 kimi_vl；docs/design/cuda_graphs_multimodal.md 在模型表格中新增 KimiVLForConditionalGeneration 行。

关键文件：

- vllm/model_executor/models/moonvit.py（模块 视觉编码器；类别 source；类型 core-logic；符号 forward, get_pos_embeds, get_freqs_cis_by_seqlens, get_freqs_cis_by_seqlens_list）：核心重构：提取 get_pos_embeds、patch_merger_packed、prepare_encoder_metadata 等图安全方法，消除 .tolist() 依赖。
- vllm/model_executor/models/kimi_vl.py（模块 多模态模型；类别 source；类型 core-logic；符号 KimiVLForConditionalGeneration, get_encoder_cudagraph_config, get_encoder_cudagraph_budget_range, _get_grid_hws）：协议实现：KimiVLForConditionalGeneration 继承 SupportsEncoderCudaGraph 并实现所有必要方法。
- tests/models/multimodal/generation/test_vit_cudagraph.py（模块 CUDA 图测试；类别 test；类型 test-coverage；符号 kimi_vl_chat_template）：测试覆盖：新增 Kimi-VL 配置，使用 dummy 权重和单 budget 验证 CUDA 图捕获与回放。
- examples/generate/multimodal/vision_language_offline.py（模块 离线示例；类别 source；类型 core-logic）：示例支持：将 kimi_vl 加入 MODELS_SUPPORT_VIT_CUDA_GRAPH 列表。
- docs/design/cuda_graphs_multimodal.md（模块 设计文档；类别 docs；类型 documentation）：文档更新：在 CUDA 图支持表格中添加 Kimi-VL 行。

关键符号：Learnable2DInterpPosEmb.get_pos_embeds, MoonVisionPatchEmbed.forward, Rope2DPosEmb.get_freqs_cis_by_seqlens_list, MoonVitPretrainedModel.prepare_encoder_metadata, MoonVitPretrainedModel.patch_merger_packed, MoonVitPretrainedModel._build_merge_gather_idx, KimiVLForConditionalGeneration.get_encoder_cudagraph_config, KimiVLForConditionalGeneration.get_encoder_cudagraph_budget_range, KimiVLForConditionalGeneration.get_encoder_cudagraph_item_specs, KimiVLForConditionalGeneration.select_encoder_cudagraph_items, KimiVLForConditionalGeneration.prepare_encoder_cudagraph_capture_inputs, KimiVLForConditionalGeneration.prepare_encoder_cudagraph_replay_buffers

关键源码片段

vllm/model_executor/models/moonvit.py

核心重构：提取 get_pos_embeds、patch_merger_packed、prepare_encoder_metadata 等图安全方法，消除 .tolist() 依赖。

```
def get_pos_embeds(
    self,
    grid_hws_list: list[list[int]] | list[tuple[int, int]],
) -> torch.Tensor:
    """ 根据 grid 列表生成打包的位置嵌入。
    输出形状 ``(sum(h * w), dim)``，通过对每个 grid 的
    学习权重 ``(height, width, dim)`` 进行插值并拼接。
    此方法在 CUDA 图外部调用，因此 per-grid Python 循环是安全的。
    """
    weight_shape = list(self.weight.shape[:-1]) # [H, W]
    pos_embs: list[torch.Tensor] = []

    for shape in grid_hws_list:
        shape_list = [int(shape[0]), int(shape[1])]
        if shape_list == weight_shape:
            # 无需插值，直接使用预训练权重
            pos_embs.append(self.weight.flatten(end_dim=1))
        else:
            # 对权重进行插值以适应不同大小的 grid
            pos_embs.append(
                F.interpolate(
                    self.weight.permute((2, 0, 1)).unsqueeze(0),
                    size=tuple(shape_list),
                    mode=self.interpolation_mode,
                )
                .squeeze(0)
                .permute((1, 2, 0))
                .flatten(end_dim=1)
            )

    if not pos_embs:
        return self.weight.new_zeros((0, self.weight.shape[-1]))
    return torch.cat(pos_embs)


def forward(self, x: torch.Tensor, grid_hws: torch.Tensor) -> torch.Tensor:
    # forward 现在调用 get_pos_embeds，传入 grid_hws 的 .tolist()
    pos_embs = self.get_pos_embeds(grid_hws.tolist())
    out = x + pos_embs
    return out
```

vllm/model_executor/models/kimi_vl.py

协议实现：KimiVLForConditionalGeneration 继承 SupportsEncoderCudaGraph 并实现所有必要方法。

```
def get_encoder_cudagraph_config(self):
```

```
    from vllm.v1.worker.encoder_cudagraph_defs import EncoderCudaGraphConfig
    return EncoderCudaGraphConfig(
        modalities=["image"], # 当前仅支持图像模态
        buffer_keys=[ # CUDA 图中使用的 buffer 键
            "pixel_values",
            "pos_embeds",
            "rope_freqs_cis",
            "cu_seqlens",
            "max_seqlen",
            "merge_gather_idx",
        ],
        out_hidden_size=self.hidden_size, # 编码器输出隐藏层大小
    )


def get_encoder_cudagraph_item_specs(self, mm_kwargs):
    """ 根据每个 grid 的 (h, w) 计算 input_size 和 output_tokens。 """
    from vllm.v1.worker.encoder_cudagraph_defs import EncoderItemSpec
    kh, kw = self.config.vision_config.merge_kernel_size
    return [
        EncoderItemSpec(
            input_size=h * w, # patch 总数 = h * w
            output_tokens=(h // kh) * (w // kw), # merge 后的 token 数
        )
        for h, w in self._get_grid_hws(mm_kwargs)
    ]
```

评论区精华

- 文档符号一致性：shen-shanshan 指出表格中 `X` 应统一为 `(?)(?)`，作者已修复。
- 测试模型规模：shen-shanshan 担忧 MoE 模型超过 CI GPU 显存，作者参考 #43082 改用 dummy_hf_overrides 和单 budget，保持测试轻量。
- 协议接口更新：shen-shanshan 指出 get_input_modality 已在 #44484 中废除，作者删除该方法。
- 变量命名：shen-shanshan 认为 normalized 参数名易混淆，作者将其重命名为 grid_pairs。
- prepare_encoder_metadata 用途：shen-shanshan 询问是否仅用于 CG 路径，作者确认并更新文档说明。
- 文档符号一致性 (style): 作者已采纳并修复。
- 测试模型规模 (testing): 作者改用 dummy_hf_overrides 和单 budget [1024]，参考 #43082 模式，避免下载完整 checkpoint。
- 协议方法废弃 (design): 作者已删除该方法。
- 变量命名清晰度 (style): 作者将 normalized 重命名为 grid_pairs。
- prepare_encoder_metadata 用途 (question): 作者确认仅用于 CUDA 图路径，并修正文档字符串，表示 eager 路径未改动（类似 InternVL #41759）。

风险与影响

- 风险：
  - 重构 moonvit.py 核心前向路径（如 forward 跳转至 get_pos_embeds），若图安全路径与 eager 路径产生数值差异可能导致回归；但测试验证了 eager 与 CUDA 图输出匹配，且遵循已有模型（Qwen3-VL）模式。
  - 当前仅支持 image 模态，video 模态未处理（modalities=["image"]），若未来 Kimi-VL 支持视频需扩展。
  - 测试使用 dummy 权重，不覆盖真实权重下 RoPE 频率计算等数值路径，存在遗漏精度问题的风险。
  - max_seqlen_override 参数可能被误用，但已在 CG 路径中强制设置。
- 影响：
  - 用户：Kimi-VL 用户可通过 --enable-vit-cuda-graph 获得显著 TTFT 降低（P99 降幅 33.5%），尤其在高并发场景下收益明显。
  - 系统：首次 CUDA 图捕获增加预热开销，但后续推理内核启动开销被消除；需额外显存存储 capture graph buffer（由 budget 控制）。
  - 团队：为后续模型（如 GLM-V、DeepSeek VL）接入 ViT CUDA 图提供可复现的模式（moonvit.py 的 prepare_encoder_metadata 模式）。
  - 风险标记：核心路径重构需验证数值对齐，仅支持 image 模态，测试使用 dummy 权重未覆盖真实路径，CUDA 图预热增加首次推理延迟，依赖 SupportsEncoderCudaGraph 协议接口稳定性

关联脉络

- PR #38175 [RFC]: Support ViT Full CUDA Graph (Tracker): 本 PR 是该 tracker issue 的具体实现项之一，目标为实现 ViT 全 CUDA 图。
- PR #38061 [MM][Perf][CG] Support ViT full CUDA graph for Qwen3-VL: 本 PR 遵循 Qwen3-VL 的 ViT CUDA 图模式，许多设计（如 prepare_encoder_metadata、buffer_keys）与之相似。
- PR #44484 [MM] Remove get_input_modality from SupportsEncoderCudaGraph (cleanup): 该 PR 移除了 get_input_modality 方法，本 PR 作者据此删除了 Kimi-VL 的对应实现。
- PR #42288 [MM][CG] Migrate encoder CUDA graph to single values dict API: 本 PR 的 select_encoder_cudagraph_items 和 buffer 管理参考了该 API 变更。
- PR #43082 [MM][CG] Step3-VL ViT CUDA graph test with dummy weights: 本 PR 的测试模式（dummy_hf_overrides + 单 budget）借鉴了 Step3-VL 的 CI 友好测试配置。