PR #52369 完整报告

vllm-project/vllm

[Perf] Avoid more GPU<->CPU syncs in multimodal encoders

合并时间：2026-08-15 01:40

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/52369

执行摘要

- 一句话：消除多模态编码器 GPU<->CPU 同步点，覆盖 8 个模型
- 推荐动作：值得精读，尤其对从事多模态编码器优化、encoder CUDA Graph 或 MRV2 相关工作的人。值得关注的设计决策：用 keep_on_cpu 在字段契约层区分 host-only 数据；用 host 端形状推导（max(h,w)、split_sizes）消除 D2H 读回；用 async_tensor_h2d 统一 H2D staging 语义。建议合入后关注 keye、glm4_1v 的回归测试结果，并在后续新模型实现中复用这套模式。

功能与动机

PR body 明确说明动机：多模态编码器路径上有大量 blocking 的 H2D 传输和设备读回（如用 CPU 张量索引设备张量、把 host 数据转成 device 张量后又以 Python int 读回），每个点都是一次 GPU<->CPU 同步。head_ref 为 avoid-gpu-syncs-4，说明这是系列化推进的第四步，目标是让编码器路径（尤其是 encoder CUDA Graph 捕获 / 重放场景）不再出现隐式同步。

实现拆解

按以下 4 个模式拆解实现：

1. host 数据 staging 非阻塞化：keye.py 与 paddleocr_vl.py 的 _process_image_input / _process_video_embeds / encode_image 中，将 torch.concat(...).to(device) 改为 non_blocking=True，把 cu_seqlens 从 torch.tensor(...).to(device) 改为 async_tensor_h2d；ernie45_vl.py 的 rot_pos_emb 在 rotary_pos_emb_full[pos_ids] 前先对 pos_ids 做 non_blocking 传输（原来是用 CPU 张量索引设备张量），fwd_placeholder 里两个 numpy 构建的 slice 索引改走 async_tensor_h2d；flash_linear_attention 的 prepare_chunk_indices 对 chunk 索引表加 non_blocking=True。
2. host 端形状推导替代设备读回：keye.py 与 paddleocr_vl.py 的 encoder forward 把 max_grid_size = pids.max() + 1 改为 max(max(h, w) for ...)（grid 在 host 端已知），并把输出切分从循环读 cu_seqlens[i] 改为用 image_grid_thw 直接计算 split_sizes 后 split；glm4_1v.py 的 Embeddings.forward 用 np.asarray + np.repeat 在 host 端展开 target_h/target_w，替代逐图 image_shapes[i, 1].repeat(lengths[i]) 的循环与 cat。
3. 消除多余张量 / 标量构造：diffusion_gemma.py 的 repeat_interleave 传入 output_size=int(valid_canvas_len_np.sum()) 避免算子回读 repeats，add_request 里

prompt_len 赋值改用 fill_()；flash_linear_attention 的 prepare_chunk_offsets 用 new_zeros(1) 替代 new_tensor([0])，避免 host list 再次进入张量构造路径。

4. 字段级 keep_on_cpu 契约：ultravox.py 的 audio_lens、interns1.py 的 image_num_patches / video_num_patches、phi4mm.py 的 image_sizes 均改为 MultiModalFieldConfig.batched(..., keep_on_cpu=True)，因为它们在模型侧只用于 Python 级 shape 计算，不需要进设备。
5. 测试配套：本次没有新增或修改任何测试文件，等价性依赖现有模型回归测试（keye、glm4_1v、ernie45_vl 等模型测试）。

关键文件：

- vllm/model_executor/models/keye.py（模块 视觉编码器；类别 source；类型 core-logic；符号 KeyeSiglipEncoder.forward, KeyeSiglipVisionTransformer.forward, _process_image_input, _process_video_embeds）：本 PR 最核心的改动文件：image 与 video 双路径的 position ids / cu_seqlens / sample indices 全部改为 non-blocking 或 async_tensor_h2d，旋转表大小从 pids.max() 改为 host 端 max(h, w)，输出切分改为 host 端 split_sizes，是三类优化模式的集中体现。
- vllm/model_executor/models/glm4_1v.py（模块 视觉编码器；类别 source；类型 core-logic；符号 Embeddings.forward, pos_embeds_interpolate）：改动最大（+18/-37）：Embeddings.forward 中原先对 lengths/image_shapes 做设备化转换后再逐图 repeat 的循环，被 numpy host 端展开 + 单次 async_tensor_h2d 替代，既是性能优化也是数据契约调整（image_shapes 不再需要进设备）。
- vllm/model_executor/models/paddleocr_vl.py（模块 视觉编码器；类别 source；类型 core-logic；符号 SiglipEncoder.forward, encode_image）：携带 keye 视觉编码器的拷贝，应用了完全相同的三类优化（non-blocking staging、host 端 max_grid_size、split_sizes），是模式复用的代表性文件。
- vllm/model_executor/models/ernie45_vl.py（模块 视觉编码器；类别 source；类型 core-logic；符号 rot_pos_emb, prepare_encoder_metadata, fwd_placeholder）：修复了 rot_pos_emb 中用 CPU 张量索引设备张量的隐式同步，cu_seqlens 与 resampler 的两个 numpy slice 索引改走 async_tensor_h2d。
- vllm/third_party/flash_linear_attention/ops/index.py（模块 线性注意力；类别 infra；类型 infrastructure；符号 prepare_chunk_indices, prepare_chunk_offsets）：第三方算子适配层的同步点清理：chunk 索引表加 non_blocking，new_zeros(1) 替代 new_tensor([0]) 避免 host list staging。
- vllm/model_executor/models/diffusion_gemma.py（模块 扩散模型；类别 source；类型 core-logic；符号 add_request, call）：repeat_interleave 显式传 output_size 避免读回 repeats，prompt_len 赋值改用 fill_()，是消除标量 / 张量隐式同步的典型示范。
- vllm/model_executor/models/interns1.py（模块 多模态输入；类别 source；类型 configuration；符号 _get_mm_fields_config）：image/video num_patches 标记 keep_on_cpu=True，与 internvl 保持一致，体现字段契约层的 host-only 分流。
- vllm/model_executor/models/phi4mm.py（模块 多模态输入；类别 source；类型 configuration；符号 _get_mm_fields_config）：image_sizes 标记 keep_on_cpu=True，该字段在模型侧只驱动 Python 级 shape 计算。

- vllm/model_executor/models/ultravox.py（模块 音频编码器；类别 source；类型 configuration；符号 _get_mm_fields_config）：audio_lens 标记 keep_on_cpu=True，它是音频编码器 attention metadata 的 host 端输入。

关键符号：KeyeSiglipEncoder.forward, KeyeSiglipVisionTransformer.forward, KeyeVisionModel._process_image_input, KeyeVisionModel._process_video_embeds, Glm4PositionalEncoding.forward, Glm4VisionTransformer.pos_embeds_interpolate, SiglipEncoder.forward, PaddleOCRVLModel.encode_image, Ernie45SiglipEncoder.rot_pos_emb, Ernie45SiglipEncoder.prepare_encoder_metadata, Ernie45TemporalFusion.fwd_placeholder, DiffusionScheduler.add_request, DiffusionSampler.call, prepare_chunk_indices, prepare_chunk_offsets, Interns1Model._get_mm_fields_config, Phi4MMModel._get_mm_fields_config, UltravoxProcessor._get_mm_fields_config

关键源码片段

vllm/model_executor/models/keye.py

本 PR 最核心的改动文件：image 与 video 双路径的 position ids / cu_seqlens / sample indices 全部改为 non-blocking 或 async_tensor_h2d，旋转表大小从 pids.max() 改为 host 端 max(h, w)，输出切分改为 host 端 split_sizes，是三类优化模式的集中体现。

```
# KeyeSiglipEncoder.forward 局部：旋转位置编码表的大小改为 host 端推导。
# pids 全部由 image_grid_thw 的 h/w 生成，max(h, w) 就是其严格上界，
# 因此不需要再从设备张量 pids.max() 读回一个值（一次 D2H 同步）。
pids = torch.stack([height_position_ids, width_position_ids], dim=-1)
max_grid_size = max(max(h, w) for _, h, w in flatten_image_grid_thw)
rope_emb_max_grid = self.rotary_pos_emb(max_grid_size)
rope_emb = rope_emb_max_grid[pids].flatten(1)
rope_emb = rope_emb.repeat(1, 2)
rope_emb = (rope_emb.cos(), rope_emb.sin())


# KeyeSiglipVisionTransformer.forward 局部：切分输出时不再逐个读设备
# 张量 cu_seqlens[i]（start/end 是 D2H 读回），而是用 host 端
# image_grid_thw 直接算 split_sizes。二者本质是同一份信息（cu_seqlens
# 就是逐图 t*h*w 的累加和），所以这个替代在语义上等价。
last_hidden_state = self.post_layernorm(last_hidden_state)
if cu_seqlens is None:
    raise ValueError('cu_seqlens cannot be None for '
                     'SiglipVisionTransformer output processing.')
split_sizes = [
    int(np.prod(thw)) for thw in self.encoder.flatten_list(image_grid_thw)
]
return [
    tensor.squeeze(0) for tensor in last_hidden_state.split(split_sizes, dim=1)
]

# _process_image_input 局部：host 端拼好的 metadata 统一走
```

```
# pinned + non-blocking staging，替代默认的阻塞 H2D 拷贝。
siglip_position_ids = torch.concat(siglip_position_ids, dim=0).to(
    pixel_values.device, non_blocking=True
)
cu_seqlens = async_tensor_h2d(
    cu_seqlens, dtype=torch.int32, device=pixel_values.device
)
sample_indices = torch.concat(sample_indices, dim=0).to(
    pixel_values.device, non_blocking=True
)
```

vllm/model_executor/models/glm4_1v.py

改动最大（+18/-37）：Embeddings.forward 中原先对 lengths/image_shapes 做设备化转换后再逐图 repeat 的循环，被 numpy host 端展开 + 单次 async_tensor_h2d 替代，既是性能优化也是数据契约调整（image_shapes 不再需要进设备）。

```
# Embeddings.forward 局部：lengths 与 image_shapes 都是 host 数据
# （调用方已不再把它们转成 device 张量），所以在 host 上直接用 numpy
# 展开，再一次性 async_tensor_h2d 跨过去；旧实现是逐图
# image_shapes[i, 1].repeat(lengths[i]) 再 cat，既慢还会在
# lengths[i] 被用于 Python 循环时强制读回。
# 数据并行下部分 GPU 可能拿不到全部 image shapes，按下标取模循环。
shapes_np = np.asarray(image_shapes)
shape_idx = np.arange(len(lengths)) % shapes_np.shape[0]
target_h = async_tensor_h2d(
    np.repeat(shapes_np[shape_idx, 1], lengths),
    device=device,
    dtype=torch.float32,
)
target_w = async_tensor_h2d(
    np.repeat(shapes_np[shape_idx, 2], lengths),
    device=device,
    dtype=torch.float32,
)

# 归一化坐标：target 已以 non-blocking 方式到达设备侧，
# async_tensor_h2d 内部会保证数据可用性后再进行后续计算。
norm_w = ((w_coords + 0.5) / target_w) * 2 - 1
norm_h = ((h_coords + 0.5) / target_h) * 2 - 1
```

vllm/third_party/flash_linear_attention/ops/index.py

第三方算子适配层的同步点清理：chunk 索引表加 non_blocking，new_zeros(1) 替代 new_tensor([0]) 避免 host list staging。

```
# ops/index.py 局部：chunk 索引表在 host 端构建完成后，用
# non_blocking 传回设备，避免此处变成 H2D 同步点。
chunk_indices = torch.stack([indices.eq(0).cumsum(0) - 1, indices], 1)
return chunk_indices.to(
    device=cu_seqlens.device, dtype=cu_seqlens.dtype, non_blocking=True
```

```
)


# 相邻函数：用 new_zeros(1) 替代 new_tensor([0])，避免再次把 host
# list 引入张量构造路径。
return torch.cat(
    [cu_seqlens.new_zeros(1),
     triton.cdiv(prepare_lens(cu_seqlens), chunk_size)]
).cumsum(-1)
```

评论区精华

PR 没有实质性的技术讨论：WoosukKwon 直接 approve 且未留下评论；claude[bot] 因 PR 来自 fork 而跳过自动 review（提示可通过 @claude review 触发）。2 条 issue 评论仅为 CI 触发与 Buildkite 链接。整体属于无人提出异议的机械性等价重构，维护者以 approve 表达对等价性判断的认可。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 无直接测试覆盖：9 个文件中 8 个是模型源码路径，无对应测试变更，行为等价性依赖现有模型回归测试，存在回归漏检风险。
  2. glm4_1v 的 numpy 重构：np.asarray(image_shapes) 要求 image_shapes 是规则形状的二维数据（如 [[t, h, w]]），旧逻辑对 device 张量 /CPU list 都兼容；若调用方传入形状异常（如空列表），shape_idx 取模与 np.repeat 的语义需与旧循环逐项核对，data parallel 下的 cycling 逻辑也要保持等价。
  3. keye/paddleocr_vl 的 split_sizes 假设：新逻辑假设 image_grid_thw 与设备端 cu_seqlens 严格对应（split_sizes 之和等于序列长度），一旦 encoder metadata 出现 padding 或 batch 采样不一致，split 会静默切错；现有代码路径上 cu_seqlens 由同一 host list 构建，风险较低但值得留意。
  4. non_blocking 的 pinned 依赖：non_blocking=True 在源数据位于 pinned memory 时才真正异步，async_tensor_h2d 内部应处理该语义；若传入普通 CPU 张量 / 列表，仍可能退化为同步传输（安全但收益打折）。
  5. diffusion_gemma 的 output_size：output_size=int(valid_canvas_len_np.sum()) 必须与 repeat 展开长度一致，否则 repeat_interleave 会报错或产生错误结果；当前两者同源于 valid_canvas_len_np，一致性有保证。 - 影响：用户 / 性能：多模态请求（图片、视频、音频）的编码阶段每次迭代减少若干次 GPU<->CPU 同步，对编码器延迟与吞吐有正向影响，尤其在 encoder CUDA Graph 捕获与重放场景下收益更明显；无 API 变化、无用户可见行为变化。系统：降低编码器在 eager 与 graph 路径上的同步点密度，减小 host 端 shape 计算与设备端张量传输的耦合。团队：确立了后续多模态模型编写时的模式共识——只用于 host 计算的字段标记 keep_on_cpu=True、host 形状在 host 推导、跨设备统一走 async_tensor_h2d。影响面广但风险集中在行为等价性，属于低风险高收益的机械优化。 - 风险标记：无直接测试覆盖，跨 8 个模型等价性风险，依赖

Host/Device 数据契约，核心编码路径变更

关联脉络

- PR #49852 [MRV2][Multimodal] Enable encoder cuda graph for model runner v2: 同一作者 njhill 的多模态编码器性能主线：为 v1 编码器启用 CUDA Graph 捕获与重放；本 PR 正是该方向下一步——消除编码器路径上的 GPU<->CPU 同步，避免破坏图捕获或拖慢重放。