

# PR #47217 完整报告

vllm-project/vllm

[Bugfix][Gemma4] Keep image bidirectional attention within the sliding window

合并时间: 2026-07-03 10:57

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/47217>

## 执行摘要

- 一句话: 修复 Gemma4 视觉模型双向注意力超滑动窗口后降级为因果注意力
- 推荐动作: 值得精读。该 PR 展示了如何通过模型标志与 kernel 标志组合实现后端无关的安全修复, 设计干净、风险隔离。尤其适合学习 CuTE mask\_mod 在 FlashAttention 中的运用及死代码消除技巧。

## 功能与动机

Gemma4 vision models that use blockwise bidirectional image attention (12B, 26B-A4B, 31B) silently fell back to causal-only attention over image tokens whenever a single image's soft-token span exceeded the text sliding window (e.g.  $\sim 1100$  soft tokens vs `sliding_window=1024`). This degraded vision/OCR quality — catastrophic on the encoder-free 12B and latent on tower-based versions. The issue is backend-agnostic and stems from GPUModelRunner dropping the entire bidirectional mm\_prefix range for an image when `range_len > sliding_window`.

## 实现拆解

1. 模型标志定义: 在 `Gemma4ForConditionalGeneration` (`gemma4_mm.py`) 添加类属性 `mm_prefix_clamp_sliding_window: bool = True`; 在 `Gemma4Attention` (`gemma4.py`) 构造函数中传入 `mm_prefix_clamp_sliding_window=self.is_sliding` 使得仅滑动层启用。`Attention` 基类 (`attention.py`) 存储该标志供后端查询。
2. Runner 层逻辑调整: 在 `GPUModelRunner._build` (`gpu_model_runner.py`) 中读取 `self.model.mm_prefix_clamp_sliding_window`, 当为真时不丢弃超长双向范围, 保留给后端处理。其他模型默认仍丢弃。
3. Triton 后端钳制: 在 `TritonAttentionImpl` (`triton_attn.py`) 中将标志传递给 kernel; 在 `compute_kv_seq_mask` (`triton_attention_helpers.py`) 中若 `MM_PREFIX_CLAMP_SW` 且 `SLIDING_WINDOW > 0`, 对双向范围附加  $(query\_abs\_pos - seq\_offset) < SLIDING\_WINDOW$  条件。
4. FA4 后端钳制: 在 `FlashAttentionImpl` (`flash_attn.py`) 的 `forward` 中计算 `mm_clamp_sw = sliding_window_size[0] + 1` 传入 `_make_mm_prefix_mask_mod`; 函数内部若 `sliding_window > 0` 对每个双向框附加  $(q\_idx - kv\_idx) < sliding\_window$  条件, 默认 0 时死代码消除。

5. 测试验证：虽无自动化测试，但在 A100 (Triton) 和 H100 (FA4) 上通过了端到端 OCR 对比和 logits 数值对比，与 HuggingFace 参考实现一致。

关键文件：

- vllm/v1/attention/backends/flash\_attn.py (模块 注意力后端；类别 source；类型 core-logic；符号 \_make\_mm\_prefix\_mask\_mod)：核心改动：修改 \_make\_mm\_prefix\_mask\_mod 支持滑动窗口钳制，并在 forward 中读取模型标志，实现 FA4 后端的双向范围裁剪。
- vllm/v1/worker/gpu\_model\_runner.py (模块 运行器；类别 source；类型 data-contract)：Runner 逻辑：根据模型标志决定是否跳过超窗口双向范围，连接模型与后端的纽带。
- vllm/model\_executor/models/gemma4\_mm.py (模块 模型定义；类别 source；类型 data-contract)：模型标志入口：新增类属性 mm\_prefix\_clamp\_sliding\_window，开启内核滑动窗口钳制。
- vllm/model\_executor/models/gemma4.py (模块 模型层；类别 source；类型 data-contract)：Gemma4Attention 构造函数中传递 mm\_prefix\_clamp\_sliding\_window=self.is\_sliding，使仅滑动层启用钳制。
- vllm/model\_executor/layers/attention/attention.py (模块 注意力基类；类别 source；类型 data-contract)：Attention 基类新增 mm\_prefix\_clamp\_sliding\_window 存储标志。
- vllm/v1/attention/backends/triton\_attn.py (模块 注意力后端；类别 source；类型 core-logic)：Triton 后端：将层标志传递给 kernel helper，启用 Triton 端的钳制。

关键符号：\_make\_mm\_prefix\_mask\_mod, compute\_kv\_seq\_mask

## 关键源码片段

### vllm/v1/attention/backends/flash\_attn.py

核心改动：修改 \_make\_mm\_prefix\_mask\_mod 支持滑动窗口钳制，并在 forward 中读取模型标志，实现 FA4 后端的双向范围裁剪。

```
def _make_mm_prefix_mask_mod(max_ranges: int, sliding_window: int = 0):  
    """Build a CuTE-DSL mask_mod implementing (causal OR mm_prefix).
```

```
    Returns a @cute.jit callable that evaluates:
```

```
    keep = (kv_idx <= q_idx) OR
```

```
           (q_idx in [r_start, r_end] AND kv_idx in [r_start, r_end]
```

```
           [AND (q_idx - kv_idx) < sliding_window])
```

```
    for each mm_prefix range stored in aux_tensors[0].
```

```
    """
```

```
    import cutlass
```

```
    import cutlass.cute as cute
```

```
    from cutlass import Int32
```

```
    from vllm.vllm_flash_attn.cute.utils import scalar_to_ssa
```

```
    @cute.jit
```

```
    def mm_prefix_mask_mod(  
        batch_idx: cute.TensorSSA,
```

```
        batch_idx: cute.TensorSSA,
```

```

head_idx: cute.TensorSSA,
q_idx: cute.TensorSSA,
kv_idx: cute.TensorSSA,
seqlen_info,
aux_tensors,
):
    keep = kv_idx <= q_idx
    ranges = aux_tensors[0]
    for i in range(max_ranges):
        r_start = scalar_to_ssa(Int32, ranges[:, :, 0, i])
        r_end = scalar_to_ssa(Int32, ranges[:, :, 1, i])
        valid = r_start < r_end
        q_in = (q_idx >= r_start) & (q_idx <= r_end) & valid
        k_in = (kv_idx >= r_start) & (kv_idx <= r_end) & valid
        mm = q_in & k_in
        if sliding_window > 0:
            # Gemma4: clamp bidirectional block to sliding window,
            # matching HF's (causal OR blockwise) AND sliding_window.
            # Future keys (kv > q) satisfy q - kv < 0 < window.
            mm = mm & ((q_idx - kv_idx) < sliding_window)
        keep = keep | mm
    return keep
mm_prefix_mask_mod.use_fast_sampling = True
return mm_prefix_mask_mod

```

## vllm/v1/worker/gpu\_model\_runner.py

Runner 逻辑：根据模型标志决定是否跳过超窗口双向范围，连接模型与后端的纽带。

```

# Compute mm_prefix bidirectional ranges before building attention metadata.
# By default, ranges exceeding sliding_window are skipped. Models that
# clamp mm_prefix to the sliding window in-kernel (e.g., Gemma4) opt out
# of the skip so the bidirectional range survives for large images.
_clamps_in_kernel = getattr(
    self.model, "mm_prefix_clamp_sliding_window", False
)
for req_id in self.input_batch.req_ids:
    image_doc_ranges = []
    req_state = self.requests[req_id]
    for mm_feature in req_state.mm_features:
        if mm_feature.modality == "audio":
            continue
        pos_info = mm_feature.mm_position
        img_doc_range = pos_info.extract_embeds_range()
        for r in img_doc_range:
            if (
                not _clamps_in_kernel
                and _bidi_sw is not None
                and (r[1] - r[0] + 1) > _bidi_sw
            ):

```

```
        continue # Drop wide range (default behavior)
    image_doc_ranges.append(r)
    req_doc_ranges[req_idx] = image_doc_ranges
```

## vllm/model\_executor/models/gemma4\_mm.py

模型标志入口：新增类属性 mm\_prefix\_clamp\_sliding\_window，开启内核滑动窗口钳制。

```
class Gemma4ForConditionalGeneration(
    nn.Module,
    SupportsMultiModal,
    SupportsQuant,
    SupportsPP,
    SupportsLoRA,
    SupportsEagle3,
):
    # Gemma4 clamps mm_prefix bidirectional ranges to the sliding window
    # in-kernel (HF's (causal OR blockwise) AND sliding_window).
    # The model runner reads this to keep image bidirectional ranges that
    # exceed the window instead of dropping them (which would make image
    # attention causal-only for images larger than the sliding window).
    mm_prefix_clamp_sliding_window: bool = True

    packed_modules_mapping = { ... }
    hf_to_vllm_mapper = WeightsMapper(...)

    def __init__(self, *, vllm_config: VllmConfig, prefix: str = ""):
        super().__init__()
        config = vllm_config.model_config.hf_config
        ...
```

## 评论区精华

Reviewer Isotr0py 建议在滑动层直接跳过 mm\_prefix 掩码计算，认为滑动窗口 + 因果掩码已足够。作者澄清：双向注意力需要允许未来 key ( $k_v > q$ )，这被因果掩码排除，所以必须显式包含；因此采用 clamp 而非跳过。FA4 后端中 mask\_mod 是唯一元素掩码，没有它会回退到纯因果 + 窗口。最终采纳 clamp 方案，讨论 resolved。

- 滑动层上 mm\_prefix 掩码是否可跳过 (design): 采用 clamp 方案：对双向区域附加  $(q\_idx - kv\_idx) < sliding\_window$  条件，而不是跳过。

## 风险与影响

- 风险：
  - 缺少自动化测试：三个后端的改动若回归可能不被 CI 捕获，但手动测试覆盖了主要场景。
  - FA4 后端在长输入上可能有残留精度问题（cjackal 观察到 gibberish，作者怀疑是 soft token 预算不足，未完全确认），H100 用户需额外验证。
  - 改动限于 Gemma4（通过死代码消除保护其他模型），风险隔离。
  - constexpr 和参数默认值确保无性能开销。

- 影响：
  - 用户：Gemma4 视觉模型 OCR/ 视觉问答正确性恢复；需使用较大 `--max-soft-tokens`（如 1120）以体验修复；H100 用户需确认 FA4 端到端质量。
  - 系统：无显著性能影响（标志默认 `false` 且死代码消除）。
  - 团队：合并后建议添加集成测试和数值回归测试。
  - 风险标记：缺少自动化测试，FA4 后端精度待确认

## 关联脉络

- 暂无明显关联 PR