

PR #53147 完整报告

vllm-project/vllm

[Kernel][Gemma4] Prune Triton sliding-window tiles for multimodal prefixes

合并时间: 2026-09-01 06:21

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/53147>

执行摘要

- 一句话: 多模态前缀恢复 SWA 剪枝, Gemma4 prefill 提速 4 倍
- 推荐动作: 值得精读。核心看点: ① 如何把 SWA 剪枝与 MM 前缀掩码的 OR 语义正确合并为并集而不是交集, 尤其是 mgoin 指出的非因果右窗口与 chunk 边界两个易错点; ② 评审驱动测试升级路径——从 bounds 几何测试到真实 kernel 的 dense parity 测试, 后者才是保证剪枝不丢 key 的关键; ③ 默认关闭新行为 (MM_PREFIX_CLAMP_SW=False) 的低风险落地方式。对从事注意力 kernel / Triton 优化, 或需要支持 window-clamped 前缀掩码的模型开发者有直接参考价值。

功能与动机

PR body 明确这是一次性能回归修复: Triton 本应剪掉 sliding window 之外的 KV tile, 但一旦存在多模态前缀 range, `compute_tile_loop_bounds` 就会整体禁用剪枝, 原代码中留有 `TODO(Isotr0py): sliding window pruning with image bidirectional mask`。对 gemma-4-26B-A4B-it-NVFP4 的 60 图 / 请求、平均 16,550 token 前缀负载, 回归导致 server prefill 高达 19.90 s、TTFT 21.99 s; 恢复剪枝后分别降至 4.80 s 与 5.67 s, 对应 4.15x 与 3.23x 的提升。

实现拆解

1. 剪枝条件与参数扩展 (vllm/v1/attention/ops/triton_attention_helpers.py) :
`compute_tile_loop_bounds` 尾部新增 5 个参数 `USE_R_SWA`、`MM_PREFIX_CLAMP_SW`、`MAX_MM_RANGES`、`mm_prefix_range_ptr`、`seq_idx`。剪枝条件由 `SLIDING_WINDOW > 0 and not USE_MM_PREFIX` 改为基于 `can_prune_sliding = (not USE_R_SWA) and ((not USE_MM_PREFIX) or MM_PREFIX_CLAMP_SW)`, 即 R-SWA 反向窗口不按正向 SWA 剪枝、普通 MM 前缀仍不剪枝, 只有 Gemma4 这类 window-clamped 前缀掩码才恢复剪枝, 保证其它模型行为零变化。
2. clamped MM 前缀的并集边界 (同文件): 先按因果 / 非因果分别计算基础窗口的 `last_allowed_key` (非因果右窗口扩展 `context_len + qpos_hi + SLIDING_WINDOW - 1` 被保留), 再遍历至多 `MAX_MM_RANGES` 个图像 range, 对与当前 Q 块相交的 range: 上界用 `tl.maximum` 并入 `range_end`, 下界用 `tl.minimum` 放宽到 `max(range_start, query_abs_lo - SLIDING_WINDOW + 1)`, 最后统一 clamp 到 `seq_len - 1` 并换算为 tile 区间。关键语义是 MM 掩码以 OR 方式叠加在基础掩码上, tile 范围必须取并集, 任何一侧少算都会剪掉实际参与注意力的 key。

3. Triton kernel 调用点透传 (vllm/v1/attention/ops/triton_unified_attention.py) :
kernel_unified_attention 中原来的 USE_MM_PREFIX or USE_R_SWA 实参拆回 USE_MM_PREFIX, 并在 CHUNK_SIZE 之后按新签名透传 USE_R_SWA、MM_PREFIX_CLAMP_SW、MAX_MM_RANGES、mm_prefix_range_ptr、seq_idx, 使 kernel 内 bounds 计算与 compute_kv_seq_mask 掩码语义保持一致。
4. 测试配套 (tests/kernels/attention/test_triton_unified_attention.py, +253 行) : 新增两个直接调用 compute_tile_loop_bounds 的 Triton helper 几何测试——test_clamped_mm_prefix_prunes_sliding_window_tiles (4096 token 序列 + 1024 窗口 + 超窗图像 range, 断言保守区间 [3, 72) 而非完整 [0, 128)) 与 test_clamped_mm_prefix_preserves_noncausal_right_window (非因果右窗口与图像 range 取并集 [3, 68)) ; 并按评审要求新增 ref_paged_clamped_mm_attn 稠密参考实现与 test_triton_unified_attn_clamped_mm_matches_dense_reference, 直接调用 unified_attention 输出与参考对比 (atol/rtol 2e-2) , 且断言去掉 MM range 后结果不同, 证明图像前缀确实参与注意力。

关键文件:

- vllm/v1/attention/ops/triton_attention_helpers.py (模块 注意力内核; 类别 source; 类型 core-logic; 符号 compute_tile_loop_bounds) : 核心实现: compute_tile_loop_bounds 新增 5 个参数并在 SWA 剪枝分支实现 clamped MM 前缀的并集边界计算, 是本次性能修复的实质所在。
- tests/kernels/attention/test_triton_unified_attention.py (模块 内核测试; 类别 test; 类型 test-coverage; 符号 _compute_clamped_mm_tile_bounds, test_clamped_mm_prefix_prunes_sliding_window_tiles, test_clamped_mm_prefix_preserves_noncausal_right_window, ref_paged_clamped_mm_attn) : 新增 253 行测试: 两个 tile 边界几何测试 + 一个稠密参考 parity 测试, 其中 ref_paged_clamped_mm_attn 是验证并集语义的独立参考实现, 直接回应评审对真实输出对比的要求。
- vllm/v1/attention/ops/triton_unified_attention.py (模块 注意力内核; 类别 source; 类型 core-logic; 符号 kernel_unified_attention) : Triton kernel 调用点: 把 USE_MM_PREFIX or USE_R_SWA 拆开并按新签名透传 5 个参数, 是 bounds 计算落到实际 kernel 的关键接线。

关键符号: compute_tile_loop_bounds, kernel_unified_attention, _compute_clamped_mm_tile_bounds, ref_paged_clamped_mm_attn, test_clamped_mm_prefix_prunes_sliding_window_tiles, test_clamped_mm_prefix_preserves_noncausal_right_window, test_triton_unified_attn_clamped_mm_matches_dense_reference

关键源码片段

vllm/v1/attention/ops/triton_attention_helpers.py

核心实现: `compute_tile_loop_bounds` 新增 5 个参数并在 SWA 剪枝分支实现 clamped MM 前缀的并集边界计算, 是本次性能修复的实质所在。

```

# compute_tile_loop_bounds 内部，SWA 剪枝分支的核心并集逻辑
# can_prune_sliding: R-SWA 反向窗口不能按正向 SWA 剪枝；普通 MM prefix
# 需要完整序列，但 Gemma4 的 MM_PREFIX_CLAMP_SW 把前缀注意力夹到滑动窗口
# 左缘，因此可以恢复剪枝。
can_prune_sliding = (not USE_R_SWA) and ((not USE_MM_PREFIX) or MM_PREFIX_CLAMP_SW)
if SLIDING_WINDOW > 0 and can_prune_sliding:
    # qpos_lo / qpos_hi 为该 Q 块覆盖的全局 query 行区间
    # （测试几何为 [1120, 1127]），q_abs 取该块最后一行
    q_abs = context_len + qpos_hi
    if CHUNK_LOOKBACK >= 0:
        # chunked attention: 下界由 chunk 回溯决定
        first_allowed_key = ((q_abs // CHUNK_SIZE) - CHUNK_LOOKBACK) * CHUNK_SIZE
    else:
        first_allowed_key = q_abs - SLIDING_WINDOW + 1

# 非因果 / 逐序列因果场景下窗口向右（未来 key）扩展
if USE_PER_SEQ_CAUSAL or (not USE_CAUSAL):
    last_allowed_key = context_len + qpos_hi + SLIDING_WINDOW - 1
else:
    last_allowed_key = context_len + qpos_hi

# Gemma4 的 clamped MM prefix: MM 掩码以 OR 方式叠加在基础掩码上，
# 因此 tile 范围必须是基础窗口与每个相交图像 range 的并集。
if USE_MM_PREFIX and MM_PREFIX_CLAMP_SW:
    query_abs_lo = context_len + qpos_lo
    query_abs_hi = context_len + qpos_hi
    for i in range(MAX_MM_RANGES):
        # mm_prefix_range_ptr 布局: [seq][range_idx][start, end]
        range_start = tl.load(
            mm_prefix_range_ptr + seq_idx * MAX_MM_RANGES * 2 + i * 2
        )
        range_end = tl.load(
            mm_prefix_range_ptr + seq_idx * MAX_MM_RANGES * 2 + i * 2 + 1
        )
        intersects_query_block = (
            (range_start < range_end)
            & (range_start <= query_abs_hi)
            & (range_end >= query_abs_lo)
        )
        # MM 掩码可重新启用 chunk 边界左侧的 key，故下界取更早者
        mm_first_allowed_key = tl.maximum(
            range_start, query_abs_lo - SLIDING_WINDOW + 1
        )
        first_allowed_key = tl.minimum(
            first_allowed_key,
            tl.where(intersects_query_block, mm_first_allowed_key, first_allowed_key),
        )
    # 上界并入相交 range 的终点（图像内容可一直读到 range_end）
    last_allowed_key = tl.maximum(

```

```

        last_allowed_key,
        tl.where(intersects_query_block, range_end, last_allowed_key),
    )
    last_allowed_key = tl.minimum(last_allowed_key, seq_len - 1)

```

换算为 tile 索引并 clamp 到有效范围

```
tile_start = tl.maximum(0, first_allowed_key // TILE_SIZE)
```

```
tile_end = tl.minimum((last_allowed_key // TILE_SIZE) + 1, num_tiles)
```

tests/kernels/attention/test_triton_unified_attention.py

新增 253 行测试：两个 tile 边界几何测试 + 一个稠密参考 parity 测试，其中 `ref_paged_clamped_mm_attn` 是验证并集语义的独立参考实现，直接回应评审对真实输出对比的要求。

```

@triton.jit
def _compute_clamped_mm_tile_bounds(
    output_ptr,
    mm_prefix_range_ptr,
    USE_CAUSAL: tl.constexpr,
    USE_PER_SEQ_CAUSAL: tl.constexpr,
    CHUNK_LOOKBACK: tl.constexpr,
    CHUNK_SIZE: tl.constexpr,
):
    # 测试专用 helper: 直接调用生产路径 compute_tile_loop_bounds,
    # 把 loop_lo / loop_hi / max_seq_prefix_len 写回 host 侧断言几何。
    loop_lo, loop_hi, max_seq_prefix_len = compute_tile_loop_bounds(
        0, # context_len
        4096, # seq_len: 4K token 序列
        4096, # cur_batch_query_len
        140, # q_block_local_idx: query 行 [1120, 1127]
        0, # segm_idx_or_0
        0, # tiles_per_segment_or_0
        32, # TILE_SIZE
        16, # BLOCK_M
        8, # BLOCK_Q
        2, # num_queries_per_kv
        1024, # SLIDING_WINDOW
        True, # USE_MM_PREFIX
        False, # IS_3D
        USE_CAUSAL,
        USE_PER_SEQ_CAUSAL,
        CHUNK_LOOKBACK,
        CHUNK_SIZE,
        False, # USE_R_SWA
        True, # MM_PREFIX_CLAMP_SW
        2, # MAX_MM_RANGES
        mm_prefix_range_ptr,
        0, # seq_idx
    )

```

```
tl.store(output_ptr, loop_lo)
tl.store(output_ptr + 1, loop_hi)
tl.store(output_ptr + 2, max_seq_prefix_len)
```

```
def test_clamped_mm_prefix_prunes_sliding_window_tiles() -> None:
    # 图像 range [1024, 2303] 宽于滑动窗口 1024, 且与 query 块相交:
    # 上界必须取 range 的含端点 2303 (tile 72)。若剪枝失效,
    # 区间会退化为完整扫描 [0, 128)。
    mm_prefix_ranges = torch.tensor(
        [[[1024, 2303], [0, 0]]], dtype=torch.int32, device=DEVICE_TYPE
    )
    bounds = torch.empty(3, dtype=torch.int32, device=DEVICE_TYPE)

    _compute_clamped_mm_tile_bounds([(1,)](
        bounds,
        mm_prefix_ranges,
        USE_CAUSAL=True,
        USE_PER_SEQ_CAUSAL=False,
        CHUNK_LOOKBACK=-1,
        CHUNK_SIZE=-1,
    ))

    # 104 = 1127 - 1024 + 1, 首 tile = 104 // 32 = 3;
    # 2303 落在 tile 71 内, 故区间为 [3, 72)
    assert bounds.tolist() == [3, 72, 4096]
```

评论区精华

评审 (mgoin) 围绕三处正确性问题展开, 作者逐一修复并补充测试:

“Could we preserve the non-causal/per-sequence-causal right-window bound here? This branch bypasses the following `USE_PER_SEQ_CAUSAL` or (not `USE_CAUSAL`) case, although `compute_kv_seq_mask` can still admit keys through `q_hi + SLIDING_WINDOW - 1`.”

首版 `clamped` 分支直接用 `query_abs_hi` 作为上界, 会丢掉非因果 / 逐序列因果的右窗口 key。作者以 commit [b7e34ef](#) 保留非因果右窗口分支, 最终实现先按因果 / 非因果分别算 `last_allowed_key`, 再与相交 `range` 终点取 `maximum` 并集。

“This lower bound is unsafe if clamped MM-prefix and chunked attention are combined. Chunking restricts the base mask, but the MM mask is OR-ed afterward and is clamped by the ordinary sliding window, so it can re-enable keys before this chunk boundary.”

下界若只用 `chunk` 回溯公式, 会把被 MM 掩码重新启用的更早 key 剪掉。作者以 commit [d14ed1db](#) 把相交 `range` 的 `mm_first_allowed_key` 与 `chunk` 下界取 `minimum`, 放宽下界。

“Please add at least one real `unified_attention` output-parity test against a dense clamped-MM reference. This direct helper test confirms one geometry, but it cannot

catch disagreement between loop bounds and `compute_kv_seq_mask`, or errors in pointer/sequence wiring.”

纯 bounds 几何测试无法发现 bounds 与掩码不一致或指针接线错误。作者以 commit `b84b1cff` 新增 `ref_paged_clamped_mm_attn` 与真实 kernel 输出 parity 测试，并随后 (`a3a47d9`) 删除冗余的 chunk-boundary tile-bounds 测试。

- 非因果 / 逐序列因果的右窗口边界被 clamped 分支绕过 (correctness): 作者以 commit `b7e34ef` 保留非因果右窗口分支，最终实现先按因果 / 非因果分别计算 `last_allowed_key`，再与相交 MM range 终点取 maximum 合并，并新增对应几何测试。
- clamped MM 与 chunked attention 组合时下界不安全 (correctness): 作者以 commit `d14ed1db` 将相交 MM range 的 `mm_first_allowed_key` (`max(range_start, query_abs_lo - SW + 1)`) 与 chunk 下界取 minimum，放宽下界以保留被重新启用的 key。
- 需要真实 unified_attention 输出 parity 测试 (testing): 作者以 commit `b84b1cff` 新增 `ref_paged_clamped_mm_attn` 稠密参考与 `test_triton_unified_attn_clamped_mm_matches_dense_reference`，以 $2e-2$ tolerance 对比真实 kernel 输出，并断言移除 MM range 后结果不同。

风险与影响

- 风险:
 - 静默数值错误风险: tile bounds 必须与 `compute_kv_seq_mask` 的掩码语义精确一致。当前实现取并集属于保守方向 (不丢 key)，但若未来掩码加严 (如图像内部因果限制) 而 bounds 未同步收紧，会出现多算而非错算；反过来若并集少算一侧就会静默剪掉有效 key。这是注意力 kernel 最危险的失败模式。
 - 共享 helper 签名扩展: `compute_tile_loop_bounds` 是 Triton 注意力路径的共享函数，新增 5 个位置参数插入在 `CHUNK_SIZE` 之后，任何其它调用点若未同步更新会参数错位。本 PR 只改了 `kernel_unified_attention` 一处调用，需人工确认仓库内无第三处调用。
 - 组合路径验证不足: 非因果 + clamped MM 只有几何测试，没有真实模型端到端验证 (Gemma4 恒为因果)；`CHUNK_LOOKBACK >= 0` 与 `MM_PREFIX_CLAMP_SW` 组合也只修复了下界，未做数值级验证 (作者删除了独立的 chunk-boundary bounds 测试)。
 - 默认路径行为保持: `MM_PREFIX_CLAMP_SW` 默认 False，非 Gemma4 多模态模型维持原“不剪枝”行为，无性能回归，但收益面也仅限 Gemma 系列。
- 影响:
 - 用户影响: 对 Gemma4 多模态长提示、高并发图片请求场景收益显著——RTX PRO 6000 上 E2E 从 31.24 s 降至 9.67 s、TTFT 从 21.99 s 降至 5.67 s、decode 吞吐从 58.58 tok/s 提升到 97.24 tok/s。
 - 系统影响: 改动集中在 Triton unified attention 路径的 helper 与 kernel 调用点，默认分支不变，其它模型与后端零影响，风险隔离良好。
 - 团队影响: 为 window-clamped multimodal prefix 注意力建立了可复用机制，`ref_paged_clamped_mm_attn` 可作为后续 Gemma 系列及类似掩码模型的测试基线。
 - 风险标记: 共享 helper 签名扩展需同步所有调用点，bounds 与掩码不一致会导致静默数值错误，非因果与 chunk 组合路径缺少端到端验证，默认路径行为保持，收益面仅限

关联脉络

- PR #53382 [Perf][Kernel] Tune cooperative topk for medium batch-sizes: 同为 kernel 层性能调优 (top-k 协作剪枝 vs SWA tile 剪枝)，都在减少注意力路径的冗余计算，可视为同一性能优化方向的不同分支。
- PR #53677 [kernel] Fused embedding kernel: 同为 kernel 性能专题，且同样围绕长序列 / 高吞吐场景优化，展示了近期 vLLM 对 kernel 层性能的持续投入。
- PR #53821 [Bugfix][ROCm] Preserve AITER unified-attention metadata during graph replay: 同属 unified attention 路径的修复，虽然一个在 Triton 实现、一个在 AITER 实现，但共享注意力边界语义，可对照理解两个后端的掩码与 bounds 处理差异。