PR #48519 完整报告

vllm-project/vllm

[ROCm][Perf] Optimize sparse attention prefill kernel for DeepSeek-V4

合并时间：2026-07-16 08:58

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/48519

执行摘要

- 一句话：优化 DSv4 稀疏注意力 prefill 内核约 2 倍
- 推荐动作：建议合并。该 PR 以极小的代码改动（+10/-3）实现了显著的性能提升，且经过充分验证（gsm8k 精度达标，E2E 基准测试全面改善）。值得关注的设计决策：手动管理 num_warps 以平衡并行度与同步开销，以及通过移除冗余 mask 来优化软件流水线。

功能与动机

DeepSeek-V4 的稀疏注意力 ragged prefill 内核在 ROCm 上存在三处可优化的低效点：num_warps 设置过高导致过多的跨线程束同步和 LDS 读写屏障；冗余 mask 操作插入在 load 与 MFMA 之间，阻碍软件流水线；缺少预取机制未充分利用计算与访存重叠。通过这三项优化，可显著降低 TTFT，提升端到端推理性能。

实现拆解

该 PR 仅修改了一个文件 vllm/v1/attention/ops/rocm_aiter_mla_sparse.py，包含三个优化点：

1. 降低 num_warps：在 _rocm_sparse_attn_prefill_ragged_triton 函数中将 num_warps 从 8 改为 4，减少跨线程束归约开销和 LDS 读写屏障，适配 DSv4-Pro TP8 下 tile 尺寸 [16,512] 缩减至 [16,16] 的场景。
2. 移除冗余 mask：在 _sparse_attn_prefill_ragged_kernel 内核中，删除加载 KV 后立即执行的 tl.where(valid[:, None] & dim_mask[None, :], kv, 0.0) 冗余 mask。由于加载时已设置无效区域为 0.0，后续 mask 多余，且其插入在 load 与 MFMA 之间妨碍软件流水线。
3. 预取下一 tile：在 for 循环末尾提前加载下一个 tile 的 slot 索引（next_k_pos），使得 KV 加载能与当前 tile 的 MFMA 计算重叠，进一步提升性能。

关键文件：

- vllm/v1/attention/ops/rocm_aiter_mla_sparse.py（模块 Attention；类别 source；类型 performance；符号 _sparse_attn_prefill_ragged_kernel, _rocm_sparse_attn_prefill_ragged_triton）：唯一修改文件，包含内核优化核心变更：降低 num_warps、移除冗余 mask、预取下一 tile。

关键符号：_sparse_attn_prefill_ragged_kernel, _rocm_sparse_attn_prefill_ragged_triton

关键源码片段

vllm/v1/attention/ops/rocm_aiter_mla_sparse.py

唯一修改文件，包含内核优化核心变更：降低 num_warps、移除冗余 mask、预取下一 tile。

```
# 文件：vllm/v1/attention/ops/rocm_aiter_mla_sparse.py
# 变更摘要：三项优化提升 DSv4 稀疏 attention prefill 内核性能约 2x

# 优化 1: 降低 num_warps 减少同步开销（在启动配置函数中）
def _rocm_sparse_attn_prefill_ragged_triton(...):
    block_h = 16
    block_d = triton.next_power_of_2(head_dim)
    block_k = 16 if head_dim >= 256 else 32
    num_warps = 4 # 原为 8；DSv4-Pro TP8 下 tile 缩小后，过高 warps 带来多余同步
    out = torch.empty_like(q, dtype=torch.bfloat16)
    _sparse_attn_prefill_ragged_kernel[(num_queries, triton.cdiv(num_heads, block_h))](
        q, ..., num_warps=num_warps, # 传递新值
        BLOCK_H=block_h, BLOCK_D=block_d, BLOCK_K=block_k,
    )
    return out

# 优化 2: 移除冗余 mask + 优化 3: 预取下一 tile slot
# 在内核函数的 for k_start 循环中
@triton.jit
def _sparse_attn_prefill_ragged_kernel(...):
    # 循环开始前预取首个 slot
    slot = tl.load(kv_indices_ptr + kv_start + k_offsets, mask=k_offsets < kv_len, other=-1)
    for k_start in tl.range(0, kv_len, BLOCK_K):
        k_pos = k_start + k_offsets
        in_range = k_pos < kv_len
        # slot 已从循环外预取，不再在此处加载
        valid = in_range & (slot >= 0) & (slot < num_kv)
        safe_slot = tl.where(valid, slot, 0)
        kv = tl.load(..., mask=valid[:, None] & dim_mask[None, :], other=0.0)
        # 移除以下冗余 mask（原地修改无用，且插入在 load-MFMA 之间妨碍流水线）
        # kv = tl.where(valid[:, None] & dim_mask[None, :], kv, 0.0)

        # 预取下一 tile 的 slot，与后续 MFMA 计算重叠
        next_k_pos = k_start + BLOCK_K + k_offsets
        slot = tl.load(
            kv_indices_ptr + kv_start + next_k_pos,
            mask=next_k_pos < kv_len, other=-1
        )
        # 后续 MFMA 计算
        scores = tl.dot(q, tl.trans(kv)) * scale
        scores = tl.where(head_mask[:, None] & valid[None, :], scores, neg_large)
        ...
```

评论区精华

没有 review 评论或争议。审核人 tjtanaa 直接批准（LGTM）。

- 暂无高价值评论线程

风险与影响

- 风险：风险较低。变更仅涉及一个内核函数，且为纯性能优化，逻辑等价性通过正确性测试（gsm8k 精度持平）。但由于修改了 kernel 的 mask 逻辑和预取顺序，如果存在非常规的 sparse 索引（例如超出范围的 slot），可能需要额外验证。目前测试覆盖了 DSv4-Pro 典型形状，但未覆盖所有可能的序列长度与头维度组合。
- 影响：对 DeepSeek-V4 在 ROCm 平台上的 prefill 性能提升显著：内核提速约 2 倍，E2E TTFT 降低 6.7%-10.5%，TPOT/ITL 也有 1-6% 的改善。影响范围限于 ROCm 上的稀疏注意力 prefill 路径，不影响其他 GPU 平台或 decode 路径。用户无需修改任何配置即可受益。
- 风险标记：缺少边界测试覆盖，仅单文件变更

关联脉络

- PR #47718 [ROCm][Perf] DSv4 two-stage compressor kernel for HCA prefill: 同为 ROCm 平台 DeepSeek-V4 prefill 性能优化，涉及内核优化和 LDS 使用，共享性能优化目标。
- PR #47677 [XPU] Add DSpark speculative decoding support for DeepSeek-V4: 同样针对 DeepSeek-V4 的推理优化，但关注投机解码而非稀疏注意力。
- PR #47881 [Feature] Migrate moe sp support to non-torch compiled path for GLM5.2: 同为 DeepSeek 系列模型优化，但针对不同模型和架构。