PR #45725 完整报告

vllm-project/vllm

[ROCm][Perf] mxfp8 moe/linear gfx950 tuning for MiniMax-M3

合并时间：2026-06-17 02:50

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/45725

执行摘要

- 一句话：MiniMax-M3 MXFP8 内核 tile 配置调优
- 推荐动作：值得精读，尤其关注 tile 选择函数的设计模式：在 compute kernel 中引入 regime-based tile 选择是一个通用的优化技巧。该 PR 代码简洁，思路清晰，适合作为 kernel 调优的参考。

功能与动机

原生 MXFP8 dot_scaled MoE/ 线性内核使用单一硬编码 tile 配置（block_m=64, BLOCK_N=128, num_warps=8），长序列预填时 tile 尺寸不足，导致计算效率低下。本 PR 旨在通过 regime-based tile 选择来优化预填和解码场景的性能，使两种 workload 都能获得适合的 tile 配置。

实现拆解

文件 vllm/model_executor/layers/fused_moe/experts/mxfp8_native_moe.py 中，_grouped_gemm_mxfp8 函数新增参数 block_n、num_warps、num_stages（默认值分别为 128、8、2），替代原先的硬编码 BLOCK_N=128 和 num_warps=8，并将 num_stages 作为显式内核参数传递。这使得调用方可根据实际 workload 灵活指定 tile 配置。

在同一文件中，新增模块级常量和辅助函数 _mxfp8_moe_tiles(num_tokens: int) -> dict。当 token 数量 >= 1024 时，返回预填 tile（block_m=128, block_n=256, num_warps=8, num_stages=2）；否则返回解码 tile（block_m=64, block_n=64, num_warps=4, num_stages=2）。该函数在 fused_moe_mxfp8_native 入口处根据实际 token 数 T 调用，获取 tile 配置后传入 _grouped_gemm_mxfp8 两次调用（GEMM1 和 GEMM2）。

文件 vllm/model_executor/kernels/linear/mxfp8/rocm_native.py 中，_mxfp8_dot_scaled_linear 函数原本也是硬编码 tile。修改为根据 M（token 数）>= 1024 选择预填 tile（BLOCK_M=128, BLOCK_N=256, num_warps=8, num_stages=2）或解码 tile（BLOCK_M=64, BLOCK_N=64, num_warps=4, num_stages=2）。

根据 reviewer 建议，简化了代码内联注释，仅说明针对 gfx950 和 MiniMax-M3 形状调优，删除了性能数字描述。

关键文件：

- vllm/model_executor/layers/fused_moe/experts/mxfp8_native_moe.py（模块 MoE 内核；类别 source；类型 core-logic；符号 _mxfp8_moe_tiles, _MXFP8_PREFILL_TILES, _MXFP8_DECODE_TILES, _MXFP8_PREFILL_MIN_TOKENS）：核心 MoE 内核调优，新增 tile 选择函数和参数传递机制，+30/-5。
- vllm/model_executor/kernels/linear/mxfp8/rocm_native.py（模块 线性层内核；类别 source；类型 core-logic）：线性层内核 tile 调优，+9/-2，实现类似的分段选择逻辑。

关键符号：_grouped_gemm_mxfp8, _mxfp8_moe_tiles, fused_moe_mxfp8_native, _mxfp8_dot_scaled_linear

关键源码片段

vllm/model_executor/layers/fused_moe/experts/mxfp8_native_moe.py

核心 MoE 内核调优，新增 tile 选择函数和参数传递机制，+30/-5。

```
def _grouped_gemm_mxfp8(
    a_q: torch.Tensor, # [M, K] fp8 e4m3
    a_scale: torch.Tensor, # [M, K//32] uint8 (E8M0)
    w: torch.Tensor, # [E, N, K] fp8 e4m3
    w_scale: torch.Tensor, # [E, N, K//32] uint8 (E8M0)
    sorted_token_ids: torch.Tensor,
    expert_ids: torch.Tensor,
    num_tokens_post_padded: torch.Tensor,
    num_valid_tokens: int,
    top_k: int,
    block_m: int,
    out_dtype: torch.dtype,
    a_div: int,
    mul_weight_by: torch.Tensor | None = None,
    expert_map: torch.Tensor | None = None,
    # New parameters for configurable tile sizes
    block_n: int = 128,
    num_warps: int = 8,
    num_stages: int = 2,
) -> torch.Tensor:
    # ... (existing code)
    grid = (triton.cdiv(sorted_token_ids.shape[0], block_m), triton.cdiv(N, block_n))
    _mxfp8_grouped_gemm_kernel[grid](
        ...,
        BLOCK_M=block_m,
        BLOCK_N=block_n, # was hardcoded 128
        BLOCK_K=BLOCK_K,
        num_warps=num_warps, # was hardcoded 8
        num_stages=num_stages, # new
    )
    return out

# Tuned native-MXFP8 launch tiles for gfx950 (CDNA4) at MiniMax-M3 MoE shapes.
```

```
_MXFP8_PREFILL_TILES = dict(block_m=128, block_n=256, num_warps=8, num_stages=2)
_MXFP8_DECODE_TILES = dict(block_m=64, block_n=64, num_warps=4, num_stages=2)
_MXFP8_PREFILL_MIN_TOKENS = 1024

def _mxfp8_moe_tiles(num_tokens: int) -> dict:
    """Pick grouped-GEMM launch tiles by regime (token count)."""
    if num_tokens >= _MXFP8_PREFILL_MIN_TOKENS:
        return _MXFP8_PREFILL_TILES
    return _MXFP8_DECODE_TILES


def fused_moe_mxfp8_native(...) -> torch.Tensor:
    T, H = hidden_states.shape
    tiles = _mxfp8_moe_tiles(T) # call tile selector
    block_m = tiles["block_m"]
    # ... then pass tiles to _grouped_gemm_mxfp8 for both GEMM1 and GEMM2
```

vllm/model_executor/kernels/linear/mxfp8/rocm_native.py

线性层内核 tile 调优，+9/-2，实现类似的分段选择逻辑。

```
def _mxfp8_dot_scaled_linear(
    x: torch.Tensor, # [M, K] bf16/fp16
    w: torch.Tensor, # [N, K] fp8 e4m3
    w_scale: torch.Tensor, # [N, K//32] uint8 (E8M0)
) -> torch.Tensor:
    M, K = x.shape
    N = w.shape[0]
    x_q, x_scale = mxfp8_e4m3_quantize(x)
    out = torch.empty((M, N), dtype=x.dtype, device=x.device)
    # Regime-gated launch tiles for gfx950, tuned at MiniMax-M3 shapes
    if M >= 1024:
        BLOCK_M, BLOCK_N, num_warps, num_stages = 128, 256, 8, 2
    else:
        BLOCK_M, BLOCK_N, num_warps, num_stages = 64, 64, 4, 2
    BLOCK_K = 128
    grid = (triton.cdiv(M, BLOCK_M), triton.cdiv(N, BLOCK_N))
    _mxfp8_linear_kernel[grid](
        ...,
        BLOCK_M=BLOCK_M,
        BLOCK_N=BLOCK_N,
        BLOCK_K=BLOCK_K,
        num_warps=num_warps,
        num_stages=num_stages,
    )
    return out
```

评论区精华

Reviewer @tjtanaa 提出了几点意见：

- 简化注释：不要在内联注释中描述性能提升，仅说明为何种模型和场景调优即可。
- 避免使用 **gemm_kw 解包：建议显式传递 block_n、num_warps、num_stages 参数，保持风格一致。 作者采纳了显式传参方式（commit 8dd39aab 中已体现）。
- 注释精简 (style): 采纳，作者删除了注释中的性能细节。
- 参数传递方式 (design): 采纳，作者在后续提交中改为显式传参。

风险与影响

- 风险：风险较低。修改集中在 tile 配置选择逻辑，未触及内核计算核心。若 tile 选择不当，可能导致性能退化而非正确性问题。阈值 1024 是基于特定模型（MiniMax-M3）的形状调优所得，对其他模型可能非最优，但不会导致错误。线性层和 MoE 层的阈值一致，保证了行为协调。
- 影响：仅影响 AMD ROCm gfx950 平台上的原生 MXFP8 内核路径。其他平台（NVIDIA、Intel）不受影响。对 MiniMax-M3 模型，预填和解码性能均有 7-9% 提升。不影响数值精度（gsm8k 准确率验证无退化）。
- 风险标记：仅影响 ROCm gfx950 平台，阈值依据特定模型调优，无测试覆盖

关联脉络

- PR #45896 [feature] MiniMax-M3-MXFP4 support added: 同属 MiniMax-M3 量化优化系列，该 PR 为 MXFP4 支持，本 PR 为 MXFP8 tile 调优，共享部分代码上下文。
- PR #45794 [Bugfix] MiniMax-M3 (AMD): add packed_modules_mapping and pass swiglu…: 同一模型（MiniMax-M3）的 AMD 平台修复，间接关联。