PR #45854 完整报告

vllm-project/vllm

[ROCm][Quant][Perf] Minimax-M3: Enable fp8_per_channel for bf16 weights on mi300x

合并时间：2026-06-17 20:02

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/45854

执行摘要

- 一句话：MiniMax-M3 在 MI300X 上启用 fp8_per_channel 量化，提升吞吐 28%
- 推荐动作：建议合入。该 PR 解决了 MiniMax-M3 在 ROCm 上的性能瓶颈，改动量小且经过充分验证。后续应关注长上下文和更多任务的精度表现。设计上通过参数传递链的轻微调整即可修复精度 bug 并启用新量化，值得参考。

功能与动机

MiniMax-M3 的 bf16 模型在 MI300X 上推理时权重占用大（99.9 GiB/GPU），KV cache 容量受限（1.55M tokens），导致 decode 阶段 HBM 带宽瓶颈。启用 fp8_per_channel 量化可将权重压缩 49%，KV cache 容量提升 75%，从而支持更高并发和更长上下文。PR body 中明确指出性能目标和 gsm8k 精度验证结果。

实现拆解

1. 配置链新增 alpha/beta 参数：在 vllm/model_executor/layers/fused_moe/config.py 的 fp8_w8a8_moe_quant_config 函数签名中增加 gemm1_alpha 和 gemm1_beta 两个 float | None 参数，并传递给 FusedMoEQuantConfig.make。
2. 转发层参数到配置：在 vllm/model_executor/layers/fused_moe/oracle/fp8.py 的 make_fp8_moe_quant_config 函数中，将 gemm1_alpha 和 gemm1_beta 传递到 fp8_w8a8_moe_quant_config 调用。
3. 读取模型层属性：在 vllm/model_executor/layers/quantization/fp8.py 的 Fp8MoEMethod.get_fused_moe_quant_config 和 vllm/model_executor/layers/quantization/online/fp8.py 的 _Fp8OnlineMoEBase.get_fused_moe_quant_config 中，通过 getattr(layer, "swiglu_alpha", None) 和 getattr(layer, "swiglu_beta", None) 读取模型层中的 SwiGLU 参数，并传入配置构造。
4. ROCm 白名单添加：在 vllm/platforms/rocm.py 的 RocmPlatform 类 supported_quantization 列表中添加 "fp8_per_channel"，以允许该量化方法在 ROCm 上使用。
5. 测试与验证：PR body 提供了详细的性能基准和 gsm8k 精度测试结果（精度损失仅 0.0031，约 0.3σ），验证了正确性和性能收益。

关键文件：

- vllm/model_executor/layers/fused_moe/config.py（模块 MoE 配置；类别 source；类型 data-contract；符号 fp8_w8a8_moe_quant_config）：核心配置函数 fp8_w8a8_moe_quant_config 新增 gemm1_alpha/gemm1_beta 参数，是数据契约变更的入口。
- vllm/model_executor/layers/fused_moe/oracle/fp8.py（模块 MoE 配置；类别 source；类型 data-contract；符号 make_fp8_moe_quant_config）：make_fp8_moe_quant_config 转发 gemm1_alpha/beta 到 fp8_w8a8_moe_quant_config。
- vllm/model_executor/layers/quantization/fp8.py（模块 量化方法；类别 source；类型 data-contract；符号 Fp8MoEMethod.get_fused_moe_quant_config）：Fp8MoEMethod.get_fused_moe_quant_config 从 layer 读取 swiglu_alpha/beta，传入配置。
- vllm/model_executor/layers/quantization/online/fp8.py（模块 量化方法；类别 source；类型 data-contract；符号 Fp8PerTensorOnlineMoEMethod.get_fused_moe_quant_config）：_Fp8OnlineMoEBase.get_fused_moe_quant_config 同样从 layer 读取 swiglu_alpha/beta。
- vllm/platforms/rocm.py（模块 平台配置；类别 source；类型 core-logic；符号 RocmPlatform.supported_quantization）：将 fp8_per_channel 添加到 ROCm 支持的量化白名单，是启用量化的关键。

关键符号：fp8_w8a8_moe_quant_config, make_fp8_moe_quant_config, Fp8MoEMethod.get_fused_moe_quant_config, Fp8PerTensorOnlineMoEMethod.get_fused_moe_quant_config

关键源码片段

vllm/model_executor/layers/fused_moe/config.py

核心配置函数 fp8_w8a8_moe_quant_config 新增 gemm1_alpha/gemm1_beta 参数，是数据契约变更的入口。

```
# vllm/model_executor/layers/fused_moe/config.py (head)

def fp8_w8a8_moe_quant_config(
    w1_scale: torch.Tensor,
    w2_scale: torch.Tensor,
    a1_scale: torch.Tensor | None = None,
    a2_scale: torch.Tensor | None = None,
    w1_bias: torch.Tensor | None = None,
    w2_bias: torch.Tensor | None = None,
    per_act_token_quant: bool = False,
    per_out_ch_quant: bool = False,
    block_shape: list[int] | None = None,
    a1_gscale: torch.Tensor | None = None,
    a2_gscale: torch.Tensor | None = None,
    g1_alphas: torch.Tensor | None = None,
    g2_alphas: torch.Tensor | None = None,
    gemm1_alpha: float | None = None, # <-- 新增参数，用于 SwiGLU-OAI alpha（如 MiniMax-M3
```

```
    的 1.702)
    gemm1_beta: float | None = None, # <-- 新增参数，用于 SwiGLU-OAI beta（如 MiniMax-M3 的
    1.0)
    gemm1_clamp_limit: float | None = None,
) -> FusedMoEQuantConfig:
    """
    Construct a quant config for fp8 activations and fp8 weights.
    """
    return FusedMoEQuantConfig.make(
        current_platform.fp8_dtype(),
        w1_scale=w1_scale,
        g1_alphas=g1_alphas,
        w2_scale=w2_scale,
        g2_alphas=g2_alphas,
        w1_bias=w1_bias,
        w2_bias=w2_bias,
        a1_scale=a1_scale,
        a1_gscale=a1_gscale,
        a2_scale=a2_scale,
        a2_gscale=a2_gscale,
        per_act_token_quant=per_act_token_quant,
        per_out_ch_quant=per_out_ch_quant,
        block_shape=block_shape,
        gemm1_alpha=gemm1_alpha, # <-- 传递到 FusedMoEQuantConfig
        gemm1_beta=gemm1_beta, # <-- 传递到 FusedMoEQuantConfig
        gemm1_clamp_limit=gemm1_clamp_limit,
    )
```

vllm/model_executor/layers/fused_moe/oracle/fp8.py

make_fp8_moe_quant_config 转发 gemm1_alpha/beta 到 fp8_w8a8_moe_quant_config。

```
# vllm/model_executor/layers/fused_moe/oracle/fp8.py (head)
    # All other backends use normal config.
    return fp8_w8a8_moe_quant_config(
        w1_scale=w1_scale,
        w2_scale=w2_scale,
        w1_bias=w1_bias,
        w2_bias=w2_bias,
        a1_scale=a1_scale,
        a2_scale=a2_scale,
        block_shape=block_shape,
        per_act_token_quant=per_act_token_quant,
        per_out_ch_quant=per_out_ch_quant,
        gemm1_alpha=gemm1_alpha, # <-- 新增转发
        gemm1_beta=gemm1_beta, # <-- 新增转发
        gemm1_clamp_limit=swiglu_limit,
    )
```

评论区精华

审核人 tjtanaa 提出了 3 处 nit 建议，要求移除代码中多余的注释（如 "PTPC: per-channel weight+per-tokenactfp8"和"#SwiGLU-OAIalpha/beta(e.g.MiniMax-M3:1.702/1.0)"），因为代码本身已足够表明意图。提交者 hongxiayang 在后续 commit 中移除了这些注释。整体 review 获两次 APPROVED，无重大争议。

- 移除多余注释 (style): 提交者采纳建议，在后续 commit 中移除了注释。

风险与影响

- 风险：精度风险：fp8_per_channel 量化在 gsm8k 上精度损失仅 0.0031（约 0.3σ），在可接受范围内，但其他任务或更长上下文下需要额外验证。兼容性：变更只影响 ROCm 平台和 fp8 MoE 配置链，对 NVIDIA 平台无影响。回归风险：新增参数均为 Optional，默认 None，不影响现有量化路径行为。
- 影响：对 MiniMax-M3 用户在 MI300X 上：decode 吞吐提升 28%，KV cache 容量提升 75%，可支持更高并发。代码影响范围小，仅修改 5 个文件共 11 行，无测试文件变更，但依赖现有 fp8_per_channel 量化方法（已存在但未在 ROCm 上启用）。
- 风险标记：缺少测试覆盖，精度小幅下降（0.3σ）

关联脉络

- PR #45896 [feature] MiniMax-M3-MXFP4 support added: 同一模型 MiniMax-M3 的量化支持 PR，涉及类似配置链和 MoE 量化方法。
- PR #45794 [Bugfix] MiniMax-M3 (AMD): add packed_modules_mapping and pass swiglu…: 同样修复 MiniMax-M3 在 AMD 上的权重加载与参数传递问题，与本 PR 的 alpha/beta 传递修复相辅相成。
- PR #44626 [ROCm][AITER][Quark] Tag per-channel FP8 weights as PER_CHANNEL so AITER pre-shuffled GEMM is selected: 之前的 ROCm per-channel FP8 修复，为本 PR 在 ROCm 上启用 fp8_per_channel 奠定了基础。