

PR #56560 完整报告

vllm-project/vllm

[ROCm][DSv4.1][Perf] Dequantize the MXFP8 weight once when dot_scaled cannot be used

合并时间: 2026-09-15 09:07

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/56560>

执行摘要

- 一句话: MXFP8 非对齐 K 权重改为加载时一次解量化, TP4/8 解码提速约 5%
- 推荐动作: 值得精读。核心看点有三: 一是把每步重复的权重预处理上提到加载期, 属于典型的“常量折叠”优化; 二是 apply_weights 用 element_size 而非 dtype 做分派, 并保留第三分支兜底, 兼顾了正确性与灵活性; 三是测试直接用生产真实 shape (K=576/288) 做回归, 并对模块级函数做 spy 断言, 避免了只测合成 shape 的盲区。对于在 ROCm 上做量化 kernel 优化或维护 MXFP8 后端的工程师, 这份 PR 是一个很好的范本。

功能与动机

RocmDotScaledMxftp8LinearKernel 依赖 tl.dot_scaled 原生 MXFP8 GEMM, 但 dot_scaled 要求 $K \% 128 == 0$; 对不满足的权重 (如 DeepSeek-V4.1 共享专家 down_proj 在 TP4/TP8 下 K 被切成 576/288), 旧代码在 apply_weights 内每步调用 dequant_mxftp8_to_bf16 重建同一张 BF16 权重。PR body 量化了代价: dequant_mxftp8_to_bf16 是 6 个 pointwise kernel, FP32 中间值是 FP8 源的 4 倍且两次穿过 HBM, 2.95 MB 权重产生约 56 MB 流量; TP4/TP8 下 40 层全部命中, 每 decode token 产生约 2.24 GB HBM 流量。这也是 EmulationMxftp8LinearKernel 在 VLLM_MXFP8_EMULATION_DEQUANT_AT_LOAD 下已有的模式。

实现拆解

实现按以下步骤推进:

1. 定义统一对齐常量: 在 vllm/model_executor/kernels/linear/mxftp8/rocm_native.py 中新增 `_DOT_SCALED_K_ALIGN = 128`, 让加载期与推理期共用同一个 K 对齐判据, 避免两个路径各自硬编码 128 造成漂移。
2. 加载期一次性解量化: `process_weights_after_loading` 在裁剪 `weight_scale` 之后检查 $K \% _DOT_SCALED_K_ALIGN != 0$, 为真则用 `dequant_mxftp8_to_bf16` 把 FP8 权重转成 BF16 并写回 `layer.weight`; 对齐权重维持 FP8 不变。这样后续每个 forward 都不再重建常量张量。
3. 推理期按权重元素大小分派: `apply_weights` 的分派顺序改为: 先看 `layer.weight.element_size() >= 2` (BF16 直接 `torch.nn.functional.linear`), 再看激活 K 是否对齐 (走 `_mxftp8_dot_scaled_linear`), 最后保留“每步解量化”的第三分支。第三分支覆盖 `process_weights_after_loading` 未运行、权重仍是 FP8 且 K 非对齐的情况, 避免

非法进入 dot_scaled 的 128 分块逻辑。

4. 回归测试配套：在 tests/kernels/test_minimax_m3_amd_ops.py 新增 test_mxfp8_rocm_native_unaligned_k_dequantizes_at_load，参数化真实生产 shape (5120, 576)、(5120, 288) (DSv4.1 TP4/TP8) 与对齐对照 (256, 2048)；用 _spy 替换模块级 _mxfp8_dot_scaled_linear，断言 unaligned FP8 权重绝不会触发 dot_scaled。
5. 性能验证：在 4x MI355X (gfx950)、ROCm 7.2.3、TP4 上用 vllm bench serve 对比，两个未打补丁臂之间漂移仅 0.28%/0.08%，而补丁带来 4.6%/2.3% 的 TPOT 改善；profile 确认 exp2_kernel_vectorized4、CUDataFunctorOnSelf_add<float> 调用数从 2600 归零。

关键文件：

- vllm/model_executor/kernels/linear/mxfp8/rocm_native.py (模块 量化内核；类别 source；类型 core-logic；符号 _DOT_SCALED_K_ALIGN, RocmDotScaledMxfp8LinearKernel, process_weights_after_loading, apply_weights)：核心实现：process_weights_after_loading 将 K 非对齐的 MXFP8 权重在加载期一次性解量化为 BF16，apply_weights 改为按 weight.element_size() 分派并保留每步解量化 fallback，同时更新模块 docstring。
- tests/kernels/test_minimax_m3_amd_ops.py (模块 内核测试；类别 test；类型 test-coverage；符号 test_mxfp8_rocm_native_unaligned_k_dequantizes_at_load, make_layer, _spy)：新增参数化回归测试，直接覆盖 DSv4.1 共享专家 down_proj 在 TP4/TP8 的真实 shape (K=576/288) 与对齐对照 (K=2048)，并用 spy 断言 unaligned FP8 权重不会触发 dot_scaled。

关键符号：process_weights_after_loading, apply_weights,
test_mxfp8_rocm_native_unaligned_k_dequantizes_at_load

关键源码片段

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

核心实现：process_weights_after_loading 将 K 非对齐的 MXFP8 权重在加载期一次性解量化为 BF16，apply_weights 改为按 weight.element_size() 分派并保留每步解量化 fallback，同时更新模块 docstring。

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

```
# dot_scaled 的 K 维按 128 分块；K 不满足对齐的权重无法走原生 MX 路径。  
_DOT_SCALED_K_ALIGN = 128
```

```
class RocmDotScaledMxfp8LinearKernel(Mxfp8LinearKernel):  
    # 原生 CDNA4 (gfx950) MXFP8 线性 GEMM，通过 Triton tl.dot_scaled 实现。  
  
    @classmethod  
    def can_implement(cls, c: Mxfp8LinearLayerConfig) -> tuple[bool, str | None]:  
        # 注意：这里不按 K 过滤，K 对齐性由加载期的解量化处理。  
        return True, None
```

```

def process_weights_after_loading(self, layer: torch.nn.Module) -> None:
    # 加载阶段只执行一次：把 K 非对齐的 FP8 权重解量化为 BF16,
    # 避免 apply_weights 每次 forward 重建同一张量（6 个 pointwise kernel,
    # 且 FP32 中间值在 HBM 中往返两次）。
    weight = layer.weight.data # [N, K] fp8
    N, K = weight.shape
    scale_k = K // MXFP8_BLOCK_SIZE
    weight_scale = layer.weight_scale.data[:N, :scale_k].contiguous()
    if K % _DOT_SCALED_K_ALIGN != 0:
        weight = dequant_mxfp8_to_bf16(weight.contiguous(), weight_scale)
    layer.weight = Parameter(weight.contiguous(), requires_grad=False)
    layer.weight_scale = Parameter(weight_scale, requires_grad=False)

def apply_weights(
    self,
    layer: torch.nn.Module,
    x: torch.Tensor,
    bias: torch.Tensor | None = None,
) -> torch.Tensor:
    if layer.weight_scale.dtype != MXFP8_SCALE_DTYPE:
        raise ValueError(
            f"Expected {MXFP8_SCALE_DTYPE} weight_scale, got "
            f"{layer.weight_scale.dtype}."
        )
    out_shape = (*x.shape[:-1], layer.weight.shape[0])
    x2d = x.reshape(-1, x.shape[-1])
    # 分派键用元素大小而不是 dtype: BF16 (>= 2 字节) 直接走普通 linear;
    # 仍是 FP8 且激活 K 对齐时才允许 dot_scaled; 两者都不满足时退回每步
    # 解量化, 覆盖 process_weights_after_loading 未运行（权重仍为 FP8 且
    # K 非对齐）的路径, 避免非法进入 dot_scaled 的分块逻辑。
    if layer.weight.element_size() >= 2:
        out = torch.nn.functional.linear(x2d, layer.weight.to(x.dtype))
    elif x2d.shape[-1] % _DOT_SCALED_K_ALIGN == 0:
        out = _mxfp8_dot_scaled_linear(x2d, layer.weight, layer.weight_scale)
    else:
        w_bf16 = dequant_mxfp8_to_bf16(layer.weight, layer.weight_scale)
        out = torch.nn.functional.linear(x2d, w_bf16).to(x.dtype)
    out = out.reshape(out_shape)
    if bias is not None:
        out = out + bias
    return out

```

tests/kernels/test_minimax_m3_amd_ops.py

新增参数化回归测试, 直接覆盖 DSv4.1 共享专家 down_proj 在 TP4/TP8 的真实 shape (K=576/288) 与对齐对照 (K=2048), 并用 spy 断言 unaligned FP8 权重不会触发 dot_scaled。

```

# tests/kernels/test_minimax_m3_amd_ops.py
# (5120, 576) 与 (5120, 288) 是 DSv4.1 共享专家 down_proj 在 TP4 / TP8 的真实

```

```

# shape; K = 2048 保留为对齐对照，必须继续走 dot_scaled。
@requires_grad_950
@pytest.mark.parametrize("shape", [(5120, 576), (5120, 288), (256, 2048)])
@torch.inference_mode()
def test_mxfp8_rocm_native_unaligned_k_dequantizes_at_load(shape):
    # dot_scaled 按 K 128 分块，unaligned 权重绝不能到达它。
    N, K = shape
    aligned = K % 128 == 0

    w_bf16 = torch.randn(N, K, device=DEVICE, dtype=torch.bfloat16) * 0.1
    w_fp8, w_scale = _mxfp8_e4m3_quantize_torch(w_bf16, is_sf_swizzled_layout=False)
    x = torch.randn(7, K, device=DEVICE, dtype=torch.bfloat16) * 0.5
    # 参考值：先解量化再做普通 linear。
    out_ref = torch.nn.functional.linear(x, dequant_mxfp8_to_bf16(w_fp8, w_scale))

    def make_layer():
        # 构造“未经过 process_weights_after_loading”的层，weight 仍是 FP8。
        layer = torch.nn.Module()
        layer.weight = torch.nn.Parameter(w_fp8.clone(), requires_grad=False)
        layer.weight_scale = torch.nn.Parameter(w_scale.clone(), requires_grad=False)
        return layer

    kernel = rocm_native.RocmDotScaledMxfp8LinearKernel(Mxfp8LinearLayerConfig())
    layer = make_layer()
    kernel.process_weights_after_loading(layer)
    # 加载后：unaligned 权重被转成 BF16 (>= 2 字节)，aligned 保持 FP8。
    assert (layer.weight.element_size() >= 2) is not aligned

    # 仍然 FP8 的 unaligned 权重必须回退到每步解量化，而不是调用 dot_scaled。
    unprocessed = make_layer()
    calls = []
    real = rocm_native._mxfp8_dot_scaled_linear

    def _spy(*a, **k):
        calls.append(1)
        return real(*a, **k)

    rocm_native._mxfp8_dot_scaled_linear = _spy
    try:
        out = kernel.apply_weights(unprocessed, x)
    finally:
        rocm_native._mxfp8_dot_scaled_linear = real
    # aligned 时 dot_scaled 必须被调用；unaligned 时绝不能。
    assert bool(calls) is aligned

```

评论区精华

核心讨论来自 ChuanLi1101 的三轮 review，均已被作者解决：

1. 分派安全缺口 (correctness) : ChuanLi1101 指出“`apply_weights` 丢失 `unaligned fallback`。如果权重仍是 FP8 且 K 非对齐, 你会总是调用 `dot_scaled`, 这在 $K \% 128 \neq 0$ 时是非法的。加载按 $K \% 128$ 判断、应用按 `element_size` 判断, 如果两者不一致 (例如 `process_weights_after_loading` 未运行), 就会走进非法路径。请保留每步解量化 `fallback`。”作者恢复第三分支, 分派自恢复安全。
 2. 缺少测试 (testing) : ChuanLi1101 要求仿照 Emulation 已有的 `test_mxfp8_linear_emulation_bf16_at_load` 增加覆盖, 验证“加载期 BF16 与每步解量化结果一致”且“仍为 FP8 的非对齐权重不调用 `dot_scaled`”。作者新增参数化回归测试。
 3. 文档过时 (documentation) : ChuanLi1101 指出文件头仍声称非对齐 K 会经 `kernel selector fallback` 到 `emulation`, 但 `can_implement` 恒为 `True`、`selector` 根本看不到 K。作者更新 `docstring`, 改为描述加载期解量化。
 4. 测试 shape 与生产不符 (testing) : ChuanLi1101 提出“回归测试用合成 $K=2080/2048$, 实际 DSv4.1 问题 shape 是 $K=576/288$, 请至少加一个真实生产 shape”。作者加入 $(5120, 576)$ 与 $(5120, 288)$, 并移除与 288 同余的冗余 $K=2080$ 。
 5. mypy pre-commit 失败 (style) : mergify 多次提示 pre-commit 失败, 作者将 `_spy` 从 `calls.append(1) or real(...)` 的 `lambda` 改为命名函数, 修复 `list.append` 返回 `None` 导致的 mypy `[func-returns-value]` 报错。
- `apply_weights` 丢失 `unaligned fallback` 的分派安全 (correctness): 作者在 `apply_weights` 中恢复了第三分支: 仍为 FP8 且 K 非对齐时走每步 `dequant`, 分派逻辑自恢复安全。
 - 缺少非对齐 K 的回归测试 (testing): 新增参数化测试 `test_mxfp8_rocm_native_unaligned_k_dequantizes_at_load`, 并用 `_spy` 断言 `dot_scaled` 调用情况。
 - 模块 `docstring` 声称 `selector fallback` 过时 (documentation): `docstring` 改为描述“K 非对齐权重在加载期解量化一次并由普通 `linear` 服务”。
 - 回归测试 shape 与生产不符 (testing): 加入 $(5120, 576)$ 与 $(5120, 288)$, 移除与 288 同余的 $K=2080$, 保留 $K=2048$ 作为对齐对照; gfx950 上 58 passed。
 - mypy pre-commit 失败: `lambda` 返回 `None` (style): 改用 `def _spy`, 行为不变且通过 ruff/mypy 四版本检查。

风险与影响

- 风险:
 1. 显存占用增加: BF16 权重体积为 FP8 的 2 倍, 实测每个 rank 增加约 0.11 GiB。对显存极度敏感的部署需评估。
 2. 分派逻辑依赖加载流程: `apply_weights` 的正确性依赖 `process_weights_after_loading` 是否改写权重。虽然第三分支已兜底, 但未来新增其他 `dtype` 或量化路径时, 必须同步更新 `element_size` 分派, 否则可能误走路径。
 3. 测试覆盖依赖硬件: 新测试带 `@requires_gfx950`, 只能在 AMD CI 的 CDNA4 上运行; 非 ROCm 平台无覆盖 (但该 kernel 也仅在 `gfx95x` 生效)。
 4. 精度差异: `unaligned` 解量化路径与旧行为数学等价; `aligned` 路径仍走 `dot_scaled`, 其激活重量化误差与解量化路径不同, 测试中 `tol` 分别为 $5e-2$ 与 $2e-3$ 。

5. 丢失 FP8 权重形态：加载期解量化后 layer.weight 不再是 FP8，若未来 dot_scaled 支持非对齐 K，需要重新加载模型才能恢复原生路径。 - 影响：影响范围集中在 ROCm CDNA4 (gfx950) + MXFP8 量化 + K 非对齐线性层这一组合，典型场景是 DeepSeek-V4.1 在 TP4/TP8 下的共享专家 down_proj；TP1/TP2 因 K 对齐不受影响。对用户而言，解码 TPOT 中位数在并发 1 与 8 下分别下降 4.6% 与 2.3%，且消除了每次 forward 的 6 个 pointwise kernel 启动与约 56 MB 权重流量。对团队而言，该实现为同类 native kernel 的“加载期做一次数据变换 + 推理期按数据形态分派”提供了可复用模式，测试中的 spy 断言方法也可推广到其他 kernel 路径。 - 风险标记：ROCm 专有路径，分派依赖加载流程，显存占用增加，测试依赖 gfx950

关联脉络

- PR #56513 [ROCm][DSV4.1][Perf] Fold the mHC post step into the delayed pre projection: 同属 ROCm DSV4.1 解码性能优化系列，均在减少每 decode token 的 kernel 启动与 HBM 流量。
- PR #56633 [Perf][DSV4.1] Fold the mHC post block into the delayed pre projection: 同样是 DSV4.1 解码路径的算子融合优化，与本 PR 的权重解量化折叠思路一致。
- PR #56628 [ROCm][DSV4.1][Perf] Stride the DSA decode candidate mask over the live context: ROCm DSV4.1 性能优化线，改进 decode 路径的掩码写入与 HBM 流量。
- PR #51204 [Quantization] Select linear backends per quantization: 量化 linear backend 选择机制，与本 PR 中 MXFP8 linear kernel 的分派逻辑相关。